IBM Transaction Analysis Workbench for z/OS 1.3

User's Guide

SC27-6747-04

Note:

Before using this information and the product it supports, read the "Notices" topic at the end of this information.

Third Edition (October 2017)

This edition applies to Version 1 Release 3 of IBM Transaction Analysis Workbench for z/OS (product number 5697-P37) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-6747-01.

Contents

About this information

IBM® Transaction Analysis Workbench for z/OS® (also referred to as Transaction Analysis Workbench) is a tool that you can use to analyze the behavior and performance of transactions on z/OS; for example, to help you improve the performance of CICS® or IMS transactions.

These topics provide instructions for installing, configuring, and using Transaction Analysis Workbench.

These topics are designed to help database administrators, system programmers, application programmers, and system operators perform these tasks:

- Plan for the installation of Transaction Analysis Workbench
- Install and operate Transaction Analysis Workbench
- Customize your Transaction Analysis Workbench environment
- Diagnose and recover from Transaction Analysis Workbench problems
- Design and write applications for Transaction Analysis Workbench
- Use Transaction Analysis Workbench with other z/OS products

To use these topics, you should have a working knowledge of the following products:

- The z/OS operating system
- ISPF
- SMP/E (only required if you are installing Transaction Analysis Workbench)

Part 1. Getting started

These topics provide an overview of Transaction Analysis Workbench, describe how to install and configure it, and present step-by-step tutorials for using it.

Chapter 1. Transaction Analysis Workbench overview

Transaction Analysis Workbench is a tool that provides a unified platform for z/OS transactional problem management. The tool enables you to look at individual z/OS subsystems or combine them into a single unified view, including the ability to replay or track the flow of the transaction within and between various z/OS subsystems.

Transaction Analysis Workbench does not require special agent software to collect data for analysis. Instead, Transaction Analysis Workbench uses the logs and other historical data that each subsystem generates during normal transaction processing and system operations.

Transaction Analysis Workbench offers coverage across z/OS subsystems, consolidation of different subsystem logs in a single user interface, and collaboration between users. You can save information about a problem, such as the locations of log files and particular log records of interest, and then share that information with other users, enabling collaboration without rework.

Transaction Analysis Workbench significantly simplifies the process of problem determination on z/OS by making the transaction not the subsystem at the core of analysis. To facilitate this capability, log file selection is automated for each subsystem. You can use a single interface to directly view complex transactions across log sources and configure work flows that will automate the process of collecting, collating, analyzing, and resolving problems from across these subsystems.

You can use Transaction Analysis Workbench to analyze logs on z/OS in an interactive ISPF-based log browser or in batch reports, or you can forward logs off z/OS to analytics platforms such as Elastic, Splunk, and Hadoop.

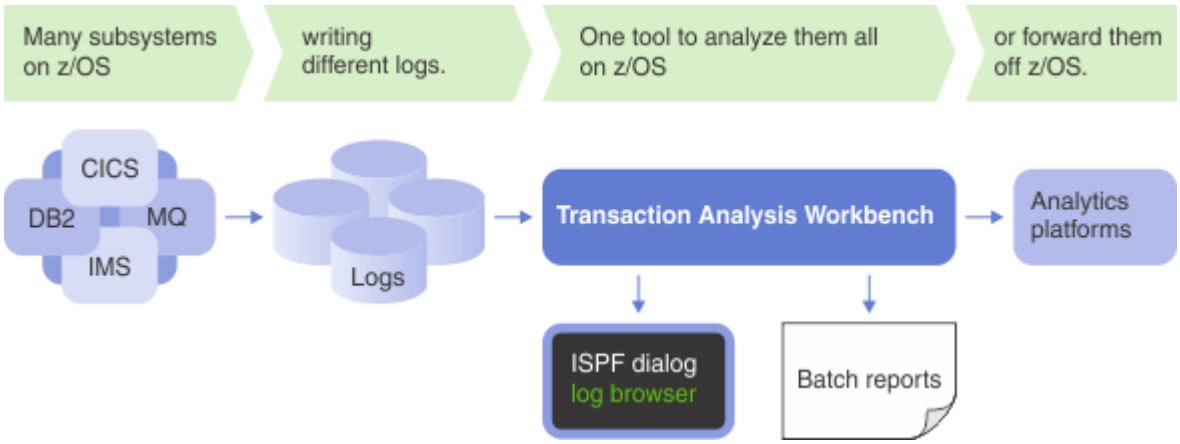


Figure 1. Analyze logs from many z/OS subsystems

What's new in Transaction Analysis Workbench

This topic summarizes technical changes in this edition.

March 2023, SC27-6747-03: updates to V1.3

IMS Version 15.3 support (APAR PH46499)

Transaction Analysis Workbench now supports IMS version 15.3.

Tracking changes for merged IMS and IBM MQ logs (APAR PH00325)

When browsing merged IMS and IBM MQ logs, a transaction tracking request initiated against an IMS log record now uses LogToken to correlate the IMS log to the MQ log. With this enhancement only MQ PUT and MQ GET log records that are directly related to the tracked transaction's input and output messages are displayed.

Previously, Transaction Analysis Workbench used the MQ unit of recovery (URID) to perform this correlation which resulted in multiple MQ PUT and MQ GET log records being displayed that are not related to the transaction associated with the selected IMS log record. Tracking requests initiated against MQ log records behave as before and use the MQ URID to correlate the IMS log to the MQ log.

See Chapter 63, “IBM MQ log extracts,” on page 293.

Previous changes for Transaction Analysis Workbench version 1.3

This topic summarizes the technical changes in previous editions for Transaction Analysis Workbench version 1.3.

October 2017, SC27-6747-02: updates to V1.3

This edition describes changes and new functions introduced in Transaction Analysis Workbench V1.3 by service since the previous edition (PTFs for the following APARs).

APAR PI82316:

Streaming logs to the Elastic Stack as JSON Lines over TCP

Previously, this documentation described forwarding logs to Elastic by running a two-step batch job:

1. The first step used Transaction Analysis Workbench to create two files in zFS:
   - Log data in CSV format
   - A corresponding Logstash config
2. The second step used Dovetailed Technologies Co:Z Co-Processing Toolkit for z/OS (Co:Z) to perform the following tasks:
   a. Transfer the two files from z/OS to the remote Elastic host
   b. Run Logstash on the remote host to forward the data to Elasticsearch

This APAR introduces a more streamlined method: a single-step batch job that forwards logs in JSON Lines format over a TCP network to an instance of Logstash that is running on the remote Elastic host.

This new method offers the following improvements:

- No need for Co:Z.
- No staging files (no "data at rest") on z/OS or the remote host.
- No record-type-specific Logstash configs. A single, concise config works for all record types.
- Single-step job with less JCL.
- Simpler ISPF dialog panel to create the JCL. If unsecure TCP meets your needs (for example, your z/OS and Elastic systems are on the same private network), you only need to fill in two fields: the Elastic host name and the TCP port on which Logstash is listening.

This new forwarding method uses features already introduced to Transaction Analysis Workbench by an earlier APAR to support streaming to Splunk. If you write your own JCL, you do not need this latest APAR to use the new method. However, this APAR makes it easier to use the new method, by replacing the complex ISPF panel that created the correspondingly complex JCL for the previous method with a simpler panel that creates the simpler JCL for the new method.

Specifically, this APAR introduces the following changes for streaming logs to Elastic as JSON Lines over TCP:

New ISPF panel for streaming logs

Previously, Transaction Analysis Workbench ISPF dialog option 5 **Analytics** offered separate menu options for Splunk and Elastic. The new menu option 5.1 **Forward data as JSON Lines over TCP** replaces those separate options, and creates JCL for streaming logs to Splunk, Elastic, or any other analytics platform that has been configured to listen for JSON Lines over TCP.

New documentation for configuring Logstash

Before forwarding logs to Elastic using this new method, you need to configure Logstash to listen on a TCP port for JSON Lines from Transaction Analysis Workbench.

Updates to the CSV, JSON, and STREAM commands of the report and extract utility

Changes to default parameter values, and related new syntax

The primary use case for the **JSON** command has evolved over time to be streaming JSON Lines over TCP. Best practice details for that use case, such as aspects of the data formatting, have also evolved. For those historical reasons, some default parameter values of the **JSON** command did not reflect that use case, so the corresponding syntax was verbose. Some defaults have been changed to streamline the syntax for that use case. These changes introduce new syntax: new parameters, or new values of existing parameters.

Previous syntax	New syntax	Notes
`LINES`	`LINES|ARRAY`	The **JSON** command now converts logs to JSON Lines by default. The previous default behavior is now specified by the new **ARRAY** parameter.
`FLAT`	`FLAT(YES|NO)`	The new syntax `FLAT(YES)` has the same effect as `FLAT`, and is now the default behavior of the **JSON** command. `FLAT` is still supported as a synonym for `FLAT(YES)`. The previous default behavior is specified by the new syntax `FLAT(NO)`.
`OMITNULL`	`MISSING(INCLUDE|EXCLUDE)`	The new syntax `MISSING(EXCLUDE)` has the same effect as `OMITNULL`, and is now the default behavior of the **JSON** command. The previous default behavior is specified by the new syntax `MISSING(INCLUDE)`.
`ASCII`	`ASCII|EBCDIC`	The new **EBCDIC** parameter enables you to explicitly specify that the **CSV** and **JSON** commands encode output in EBCDIC, which remains the default behavior. There is one new exception: if a **CSV** or **JSON** command refers to a **STREAM** command, then **ASCII** is now the default. The typical use case for streaming involves a destination that expects ASCII encoding.
`CR|LF|CRLF|NL`	`EOL(CR|LF|CRLF|NL|NONE)`	The new **EOL** parameter consolidates the previous separate parameters for end-of-line delimiters. The new syntax `EOL(NONE)` enables you to explicitly specify that the **CSV** and **JSON** commands do not append any EOL delimiter, which remains the default behavior. There is one new exception: if a **CSV** or **JSON** command refers to a **STREAM** command, then `EOL(LF)` is now the default. Without EOL delimiters, the stream would be an undelimited concatenation of records (garbage).
`FIELDCASE(ASIS|LOWER|UPPER)`	`FIELDCASE(LOWER|UPPER|ASIS)`	`FIELDCASE(LOWER)` is the new default behavior. Otherwise, this syntax is unchanged.

Previous syntax	New syntax	Notes
`ZONE`	`TIMEFORMAT(…ZZ)`	The ZONE parameter has been replaced by a trailing ZZ in the pattern-based value of the new **TIMEFORMAT** parameter.
`NOTITLE`	*None*	The NOTITLE parameter of the **JSON** command has been deprecated, and is now the default (and only) behavior. The JSON command never outputs a `title` property with the event data.

For example, the following previous syntax:

```
STREAM NAME(S1) HOST(analytics) PORT(6789)
JSON CODE(log_type:log_code) STREAM(S1) +
     LINES FLAT OMITNULL NOTITLE FIELDCASE(LOWER) ASCII LF
```

can now be specified more concisely, but still equivalently, by entirely omitting the last line (and the trailing + on the preceding line). For details on the behavior of this particular example, see Chapter 47, “Streaming logs as JSON Lines over TCP to an analytics platform,” on page 235.

The previous syntax is still supported, but has been removed from the syntax diagrams and documentation topics for these commands.

New TYPE parameter

Available on the **CSV** and **JSON** commands only; not **STREAM**.

The **TYPE** parameter specifies a string that characterizes the type of event being forwarded.

Previously, Transaction Analysis Workbench did not allow you to specify a custom value that analytics platforms could use as metadata to characterize incoming events. For example, that Elastic could use as the document type, or that Splunk could use as the source type. Instead, you were restricted to the log type and code values used by Transaction Analysis Workbench.

The **TYPE** parameter enables you to specify your own custom value for this metadata. If you omit the **TYPE** parameter, Transaction Analysis Workbench supplies a default value based on the log type and code of the event being forwarded.

Related changes:

- The **JSON** command no longer includes a `code` property in each output event. Instead, the default value of the `type` property now includes the log code.
- In Logstash configs for "transaction index records", the **CSV** command no longer uses special values for the Elasticsearch document type, such as "cics", "db2", or "ims". Instead, the document type now consistently uses the **TYPE** parameter value, such as the default values "cmf-6e13", "dtr-003", or "ims-ca01".
- In the output of the **METADATA** parameter of the **CSV** or **JSON** command, the first-level `type` property matches the value of the **TYPE** parameter.

New TYPECOLUMN parameter

Available on the **CSV** and **STREAM** commands only; not **JSON**.

The **TYPECOLUMN** parameter inserts the value of the **TYPE** parameter as the second column of output CSV data.

Some analytics platforms can use data in each incoming event to detect the event type, and then interpret the data accordingly. Inserting such data as a column in each CSV row enables analytics platforms to interpret streams of event data in CSV format that might be *heterogeneous*: different column layouts on each row.

New TIMEFORMAT parameter

Available on the **CSV**, **JSON**, and **STREAM** commands.

The **TIMEFORMAT** parameter specifies how time stamps are represented in output CSV and JSON data.

TIMEFORMAT replaces the multiple parameters that were previously used for this purpose: **THM**, **CTHM**, and **ZONE**. Those obsolete parameters still work, but have been removed from the command documentation.

TIMEFORMAT covers the same options as the obsolete parameters, with one new option: you can use the letter T instead of a blank to separate the date and time parts of a time stamp. The default blank separator matches the JDBC time stamp format; the letter T matches ISO 8601.

The pattern-based values of **TIMEFORMAT** are more transparent than the obsolete parameters; closer to what you get in the output data. For example, to output ISO 8601 time stamps with zone designators, like this:

```
2015-12-31T13:59:00.123456+08:00
```

you specify:

```
TIMEFORMAT(yyyy-mm-ddThh:mm:ss.SSSSSSZZ)
```

New `FIXEDWIDTH` parameter

Available on the **CSV** and **JSON** commands only; not **STREAM**.

The **FIXEDWIDTH** parameter right-pads field values, including numeric field values, with blanks to the maximum width of each field. This behavior is useful for destinations that require data in fixed-width columns. (Supported for JSON output, but more likely to be useful for CSV.)

`NOTITLE` parameter deprecated: `title` property no longer output in `JSON` event data

Applies only to the **JSON** command.

Previously, unless you specified the **NOTITLE** parameter on the **JSON** command, the JSON output data for each event included a `title` property that contained a brief description of the input log record type. For most use cases, this property was undesirable, because it unnecessarily increased data volume. (This information was already available in more concise form in the `type` and `code` properties; now condensed into the single `type` property.) Specifying **NOTITLE** omitted the `title` property. In practice, whenever you used the **JSON** command, you specified **NOTITLE**.

Now, the behavior previously triggered by **NOTITLE** is the default behavior of the **JSON** command. The `JSON` command never outputs a `title` property with the event data. The **JSON** command still accepts the `NOTITLE` parameter, but it has no effect, and it has been removed from the command documentation.

The record type descriptions that were provided by the `title` property are now included in the output of the **METADATA** parameter as the new first-level `description` property.

Updates to Logstash configs created by the `LOGSTASHCONFIG` parameter

Applies to the **CSV**, **JSON**, and **STREAM** commands.

Previously, the **LOGSTASHCONFIG** (or **LSCONFIG**) parameter created a Logstash config that was specifically for processing CSV data, regardless of whether the parameter was on a **CSV** command or a **JSON** command. That config could not be used to process JSON data from the **JSON** command.

Now, **LOGSTASHCONFIG** creates one of four permutations of config, depending on whether the command is **CSV** or **JSON**, and whether the command writes to a stream or a file.

Other updates to the configs created by the **LOGSTASHCONFIG** parameter:

- CSV-specific configs now remove the `message` field before output to Elasticsearch. These configs put each incoming row of CSV data into a `message` field, then parse that data into separate columns; multiple fields. If you do not explicitly remove the `message` field, the event forwarded to Elasticsearch contains the `message` field in addition to the individually parsed fields: an unnecessary duplicate of the event data. The new JSON-specific configs do

not have this problem, because they use a `json_lines` codec, which does not introduce a `message` field.

- CSV-specific configs now omit `date` filters for all time stamps fields except the `time` field. The `date` filter for the `time` field sets the value of the Logstash `@timestamp` field.

  Previously, these configs used `date` filters to convert the time stamp string values output by Transaction Analysis Workbench to the ISO 8601-format string values that Elasticsearch automatically detects as dates by default.

  Now, the new **TIMEFORMAT** parameter enables Transaction Analysis Workbench to output time stamps in ISO 8601 format, removing the need for those `date` filters. And the JCL created by ISPF dialog option 5.1 **Forward data as JSON Lines over TCP** includes a **TIMEFORMAT** parameter that outputs ISO 8601-format time stamps.
- CSV-specific configs no longer contain a `mutate` filter. The `convert` settings that were in the `mutate` filter are now in the `csv` filter. (The `csv` filter introduced support for `convert` in Logstash 2.3.)
- If the **CSV**, **JSON**, or **STREAM** command does not specify a **LOGSTASHINDEX** (or **LSINDEX**) parameter, then the **LOGSTASHCONFIG** parameter inserts a default value for the `index` setting of the `elasticsearch` output plugin.

  Previously, if **LOGSTASHINDEX** was not specified, **LOGSTASHCONFIG** would create a config without an `index` setting, and Logstash would use a default value that might not be appropriate.
- The configs set `manage_template` to false. The default index template supplied with Logstash does not refer to index patterns that match the default index names created by Transaction Analysis Workbench.

Integer values of floating-point fields now output with a trailing .0

Previously, Transaction Analysis Workbench output integer values of floating-point fields in CSV or JSON data without a trailing .0 (decimal point, then 0), for conciseness. For example, if the value of a floating-point field was exactly zero, the CSV or JSON output data would contain the value:

0

rather than:

0.0

However, some analytics platforms use the absence or presence of a decimal point in numeric values to determine which data type to use to represent fields internally: an integer data type or a floating-point data type. If the first value of a floating-point field that you forward to such a platform happens to be an integer, with no decimal point, then the platform will incorrectly map that field internally to an integer data type. The incorrect mapping can cause unexpected and undesirable behavior, such as truncation of floating-point values that you subsequently forward for that field.

To avoid such issues, Transaction Analysis Workbench now always outputs integer values of floating-point fields in CSV or JSON data with a trailing .0. For example:

- If the value of a floating-point field is exactly twelve (12), with no decimal fraction, Transaction Analysis Workbench outputs:

  12.0
- If the value of a floating-point field is exactly zero (0), with no decimal fraction, Transaction Analysis Workbench outputs:

  0.0

Zone designator for UTC output as Z

Transaction Analysis Workbench now outputs the UTC designator Z instead of the equivalent zone designators `+00:00` or `-00:00`.

STREAM command now supports the OUTLIM parameter
You can now specify the **OUTLIM** parameter of the **CSV** or **JSON** command on the **STREAM** command.

New OUTZONE global command of the report and extract utility
The **OUTZONE** command specifies the time zone of time stamps in CSV and JSON data output by the **CSV** and **JSON** commands. **OUTZONE** has no effect on time stamps in other output, such as formatted batch reports.

Previously, the time zone of time stamps in output CSV and JSON data was the default time zone specified by the **ZONE** global command.

The new **OUTZONE** command enables you to specify a different output time zone. For example, you can now choose to output time stamps in CSV or JSON data as UTC (that is, with a trailing Z zone designator), regardless of your local time zone.

ZONE global command of the report and extract utility: now allows values with a colon separator
Previously, the **ZONE** command only allowed values in the format +*hhmm* or -*hhmm*, with no colon separator between the hour and minute parts. This input value was inconsistent with the zone designators in CSV or JSON output, which do contain a colon separator. You can now specify **ZONE** command values with or without a colon.

DB2® log records: updated documentation
The description of how Transaction Analysis Workbench maps actual DB2 log codes to "umbrella" log codes has been updated and corrected.

Support for OMEGAMON® for IMS on z/OS Application Trace Facility (ATF) log streams
To process ATF log streams, specify the log stream name with the prefix ATF :

APAR PI80602:

Chapter 76, “System take-up utility,” on page 569
If you have many systems to define to Transaction Analysis Workbench, consider using the new batch system take-up utility instead of the existing ISPF panels.

APAR PI74727:

Updated support for OMEGAMON for IMS on z/OS Application Trace Facility (ATF) summary records
Transaction Analysis Workbench now supports the expanded ATF summary records introduced by APAR OA50499 for OMEGAMON for IMS, V5.3. This support includes ATF summary records that have been written to IMS logs.

November 2016, SC27-6747-01: updates to V1.3

This edition describes changes and new functions introduced in Transaction Analysis Workbench V1.3 by service since the release of V1.3 (PTFs for the following APARs).

APAR PI71601:

- Support for IBM MQ (formerly WebSphere® MQ) versions 8 and 9.
- References to WebSphere MQ have been changed to IBM MQ.

APAR PI65072 (improved support for log analytics), and earlier:

Forwarding logs to Splunk
You can stream log data in JSON Lines format to a TCP data input on a remote Splunk indexer.

For example, a batch job that runs the **JSON** command of the Transaction Analysis Workbench report and extract utility with the following combination of new parameters:

```
LINES STREAM(SPLUNK) FLAT OMITNULL NOTITLE ZONE
```

and existing parameters:

```
FIELDCASE(LOWER) ASCII LF
```

with the corresponding new **STREAM** command:

```
STREAM NAME(SPLUNK) TRANSPORT(TCP) HOST(splunk.my.com) PORT(6789)
```

outputs a network stream of JSON Lines that can be ingested by a "raw" Splunk TCP data input.

To run these commands, you can either write JCL yourself or you can use Transaction Analysis Workbench ISPF dialog primary menu option 5 **Analytics** to create JCL for you.

Loading log data into DB2

The Transaction Analysis Workbench ISPF dialog primary menu option 5 **Analytics** contains a new option for loading log data into DB2. This option creates JCL that writes DB2 DDL **CREATE TABLE** statements, and extracts logs in CSV format with a matching **LOAD** control statement for the DB2 load utility. You can submit this JCL to load data into DB2.

To support this option, the **CSV** command of the report and extract utility has a new **DB2LOAD** parameter that writes a DB2 load utility **LOAD** control statement.

Enhancements to CSV and JSON output

Many analytics platforms and other applications can ingest log data in comma-separated values (CSV) format or JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands of the report and extract utility convert logs to these formats. The **CSV** and **JSON** commands have been enhanced with the following new features:

Streaming over TCP

You can stream log data in JSON Lines or CSV format over a TCP network connection to analytics platforms such as Splunk. The new **STREAM** parameter of the **CSV** and **JSON** commands writes output to a TCP socket. The TCP connection details, which can include security (SSL/TLS), are specified by a corresponding new **STREAM** command.

Appending a zone designator to time stamps

By default, the **CSV** and **JSON** commands write time stamps without a zone designator. To append a zone designator to each time stamp, specify the new **ZONE** parameter.

Selecting all numeric fields without naming them individually

For some types of log record, most or all numeric fields might be useful for performance analysis, while only some non-numeric fields might be useful for this purpose. For these records, you might choose to select all numeric fields for output to CSV or JSON, and then select non-numeric fields individually by name as required. Selecting all numeric fields without naming them individually can also be useful for logs that frequently introduce new metrics.

To select all numeric fields, specify the new **ALLNUMBERS** parameter. To select additional fields by name, use the existing **FIELDS** or **FORM** parameter.

Scrambling character fields to make them unrecognizable

In the existing **FIELDS** parameter, you can now follow a field name with the SCRAMBLE option. The SCRAMBLE option scrambles character field values, making the original values unrecognizable.

Limiting output to a number of megabytes

The existing OUTLIM(*n*) parameter limits output to *n* records. The new **MB** parameter changes the behavior of OUTLIM(*n*) to limit output to *n* megabytes.

Logstash configuration files use `document_type` instead of `index_type`

The **LOGSTASHCONFIGURATION** parameter now creates Logstash configuration files with the `document_type` setting instead of the deprecated `index_type` setting.

Writing "schema-only" outputs without any input logs

If you specify the new **SCHEMAONLY** command, then you can use the **CSV** and **JSON** commands to produce some types of output, such as table schemas, without any input logs, based on knowledge that is built into Transaction Analysis Workbench.

CICS monitoring facility (CMF) performance fields: informal names
For CMF performance class (SMF type 110, subtype 1, class 3) records, the field names output by the **CSV** and **JSON** commands are now the informal names from the CICS dictionary, rather than the field names from the corresponding Transaction Analysis Workbench knowledge module.

Writing bit flags as separate fields
The existing **FIELDS** parameter now supports bit flag field names. Rather than parsing bit flag values from a parent field in CSV or JSON output, you can now write bit flags as separate fields.

Writing to stdout
The existing **OUTPUT** parameter supports the new value STDOUT that writes output to the standard output stream. Use `OUTPUT(STDOUT)` when you invoke the report and extract utility program, FUWBATCH, in a z/OS UNIX environment, such as a shell script, and you want to pipe log data in CSV or JSON format directly to another program rather than writing to an intermediate "staging" file.

The following enhancements apply only to JSON:

JSON Lines
By default, the **JSON** command writes a JSON array. Each element of the array is a JSON object that represents a log record.

A JSON array is appropriate in contexts that require valid JSON, such as the JavaScript `JSON.parse()` method in web browsers. However, a JSON array might not be ideal if you are writing many thousands of log records, or if several **JSON** commands write to the same destination, such as a TCP socket.

The new **LINES** parameter of the **JSON** command writes JSON Lines: a sequence of JSON objects on separate lines, where each line represents a log record.

"Flat" JSON
By default, the **JSON** command writes JSON with nested structures. Log data is nested in a `data` property; inside the `data` property, fields are nested in sections.

The new **FLAT** parameter of the **JSON** command writes JSON without nested structures, as a "flat" sequence of key-value pairs.

Omit fields with null value
By default, the **JSON** command writes the JavaScript value `null`, without enclosing double quotation marks, for fields that have been selected for output but are missing from the input log record.

The new parameter **OMITNULL** omits such fields from JSON output.

Omit the title property
By default, the **JSON** command writes a `title` property for each event. The `title` property contains a short description of the original log record.

The new **NOTITLE** parameter of the **JSON** command omits the `title` property.

ISPF dialog primary menu option 5 Big Data enhanced and renamed Analytics
Transaction Analysis Workbench ISPF dialog option 5 has been renamed from **Big Data** to **Analytics**. The panels have been redesigned, introducing new options that create JCL to perform the following tasks:

- Stream logs in JSON Lines format over TCP to Splunk
- Load logs in CSV format into DB2
- Write logs in JSON or CSV format to an MVS™ sequential data set, z/OS UNIX file, or SYSOUT

New data source: CICS VSAM forward recovery and autojournaling records
Transaction Analysis Workbench now supports CICS VSAM forward recovery and autojournaling records as a new log type, VSR. Transaction Analysis Workbench can read these records either directly from a log stream or from a data set that contains records extracted from a log stream.

To browse a log stream of these records in Transaction Analysis Workbench ISPF dialog option 4 **Process**, specify the log stream name with the prefix VSAM followed by a colon. For example:

```
VSAM:RGNUSR.APPLA.GRP1
```

To add a log stream or data set of these records to a session in ISPF dialog option 1 **Sessions**, select option 3 **Files**, and insert a file of type JOURNAL for the corresponding CICS system.

June 2015, SC27-6747-00: V1.3

This edition describes changes and new functions introduced in Transaction Analysis Workbench V1.3.

Big data analytics

You can use Transaction Analysis Workbench to extract, transform, and load z/OS-based logs into big data applications.

In addition to extracting and transforming logs into popular data formats such as JSON and delimiter-separated values (DSV), including comma-separated values (CSV), Transaction Analysis Workbench V1.3 creates metadata and configuration files specifically for use with Logstash or software based on Hadoop, such as IBM BigInsights® and Cloudera.

The new Transaction Analysis Workbench ISPF dialog primary menu option 5 **Big Data** creates JCL that uses the Transaction Analysis Workbench report and extract utility to extract log data to CSV and, optionally, create the corresponding files for use with Logstash or Hadoop. Also optionally, the dialog creates JCL that loads the extracted data onto remote systems running your big data applications.

Support for Mobile Workload Pricing for z/OS (MWP)

The new **MWP** command of the Transaction Analysis Workbench report and extract utility creates customer mobile CPU consumption CSV (comma-separated values) files for use with the Mobile Workload Reporting Tool (MWRT).

Enhancements to extracting log records to CSV

Enhancements to comma-separated values (CSV) processing include support for repeating sections.

Extract log records to JSON

You can now extract any log records supported by Transaction Analysis Workbench to JavaScript Object Notation (JSON) format.

Filter CICS trace records by trace level

As an alternative to using filters to explicitly exclude specific CICS trace point IDs, you can set a trace level to exclude CICS trace records in the log browser.

New data source: Tivoli OMEGAMON XE for DB2 Performance Expert near-term history

You can analyze DB2 trace records that have been written to a Tivoli OMEGAMON XE for DB2 Performance Expert near-term history file.

IMS version 14

Support for the new instrumentation in the IMS log and monitor.

Changes that affect the Transaction Analysis Workbench plug-in

The server component that enables the plug-in to communicate with Transaction Analysis Workbench has a new name and is packaged in a new product:

- IBM Functional Support Library Server is now known as Common Services Library server.
- Common Services Library server is a component of a new product, IBM Common Services Library for z/OS (Common Services Library), V1.1.

  Common Services Library server supersedes the IBM Functional Support Library Server component of IBM Tools Base for z/OS (Tools Base), V1.5.

Changes for Transaction Analysis Workbench version 1.2

This topic summarizes the technical changes in previous editions for Transaction Analysis Workbench version 1.2.

October 2014, SC19-4006-01: updates to V1.2

This edition describes changes and new functions introduced in Transaction Analysis Workbench V1.2 by service (PTFs for the following APARs).

APAR PI25744:

Changes that affect the Transaction Analysis Workbench plug-in

- The following changes in IBM Tools Base for z/OS, V1.5 (Tools Base) affect the Transaction Analysis Workbench plug-in:
  - Tools Base no longer includes an Eclipse rich client platform (RCP) executable into which you install the plug-in. Instead, you install the Transaction Analysis Workbench plug-in into IBM Explorer for z/OS (z/OS Explorer).
  - Tools Base no longer includes a framework common plug-in that you need to install before installing the Transaction Analysis Workbench plug-in. That functionality is now included in the plug-in software site archive file supplied with Transaction Analysis Workbench.
  - The Tools Base Connection Server, which is the z/OS-based software that enables the plug-in to communicate with Transaction Analysis Workbench, has been renamed IBM Functional Support Library Server.
  - Most of the identifiers that were FUD-prefixed are now FUN-prefixed. For example:
    - The IBM Functional Support Library Server program name is FUNSRV. Related ddnames are also now FUN-prefixed.
    - IBM Functional Support Library Server messages are FUN-prefixed.
  - Tools Base V1.5 does not contain a separate user's guide for IBM Functional Support Library Server corresponding to the *Connection Server User's Guide* that was supplied with Tools Base V1.4. Except for the initial SMP/E installation procedure in the *Tools Base Program Directory*, Tools Base V1.5 contains no documentation for IBM Functional Support Library Server. Instead, IBM Functional Support Library Server documentation is included in the Transaction Analysis Workbench installation topics.

APAR PI26435 for IBM Functional Support Library Server, only relevant if you use the Transaction Analysis Workbench plug-in:

IBM Functional Support Library Server security
The high-level qualifier of the general resource profile that the server uses to perform security checks has changed from FUDPRD to FUNPRD.

May 2014, SC19-4006-00: V1.2

This edition describes new functions introduced in Transaction Analysis Workbench version 1.2:

DB2 trace records: enhanced support
Previously, Transaction Analysis Workbench formatted only a few IFCID (trace event) records, and only if they had been written to SMF. For example, DB2 accounting records (IFCID 3) were presented as log type SMF, log code 66. Now, Transaction Analysis Workbench provides specific formatting for many more IFCIDs, written either to SMF or to the generalized trace facility (GTF). DB2 trace records are presented as the new log type DTR, regardless of whether the records were written to GTF or SMF. The DTR log codes are the IFCIDs formatted as a 3-digit decimal number with leading zeros. For example, DB2 accounting records are presented as log type DTR, log code 003.

Recognizing that collecting DB2 trace records can be expensive and result in large volumes of data, Transaction Analysis Workbench introduces the concept of DB2 *trace levels*. Each trace level defines a set of IFCIDs. You can use trace levels to help decide which IFCIDs to activate and also to filter IFCIDs

that have already been collected, depending on the level of detail that you want to analyze. To filter the display of IFCIDs in the ISPF dialog log browser, enter the **TRACE** command.

DB2 accounting exception reports and extracts
You can report and extract DB2 accounting (SMF type 101) records that meet or exceed thresholds that you specify for common performance measurements, such as response or CPU time.

DB2 log records: improved classification and descriptions
The log codes and descriptions for DB2 log records (log type DB2) have been improved to offer more detail.

DB2 11 support
Transaction Analysis Workbench now supports log records and trace records written by DB2 11 for z/OS. This support includes the new extended 10-byte format for relative byte address (RBA) and log record sequence number (LRSN).

CICS TS 5.2 support
Transaction Analysis Workbench now supports CICS Transaction Server for z/OS, V5.2.

Session workflows
A session workflow is a sequence of tasks in a session. Each task consists of either JCL for a batch job or a note offering instructions or tips to the user. Typical batch job tasks select log files, create extracts of log files, or generate reports. A note task might describe how to analyze a report that was created by a previous batch job task. You can perform tasks individually or you can schedule multiple tasks to run in sequence.

Session templates
A session template is a blueprint for a session. When users create a session, they can optionally select a template. A session template can offer a starting point and guidance for analyzing a particular type of problem, such as a problem with a particular application. Session templates are typically created by subject-matter experts.

Eclipse plug-in
The Eclipse plug-in for Transaction Analysis Workbench is aimed at "first responders", such as help desk staff, who record the details of problems reported by users of z/OS-based transactions. You can use the plug-in to perform the following functions:

1. View or create sessions
2. Submit batch jobs on z/OS for session workflow tasks
3. View batch job task output

Automated SMF file selection
If your CICS, DB2, IBM MQ, or MVS image system definitions specify an SMF file generation data group (GDG) base, data set name pattern, or log stream, then you can now use Transaction Analysis Workbench to locate the SMF files or log stream that cover the time period when a problem occurred.

SMF 42.6 DASD Data Set I/O report
The DASD Data Set I/O report uses SMF type 42, subtype 6 records to analyze the performance of data set I/O, including read and write frequency, DASD response time, and cache usage. You can specify selection criteria to limit the report to job names and data set names that match particular patterns. These selection criteria enable you to create reports for specific systems, or subsystems, and their data sets. For example, to analyze the performance of DB2 data set I/O, you could specify a job name pattern of DB2* and a data set name pattern of DSNDB*.

Create IMS transaction indexes without IMS Performance Analyzer for z/OS
You can now use Transaction Analysis Workbench to create IMS transaction indexes from the IMS log files of any IMS environment. Previously, Transaction Analysis Workbench could create IMS transaction indexes only for DBCTL environments, for input to CICS-DBCTL reporting; to create IMS transaction indexes for other IMS environments, you needed to use IMS Performance Analyzer for z/OS.

Create CICS-DBCTL reports from the ISPF dialog
Previously, to create CICS-DBCTL reports, you had to edit JCL. You can now create these reports from the ISPF dialog: from a session menu, select option 4 **Reporting**.

ISPF dialog usability enhancements

To select a session, enter its key on the command line of the Session Manager panel

As an alternative to entering line action S next to a session, you can now select a session by entering its key on the command line of the **Session Manager** panel. You can omit leading zeros from the key. For example, to select session 00000042, enter 42. You can optionally prefix the key value with the command **S** followed by a space: for example, S 42.

Improved file duration calculation for small log files

Previously, if the size of a log file was less than one complete disk track, the **Investigate** panel displayed the file duration as <1 TRACK. Now, the panel displays a time value.

IMS Connect event 2-byte log codes

Previously, for IMS Connect events, Transaction Analysis Workbench used the log code A0*xx*. The first byte, A0 or some other fixed value used by IMS Connect, identified the record as an IMS Connect event; the second byte identified the specific event. Now, Transaction Analysis Workbench omits the leading fixed prefix (A0); the log code matches the full 2-byte IMS Connect event code. For example, the old log code A042 (Message received from OTMA) is now 0042. Extended IMS Connect events have a log code with a nonzero first byte. For example, 0802 (ISC communications thread started).

Filters and forms now refer to IMS Connect log records by their full 2-byte code.

Log browser usability enhancements

To switch the filter on or off, press F6

Instead of entering the primary commands `FILTER ON` and `FILTER OFF`, you can now press function key F6 to flip (toggle) the filter on and off. Pressing F6 is a shortcut for entering the new primary command `FILTER FLIP`.

To view or edit the filter, press F18

As a shortcut to entering the primary command `FILTER`, which displays the **Filter** panel, you can now press function key F18.

Tip: Most modern keyboards lack a dedicated F18 key. Instead, terminal emulators support keyboard mapping, allowing you to press an alternative key or combination of keys to reproduce the effect of pressing F18. The typical default key combination for F18 is Shift+F6.

Streamlined log browser navigation controls

The **Slice** field has been removed from the log browser navigation controls. To change the current time slice, exit the log browser and set the time slice on the **Investigate** panel. Removing the **Slice** field streamlines the navigation controls in the log browser and eliminates any doubt regarding the time and duration of the current time slice.

To scroll to a particular date or time, you can now simply change either the date or the time value, or both, displayed next to the **Date/Time** label, and then press Enter. After scrolling to other records, to return to that date and time, select the **Date/Time** label, which is a point-and-shoot field.

Reasons for bottom of data

If the log browser reaches the end of the selected log files, the bottom of data marker after the last displayed record is a line of asterisks with the label `Bottom of Data`. If the log browser does not reach the end of the selected files, the label now offers one of the following reasons:

- `FINDLIM reached (`*`find limit`*`)`
- `ATTN interrupt`
- `TIMEOUT reached (`*`timeout value`*`)`
- `DURATION reached (`*`time slice duration`*`)`

Set color and highlighting according to log type

To customize the color and highlighting of records according to their log type, enter **HILITE** or select **Options** > **Color highlighting**.

Prepend log sequence number (LSN) with log type
To prepend the LSN with the log record type, enter **DISPLAY** or select **Options** > **Display**, and then set the **Display LSN** option.

System groups
You can define groups of systems, enabling you to perform actions on multiple related systems by selecting a single group. For example, you can create a report for multiple systems by selecting a single group, or add multiple systems to a session by selecting a group. There are two types of groups:

- Groups of IMS and IMS Connect systems. These group definitions are stored in the IMS system definition repository.
- Groups of CICS and related systems, including DB2, WebSphere MQ, and MVS images. These group definitions are stored in the CICS and related systems definition repository.

REXX API returns the record log type, not just the log code
The ALZEXEC host command environment now predefines the REXX variable `ALZ.LOGTYPE` containing the log type of the current record. Similarly, the "READ" on page 614 command sets the variable *stem*`.LOGTYPE`.

Changes for Transaction Analysis Workbench version 1.1

This topic summarizes the technical changes in previous editions for Transaction Analysis Workbench version 1.1.

January 2013, SC19-2920-02: updates to V1.1

This edition describes new functions introduced in Transaction Analysis Workbench version 1.1 by service (PTFs for the following APARs).

APAR PM75605:

IMS version 13
Support for the new instrumentation in the IMS log and monitor.

Save log records displayed in the log browser to an extract data set
To create an extract containing the records displayed in the log browser, enter **EXTRACT** on the command line of the log browser. For details, see the online help: in the log browser, press the Help function key (F1).

Support for WebSphere Application Server for z/OS request activity performance statistics
Transaction Analysis Workbench now provides specific formatting support for SMF type 120, subtype 9 records (log code 7809).

APAR PM69954:

New data source: CICS trace entries in GTF data sets
You can analyze CICS trace entries that have been written to a z/OS generalized trace facility (GTF) data set.

New data source: IMS Connect transaction index
An IMS Connect transaction index is a data set that IMS Performance Analyzer for z/OS creates from an IMS Connect Extensions for z/OS journal. Each index record represents an IMS Connect transaction recorded in the journal, and includes all the cumulative information from the journal about that transaction.

Filter field: qualify log code with log type
The **Filter** field on the **Process** panel (for ad hoc log files) and the **Investigate** panel (for sessions) enables you to restrict the log records initially displayed by the log browser. You can specify either a filter name or a log code in this field. You can now explicitly qualify the log code with its log type, by specifying the log type, a colon, and then the log code (for example, `IMS:08`).

To filter CICS trace entries using this method, you *must* qualify the log code (for example, `CTR:AP`), otherwise the **Filter** field interprets the alphanumeric log code of the CICS trace entry as a filter name.

Exclude session log files from interactive investigation

On the **Locate and Manage Log Files** panel for a session, enter line action F to "flip" (toggle) the file exclude status. Excluded files do not appear on the **Investigate** panel for the session.

Extract data set name templates

Rather than requiring you to explicitly specify a data set name for each extract that you create, the Transaction Analysis Workbench ISPF dialog suggests an extract data set name that is based on a template. To edit the template, go to dialog option 0.3 **Extract Data Set Allocations**.

Specifying files for a session: quicker method for specifying multiple new files for the same system

The **Specify File Details** window now allows you to specify multiple data sets for the same system.

APAR PM65787:

XT line action in log browser to exclude a log type

When browsing logs, you can now hide all records of a particular log type by entering line action XT next to a record of that type. That record, and all other records of that type, are excluded from the display. XT inserts a line in the current temporary filter to exclude a log type. To reinstate these records, enter RESET XT on the command line.

TRACE command for IMS trace records

IMS log records of type 67FA, and 67FF records for IMS trace tables, contain IMS trace table entries. Rather than displaying these 67FA or 67FF IMS trace table records, the log browser now formats each IMS trace table entry individually (with a new log record type, ITR). To display IMS trace entries when browsing logs, enter TRACE ON; to hide them, enter TRACE OFF.

When creating extracts, you can use the **TRACE** batch command to specify whether to process 67FA and 67FF IMS log records in their original format or as individual trace entries.

FIND primary command for selection prompt lists and list panels

In many selection prompt lists (that appear when you press the Prompt function key (F4) on a field) and list panels, you can now use the **FIND** primary command to scroll forward to the next line containing your specified string. For example, if you enter F DBCTL on the command line of the **Session Manager** panel, the panel scrolls forward to the next session that contains "DBCTL" (in its description, or any other column on the panel). To find the next occurrence, press the Repeat Find function key (F5).

APAR PM54715:

CICS-DBCTL exception analysis

New batch reports display information from CICS monitoring facility (CMF) performance class (SMF type 110) records and IMS log records to help diagnose problems with CICS-DBCTL transactions.

Brief format for formatted record reports

The brief format reproduces the format of the log browser list panel, where each log record occupies a few lines at most, showing only general and high-level information such as the log record code, description, and global fields.

APAR PM50250:

IMS version 12 support, and updates to support for IMS versions 10 and 11

Support for all the new instrumentation in the IMS log and monitor:

- IMS V10: Transaction accounting record 56FA containing accurate transaction CPU time, DLI call, database I/O, and ESAF usage
- IMS V11: Synchronous call-out (ICAL) and Open Database
- IMS V12:
  - 4517 User Exit statistics
  - Improved MSC statistics
  - Repository Audit log stream

Support for all new IMS V12 Connect event records collected by IMS Connect Extensions event journaling.

May 2011, SC19-2920-01: updates to V1.1

This edition describes new functions introduced by the PTF for APAR PM26786:

Bookmark log records with permanent tags

When you return to a session, you can now choose to resume browsing logs from where you left off. While browsing logs in a session, you can now tag a record, and then you or other users can later resume browsing at that position by selecting the tag in the session history.

In addition to tags that users have explicitly created while browsing logs, each session history contains a tag for the most recent browsing position of each user, enabling you to resume browsing where another user has left off.

Resuming from a tag resumes not just the browsing position, but also other aspects of the browsing state, such as the active filter and selected log files.

Log streams now supported as log files

In addition to analyzing log records in data sets, you can now use Transaction Analysis Workbench to analyze log records in log streams. Transaction Analysis Workbench can format log records in the following types of log stream:

- IMS Common Queue Server (CQS) log streams: both message queue (MSGQ) primary (and optional overflow) structures and expedited message handler queue (EMHQ) primary (and optional overflow) structures
- OPERLOG: the z/OS operations log
- SMF log streams

You can also use Transaction Analysis Workbench to analyze any other log stream, but in dump format only (with no record formatting).

To distinguish log stream names from data set names, log stream names in Transaction Analysis Workbench have one of the following prefixes:

CQS:
OPERLOG:
SMF:
OTHER: (any other type of log stream; no record formatting)

For example:

```
OPERLOG:SYSPLEX.OPERLOG
SMF:IFASMF.PROD1
```

Records from OPERLOG and OTHER log streams have the log type MVS; OPERLOG records have the log code CA52, OTHER log stream records have the log code CAFF. Records from CQS and SMF log streams have the same log type and code as records from CQS extract data sets and SMF data sets, respectively.

Log streams with the prefix OTHER: are supported only in batch jobs and on the **Process Log Files** panel (ISPF dialog option 4 **Process**); they are not supported in sessions (option 1 **Sessions**).

Transaction Analysis Workbench refers to log data sets and log streams collectively as *log files*.

WebSphere MQ thread-level and queue-level accounting: new SMF report

A new report for SMF type 116, subtype 1 and 2 records shows WebSphere MQ class 3 accounting information (thread-level and queue-level accounting). To collect this information for reporting, you must use the WebSphere MQ **START TRACE** command for accounting class 3. To request one of these new reports in the ISPF dialog:

1. Define a session
2. Select option 4 **Reporting** from the session menu
3. Select 4 **SMF**
4. Select the **WebSphere MQ thread accounting** report

To request one of these reports by writing your own JCL for the report and extract utility, specifying the command **REPORT SMF(116)**.

Filters can now be stored in the control repository for later reuse
Previously, you had to define filters each time you used the ISPF dialog to browse logs. The new dialog option 2 **Controls** enables you to define filters in the control repository, and then refer to them by name.

Forms: hiding fields that are not of interest to you
To hide fields that are not of interest to you, define and then use a form. A form consists of the list of fields in a specific type of log record, with some fields removed. You can use forms in the ISPF dialog when browsing a log record, and when using the report and extract utility to create a formatted record report or a CSV file (see the following item). To define forms, select dialog option 2 **Controls**.

Comma-separated values (CSV) file output
You can save log records to CSV files for use with other applications, such as PC-based spreadsheet applications. A CSV file created by Transaction Analysis Workbench contains selected fields from one type of log record, as specified by a form.

IMS user log records
IMS applications can use the DL/I **Log** call to write records to the IMS system log. Such records are known as IMS user log records; they have a log code of X'A0' or greater, which is outside the range of log codes used by IMS itself. To enable Transaction Analysis Workbench to process and format IMS user log records, you create knowledge modules for the corresponding log codes.

REXX API
Transaction Analysis Workbench provides a REXX application programming interface (API) that you can use to format and analyze logs.

Process Log Files panel for analyzing ad hoc log files
Previously, before you could start analyzing logs in the ISPF dialog, you had to register a session, and then add log files to the session. The new dialog option 4 **Process** enables you to analyze log files without defining a session. You can use this option to analyze log files privately before registering a session that you share with other users.

Time slicing: now integrated into the log browser
Previously, to define a time slice (a time range of the log records that you want to analyze, from one or more log files that might cover a longer period), you had to specify the time slice details on a separate ISPF panel before browsing logs.

Sessions: Investigate panel tolerates unavailable/uncataloged data sets
Previously, if a session defined log files that were unavailable or uncataloged, you could not access the **Investigate** panel for that session. Now the **Coverage** column of the **Investigate** panel displays error information for these files.

Timeout setting in the log browser
To avoid long delays caused by the log browser reading log files to locate all records required for display, you can now set a timeout. In the log browser action bar, select **Options** > **Timeout...**. You can set a timeout value of 1 - 99 seconds, or 0 for no timeout.

October 2010, SC19-2920-00: V1.1

First edition for first product release.

What does Transaction Analysis Workbench do?

Transaction Analysis Workbench interprets a wide range of logs from many z/OS subsystems, and consolidates them in a consistent, feature-rich environment.

You can use Transaction Analysis Workbench to identify and analyze transaction problems quickly, without requiring an expert understanding of log data structures and the relationships between log records.

Transaction Analysis Workbench interprets the following types of log record, listed here by system or subsystem.

CICS

- CICS monitoring facility (CMF) performance class (SMF type 110, subtype 1, class 3)
- CICS trace entries that have been written to a z/OS generalized trace facility (GTF) data set or the CICS auxiliary trace (DFHAUXT)

DB2

- DB2 log
- DB2 trace records that have been written to one of the following destinations:
  - GTF data set
  - SMF (as SMF type 100, 101, or 102)
  - Tivoli OMEGAMON XE for DB2 Performance Expert near-term history

IMS

- IMS log and trace
- IMS transaction index
- IMS monitor and DB monitor
- Common Queue Server (CQS) log stream
- IMS Connect event data, collected by IMS Connect Extensions
- IMS Connect transaction index: created by IMS Performance Analyzer from IMS Connect event data in an IMS Connect Extensions journal
- Tivoli OMEGAMON XE for IMS (OMEGAMON for IMS) Application Trace Facility (ATF):
  - Journal
  - Log stream
  - Summary records written to an IMS log
- Internal resource lock manager (IRLM) long lock detection (SMF type 79, subtype 15)

IBM MQ

- IBM MQ log extract
- IBM MQ statistics (SMF type 115, subtypes 1 and 2)
- IBM MQ accounting (SMF type 116)

WebSphere Application Server for z/OS

- WebSphere Application Server for z/OS request activity performance statistics (SMF type 120, subtype 9)

z/OS

- Selected SMF record types applicable to problem analysis (including RMF, APPC, job-level accounting, and VSAM activity), in either log streams or dumped SMF data sets
- OPERLOG, the z/OS operations log (log stream)

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Related reference

Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system

that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

SMF codes
The SMF log type consists of log codes that identify MVS System Management Facilities records.

Log stream types
Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

Transaction Analysis Workbench features

Transaction Analysis Workbench contains features to help simplify problem analysis across a wide range of z/OS subsystems.

Interactive log browsing

The Transaction Analysis Workbench ISPF dialog user interface includes a log browser that offers a consistent user interface for all supported record types, with features for navigating, finding, filtering, bookmarking, formatting, and extracting log records. You can select any combination of supported logs and browse them merged in a single view.

To start browsing, you select one or more logs and, optionally, a time period. The log browser displays summarized records from the logs. You can select a single record to view all of its fields, and then zoom on a field for a description of that field and its values.

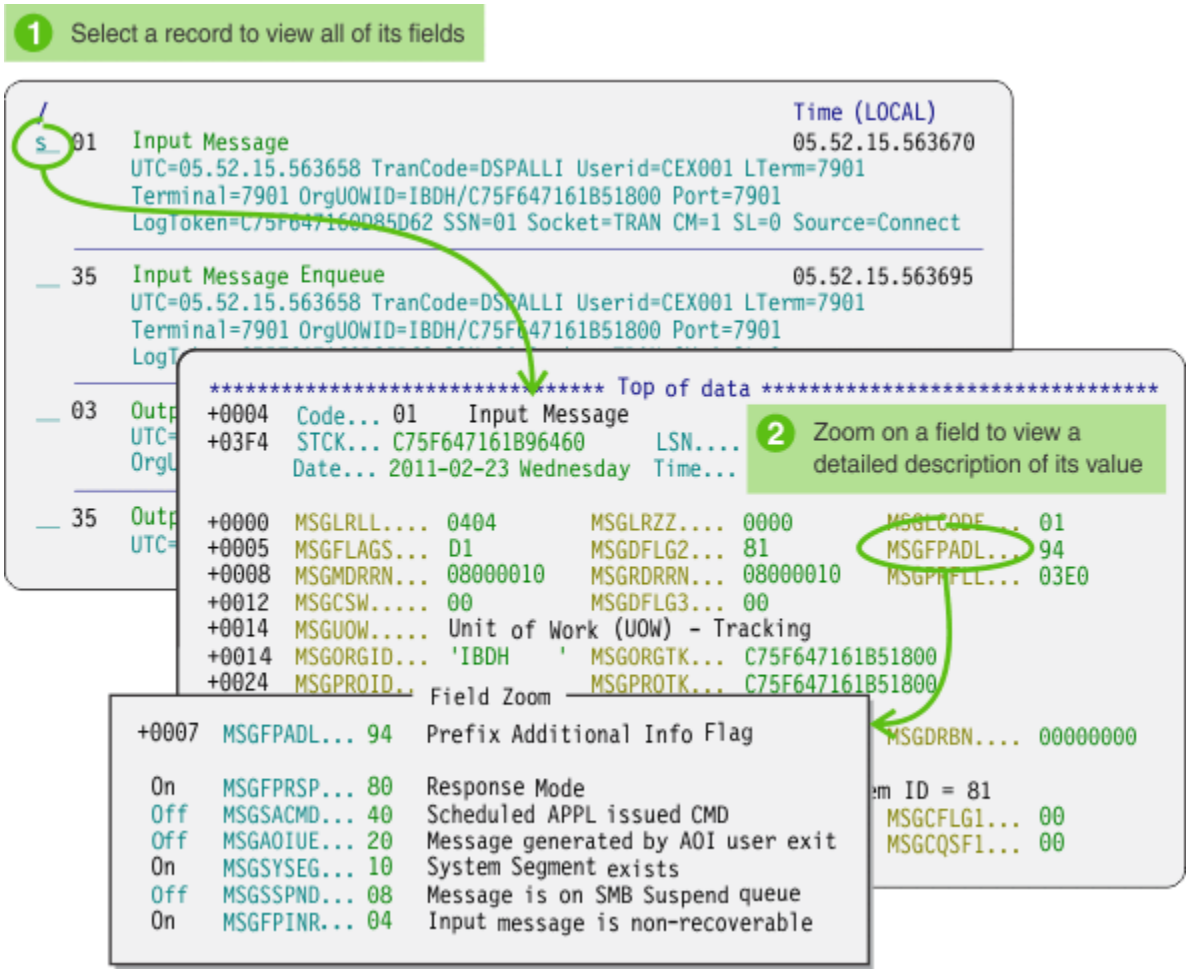


Figure 2. Example interactive analysis of an IMS log file

By providing a consistent user interface for all supported record types, the log browser makes it easy to extend your analysis into record types and subsystems that you might not be familiar with. Learn how to use the log browser to analyze log records from subsystems that are familiar to you, then reuse those skills to analyze other record types.

Related tasks

Browsing logs interactively
You can use the Transaction Analysis Workbench ISPF dialog to browse records from different logs merged in a single view. You can select a record to browse its fields, and then select a field to browse a detailed description of its contents.

Transaction tracking

Transaction tracking identifies the significant events in the lifecycle of a single transaction across subsystems and logs.

Identifying the log records that are related to an individual transaction can be difficult:

- The log records for a transaction are typically interspersed among records for other transactions.
- The log records for a transaction are correlated by a variety of tokens.
- The log records for heterogeneous transactions, such as CICS-DB2 transactions (CICS transactions that call DB2), are dispersed across different logs written by multiple subsystems.

The Transaction Analysis Workbench log browser makes transaction tracking easy: you simply enter TX next to any log record, and the log browser displays the records that are related to the same transaction, hiding all other log records. The log browser tracks the transaction across all available logs.

For example:

- CICS with DBCTL: from a CICS transaction record, view the related IMS log records
- CICS or IMS with DB2: from a CICS or an IMS transaction record, view the related DB2 accounting and trace (IFCID) records, and DB2 log events
- CICS or IMS with IBM MQ: from a CICS or an IMS transaction record, view the related MQ accounting records (SMF) and MQ log (extract) events
- DB2: track the lifecycle of an individual thread across DB2 log records and IFCID records
- IMS Connect: view IMS transaction and DRDA ODBM database activity

The term *transaction* is used here loosely. For DB2, in situations where there is no driving transaction from another system such as CICS or IMS, Transaction Analysis Workbench tracks DB2 *threads*.

For IMS, you can track a unit of recovery (UOR) within a transaction by entering TU next to a log record.

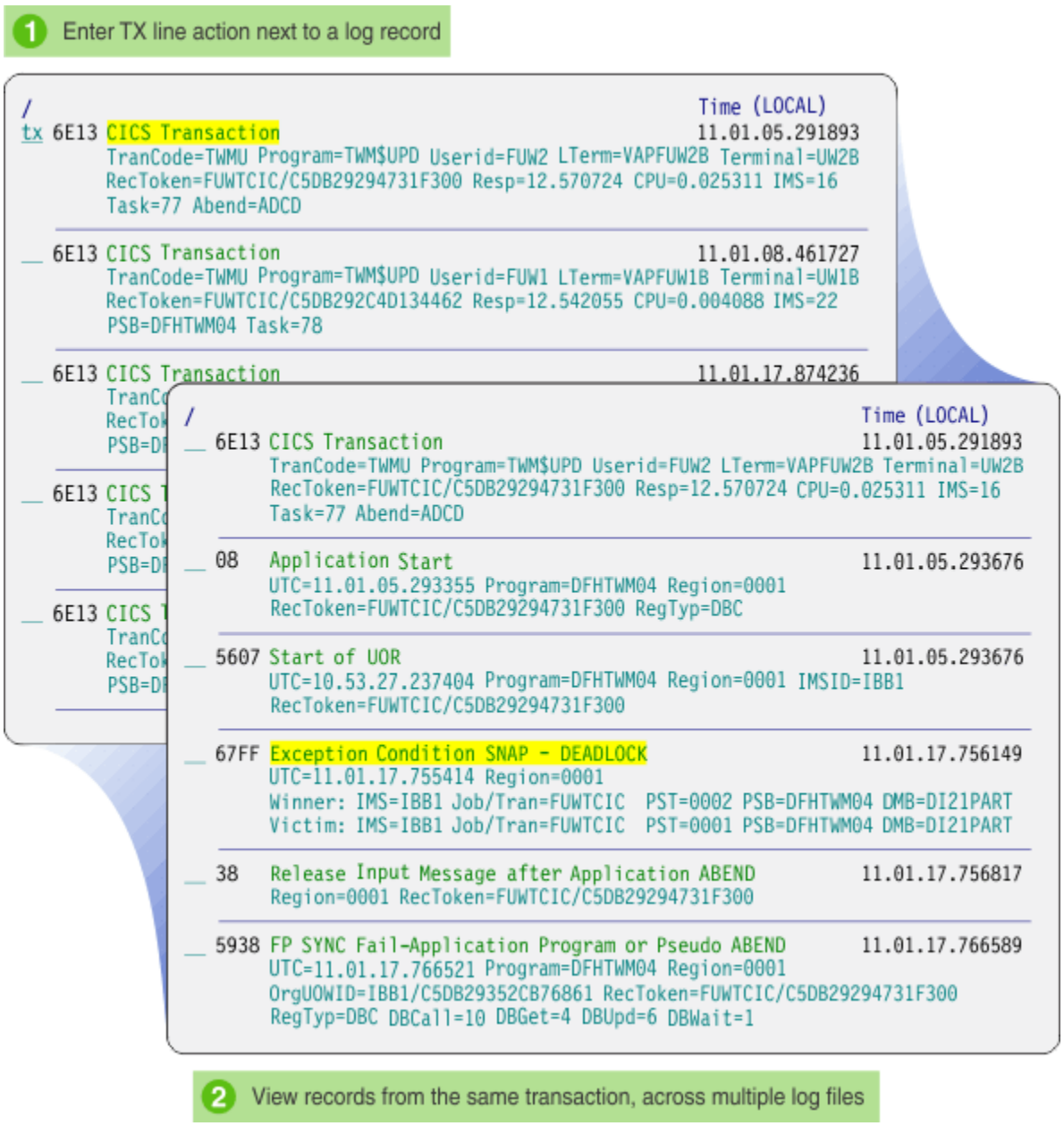


Figure 3. Transaction tracking example: a CICS-DBCTL transaction that causes a deadlock in IMS

Tracking simplifies and accelerates problem analysis by condensing all of the available log records for a transaction (or thread, or unit of recovery) in a single view.

Tracking can help both expert and non-expert users to analyze problems.

For example, suppose you have applied a filter in the log browser to display log records that report a long response time, or excessive CPU time, or some other metric with an abnormally high value. Tracking (entering TX next to one of those records) can help to analyze the problem in the following ways:

- Tracking can uncover a related log record that reports the specific exception condition that is responsible for the abnormal value. Non-expert users can then refer to the documentation for that exception condition.
- Tracking displays log records that correspond to events in the transaction lifecycle. Elapsed times between those log records typically correspond to elapsed times between events. Often, you don't need an expert understanding of the events in a transaction to pinpoint which event caused the problem: just look for the log record with the longest elapsed time.

Tracking is not just useful for analyzing problems. Tracking can help to educate developers and other technical staff about the behavior of their own systems by showing a time line of events in a transaction.

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Related tasks

Tracking a transaction or an IMS unit of recovery

Tracking displays log records that are related to a particular transaction. For IMS transactions, you can track log records that are related to a particular unit of recovery (UOR) within a transaction.

Log forwarding

You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

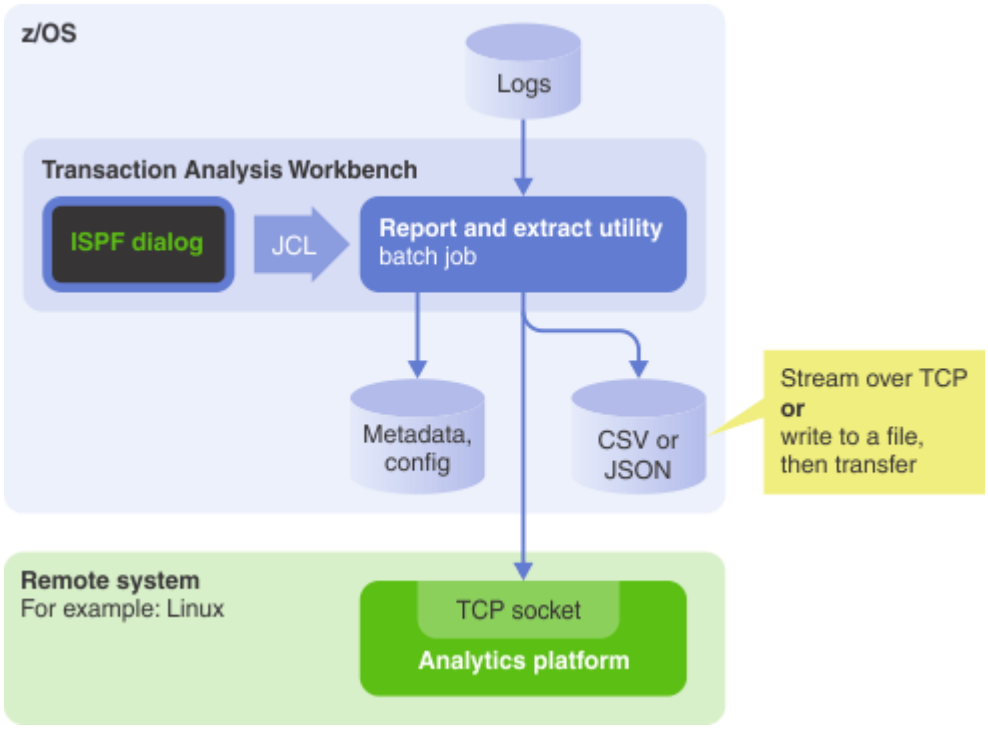


Figure 4. Forwarding logs to analytics platforms

Data format: CSV, DSV, JSON, or JSON Lines

You can forward logs in common formats supported by many analytics platforms: comma-separated values (CSV), other delimiter-separated values (DSV) formats, JavaScript Object Notation (JSON), or JSON Lines.

Note: For brevity, in this Transaction Analysis Workbench documentation, the term CSV includes DSV, and JSON includes JSON Lines, except where these different terms are required to distinguish between these related formats.

Batch forwarding

To convert logs to CSV or JSON, you submit a z/OS batch job that runs the **CSV** or **JSON** command of the Transaction Analysis Workbench report and extract utility.

Streaming versus file transfer

The batch job that converts logs to CSV or JSON can either use the **STREAM** command of the report and extract utility to stream the output to a TCP socket on a remote system, or write to "staging" files on z/OS, and then, in a subsequent job step, transfer the files to a remote system.

JCL: Write your own or use the ISPF dialog

You can either write your own JCL to forward logs, or use the Transaction Analysis Workbench ISPF dialog (option 5 **Analytics**) to create JCL tailored for destinations such as Elastic, Splunk, DB2, or Hadoop.

Simple, self-contained JCL

You can extract logs in CSV or JSON format in less than a dozen lines of self-contained JCL. All you need to know is:

- Where Transaction Analysis Workbench is installed on your z/OS system.
- The location of a log that contains the records you want to extract.
- The record types that you want to extract.

You can specify the **CSV** or **JSON** command in an in-stream SYSIN data set in the JCL. Streaming to a TCP socket instead of writing to a file simply involves adding a **STREAM** command to the SYSIN data set.

Forward the fields you want, from the records you want, when you want

Transaction Analysis Workbench puts you in control of what you forward and when you forward it.

Transaction Analysis Workbench does not limit you to forwarding a fixed subset of fields from each record type. You can select as many or as few fields as you want.

You can filter records for forwarding based on combinations of field values. For example, you might want to only forward records from particular subsystem IDs and user IDs; perhaps also limited to records with abnormal metrics, such as high CPU or long response times.

Transaction Analysis Workbench log forwarding is not real-time. To forward logs with Transaction Analysis Workbench, you run batch jobs on z/OS. You decide when to run those jobs, and how often.

Platform-specific metadata or configuration files

In addition to converting logs to CSV or JSON, the report and extract utility can also create the following files for getting data onto analytics platforms:

Logstash configuration file for Elasticsearch
Forwards JSON or CSV data to Elasticsearch.

Transaction Analysis Workbench creates Logstash configs for output to Elasticsearch. You can edit the configs to specify different outputs supported by Logstash.

HCatalog table schema for Hadoop
Contains a Hive DDL **CREATE TABLE** statement that creates a catalog table; specifically, an external table that refers to an HDFS directory that contains one or more corresponding CSV files. You can open the catalog table in various Hadoop-based applications, such as IBM BigSheets.

Many applications can use CSV data directly. However, HCatalog table schemas offer several benefits: for example, they explicitly specify the data type of each column. Without a schema, applications must infer data types from the CSV data.

DB2 table schema
Contains a DB2 DDL **CREATE TABLE** statement.

DB2 load utility control statement
A **LOAD** control statement for the DB2 load utility that loads CSV data into DB2.

Pulling logs

Instead of forwarding, or "pushing", logs to a remote system, you can use various techniques to "pull" them from z/OS. For example, if you write CSV or JSON files to a z/OS UNIX directory, you can use NFS or SFTP to pull the files to a remote system.

Related tasks

Extracting logs to CSV or JSON
Transaction Analysis Workbench can extract logs from their original z/OS-based formats to comma-separated values (CSV) or JavaScript Object Notation (JSON) format for ingestion by analytics platforms and other applications.

Forwarding logs to analytics platforms
You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

STREAM command
Forwards log data in JSON Lines or CSV format to a TCP socket.

Filter conditions
A filter condition is an expression that selects log records based on a field value. Conditions refine filters. Without conditions, filters select log records based only on log types and codes.

Reporting

Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

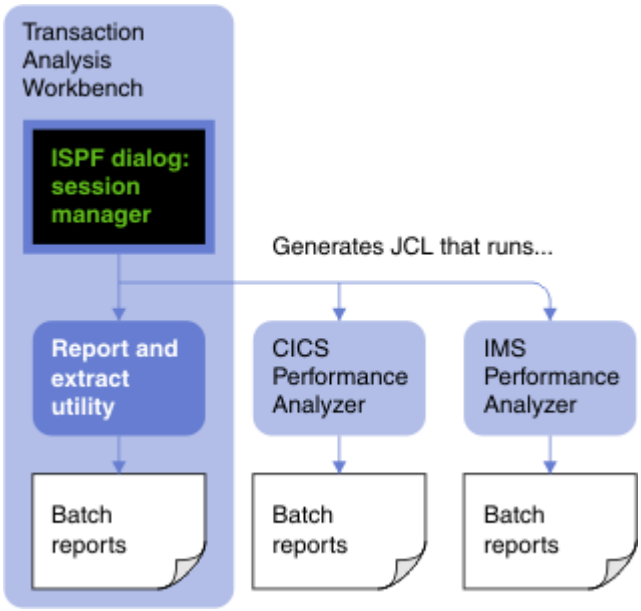


Figure 5. Transaction Analysis Workbench combines its own reporting with other products

For example:

Using its own report and extract utility

- z/OS system-level analysis, including: OPERLOG; address space activity; system resource utilization for CPU processors; virtual storage and page data sets; MVS system logger and DASD data set performance
- DB2 thread accounting, including SQL call elapsed time breakdown
- CICS-DBCTL transaction analysis
- IBM MQ thread accounting, including GET and PUT call counts and CPU usage
- APPC/MVS transaction performance analysis

Using CICS Performance Analyzer for z/OS

- CICS transaction performance analysis

Using IMS Performance Analyzer for z/OS

- IMS transaction performance and system analysis

Related concepts

Report and extract utility
The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Related tasks

Creating reports
You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Related reference

CICS-DBCTL reports
CICS-DBCTL reports process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.

DB2 accounting exception reports and extracts
DB2 accounting exception reports and extracts are generated from DB2 accounting (SMF type 101) records. DB2 accounting records from the specified input files are checked against optional *filtering criteria* for the DB2 subsystem ID (SSID), plan, and connection type. Records that match the filtering criteria are checked against *exception criteria* that specify thresholds for various performance-related values, such as CPU time.

SMF reports
Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

Sessions, workflows, and templates

Transaction Analysis Workbench offers an optional framework for analyzing problems in *sessions*. A session encapsulates information about a problem and the analysis of that problem. This information can include the time period in which a problem occurred, the systems involved, and the corresponding log files.

A session can also describe a sequence of tasks for analyzing a problem. This sequence of tasks is known as a *workflow*.

When creating a session for a problem, you can optionally select a session *template*. A session template is a blueprint for a session.

Sessions

You can use Transaction Analysis Workbench without using sessions, and analyze problems using ad hoc sets of log files instead. However, using sessions offers the following advantages that make it easier to manage problems and collaborate with other users:

- Sessions are stored in a session repository that you can share with other users. Each session has its own list of log files. If you want to collaborate with other users when analyzing a problem, then, rather than sending them the data set names of the corresponding log files and other information about the problem, you can simply refer them to a session.
- You can filter the list of sessions to display only the ones that you are interested in, such as open problems assigned to you.
- Rather than explicitly specifying log files yourself, you can specify a time period for the problem and the names of the systems involved, and then submit an automated file selection job to populate the list of files for the session.

  If you create a session from a session template, then the session template might already specify the systems involved; the template might also contain workflow tasks that perform the automated file selection. In that case, all you need to do is specify a time period and then run the tasks.
- Sessions offer a structured approach to managing the lifecycle of a problem:
  1. Create a session to register the problem; optionally, selecting a session template as a starting point
  2. Select the log files required to analyze the problem; optionally, by running workflow tasks inherited from the template
  3. Create batch reports that are specific to the problem; optionally, by running workflow tasks inherited from the template
  4. Browse log files individually, or merged in any combination
  5. While browsing log files, tag any log records that are of special interest: you, or other users, can later resume browsing at that position
  6. Write notes about your analysis
  7. Reassign the problem to the appropriate subject-matter expert

A session contains the following information about a problem:

- A unique identifier, known as a session *key*
- Optional metadata to help you track, assign, and share the analysis:
  - A one-line summary
  - A multi-line description
  - When the problem occurred
  - The systems involved
  - The problem severity (a number)
  - The problem status, such as open or closed
  - Who reported the problem
  - Who the problem is currently assigned to
  - A reference to an entry in your enterprise's problem tracking system
- A list of associated log files
- A history consisting of jobs that have been run for the session, notes that you or other users have written about the session, and tags that bookmark a position in the log browser.

Workflows

A session workflow consists of a sequence of two types of task: batch job tasks and note tasks. Batch job tasks contain JCL; note tasks contain instructions or tips for the user.

A typical workflow consists of the following sequence of tasks:

Log file selection
Batch job tasks to automate log file selection for the systems involved and for the time when the problem occurred.

Data reduction
Batch job tasks to create extracts of the selected log files, limited to records for the time when the problem occurred and the record types that are relevant to problem analysis.

Reporting
Batch job tasks to create reports to begin to identify the cause of the problem; or at least, to identify the particular system that is likely to be the cause of the problem.

Instruction
A note task describing what to do next, such as analyzing reports created by earlier tasks, and then assigning the problem to the appropriate subject-matter expert, such as a CICS or DB2 administrator.

A session workflow has the following two purposes:

- To *offer guidance* for analyzing a problem. Subject-matter experts can develop a workflow and save it in a session template.
- To *record* the steps that have been performed to analyze a problem. When you select an option in a session that involves running a batch job, such as automated file selection, extraction, or reporting, Transaction Analysis Workbench saves the JCL as a new batch job task. (Running an existing batch job task does not create a new task.) You can also add note tasks and your own custom batch job tasks.

You can run batch job tasks individually or you can schedule multiple tasks to run in sequence.

Each task has a status. When a batch job task completes, Transaction Analysis Workbench updates the task status to reflect the job completion code. You can also change the status of tasks manually.

You can use Transaction Analysis Workbench to automatically create batch job tasks that select log files for systems, and then create extracts and transaction indexes from those log files, according to the time period of the problem.

Batch job tasks include a step that saves the task sysout data sets as members of a library data set, enabling you to view the output even when the original job is purged. You can view the output in the Transaction Analysis Workbench ISPF dialog or plug-in.

Session templates

When you create a session for a problem, you can optionally select a template. The session inherits details from the template. A session template offers a starting point and guidance for analyzing a particular type of problem, such as a problem with a particular application.

Typically, a session template contains at least the following details:

- The systems that are involved in an application
- Workflow tasks to select the log files for those systems, and then create extracts and transaction indexes from those log files, according to the time period of the problem

To create a template, you can either start with a new, blank template or save an existing session as a template.

Subject-matter experts create session templates to help other users, such as help desk staff or other first responders, begin to analyze a problem themselves; or at least, to select the log files necessary for analysis, rather than immediately assigning the problem to an expert.

Related tasks

Analyzing problems using sessions
Using sessions offers a structured, team-oriented approach to analyzing problems. Sessions encapsulate information about a problem and its analysis that you can share with other users.

Creating session templates

Subject-matter experts create session templates to offer other users blueprints for analyzing particular types of transaction rather than starting with a blank session. If users create a session from a template, then the new session inherits details, systems, files, and tasks from the template.

Automated file selection

Automated file selection eliminates the tedious process of manually locating the data required for analysis. You specify a time period and the systems that you are interested in; Transaction Analysis Workbench locates the corresponding log files.

Transaction Analysis Workbench supports automated file selection for the following types of log file:

- DB2 log
- Tivoli OMEGAMON XE for DB2 Performance Expert near-term history
- IMS log
- IMS Connect Extensions journal
- SMF files and log streams

Related concepts

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Extracts

You can use Transaction Analysis Workbench to create extracts of original log files, containing the subset of records that you need to analyze a problem, within the relevant time period.

You can use extracts in Transaction Analysis Workbench as substitutes for the original log files.

Using extracts solves the following issues with original log files:

Data reduction
Original log files are often very large and contain more data for a longer time period than required. Extracts are smaller and more convenient to work with.

Data retention
Original log files can expire and be deleted after a period of time. The original log files might be deleted before you have completed analyzing a problem. Extracts offer a way to retain the data that you need for analysis when the original log files have been deleted.

Data access
Original log files can be on tape, and therefore unavailable for interactive investigation in the ISPF dialog, or they can be physically located on another system, with DASD that is not accessible to your ISPF dialog and batch reporting system.

Transaction Analysis Workbench can create extracts that contain log records from a single type of log file or a mixture of log records from different types of log file. You can use extracts created from a single type of log file as input to any tool that can read the original log files, including Transaction Analysis Workbench. However, *mixed* extracts, containing a mixture of log records from different types of log file, can only be read by Transaction Analysis Workbench.

You can reduce the need for extracts by using the time slicing feature of the Transaction Analysis Workbench log browser when browsing large log files. Rather than locating log records for the start of the desired time period by reading records in sequence from the start of a log file, time slicing samples records at intervals, locating the correct file position more quickly.

You can use the Transaction Analysis Workbench report and extract utility to extract and transform data from log records into JavaScript Object Notation (JSON) and comma-separated values (CSV) for use with other applications.

Related concepts

Report and extract utility

The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Related tasks

Creating extracts of log files in a session
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Related reference

Log stream types
Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Each record in a transaction index contains summarized information about the performance of a transaction and the resources that it consumed. You can use criteria that refer to field values in transaction index records to quickly identify problem transactions.

Using transaction indexes to identify problem transactions is a useful preliminary step for more detailed analysis. For example:

1. Use the Transaction Analysis Workbench ISPF dialog log browser to display a combined view of a transaction index with related log files.
2. Use filtering criteria to identify problematic transaction index records based on field values for response or CPU time.
3. Use tracking to see the records for that transaction from the related log files.

   Depending on the type of transaction and the available log files, the resulting set of tracked records can provide a view of the individual events in the life cycle of the transaction and offer more detailed information about the cause of the problem. For example, while the response time reported in a transaction index record identifies a problem with the transaction, elapsed times between the tracked records can help identify exactly where in the transaction life cycle the problem occurred.

The following table lists the types of transaction index. The transaction indexes for DB2 and IBM MQ use the specific term *accounting index* to more closely reflect the native terminology of those systems. The general term *transaction index* encompasses all of these files.

Table 1. Transaction indexes

Type of transaction index	Original source records	Transaction index record log type and code (*type*:*code*)	Description
CICS transaction index	CICS monitoring facility (CMF) performance class records, SMF type 110	CMF:6E13	Each original CMF record can contain information for multiple CICS transactions. Transaction Analysis Workbench divides the original CMF records into one record per transaction. A CICS transaction index record is a valid CMF record, suitable for use with tools that read CMF records, such as CICS Performance Analyzer.

Table 1. Transaction indexes (continued)

Type of transaction index	Original source records	Transaction index record log type and code (*type*:*code*)	Description
DB2 accounting index	DB2 accounting trace records, SMF type 101	DTR:003	Original source record format, unchanged.
IMS transaction index	IMS log records	IMS:CA01	An IMS transaction index consolidates information from multiple IMS log records into one record per transaction. You can create IMS transaction indexes using either IMS Performance Analyzer or Transaction Analysis Workbench. IMS Performance Analyzer creates transaction index records in the order that transactions complete their transit processing in the IMS log file. The index is not sorted in transaction start time sequence. Sorting the index is optional, particularly if processing it standalone. However, you might find it best to work with a sorted index so that the records are reported in the more meaningful transaction arrival time sequence, particularly if merging with other log files. You can use the Transaction Analysis Workbench ISPF dialog to generate JCL that creates an IMS transaction index and then performs an additional step that sorts the index in transaction arrival time sequence. For more accurate IMS transaction index records, use the IMS parameter TRANSTAT=YES to collect type X'56FA' transaction-level accounting IMS log records.
IBM MQ accounting index	IBM MQ accounting records, SMF type 116	SMF:74	Original source record format, unchanged.

Example IMS transaction index record

IMS transaction index records are useful for identifying problematic IMS transactions because they contain fields that offer cumulative information about an IMS transaction that are not available in any single IMS log record. For example, the Process field in an IMS transaction index record contains the total processing time for an IMS transaction.

```
+0004  Code... CA01  IMS Transaction
+0578  STCK... CC9B05551263D059      LSN.... 000000000000002B
       Date... 2014-01-23 Thursday   Time... 17.41.43.336509.021

+0004  Type....... CA          Subtype.... 01

+00A4  ID......... Transaction Identification section
+00A4  TranCode... ©WITHDRAW©  Program.... ©BANKING ©
+00B4  Userid..... ©JOHN    ©  LTerm...... ©CENTRAL ©
+0108  ISO........ IMS transaction start time (local)
+0108  Date....... ©2014-01-23©
+0113  Time....... ©03.41.43.336510©

+0188  Transit.... Transaction Transit accounting section
+0188  InputQ..... 0.029111    Process.... 0.033082    OutputQ.... 0
+01B0  TotalTm.... 0.062083    SwitTime... 0.029001    CPUtime.... 0.000889
       SYNCELAP... 0.000065    SYNCPH1E... 0.000015    SYNCPH2E... 0.000050

+0320  Calls...... DB call summary section
+0324  FFGets..... +46         FFUpdats... +9          FPGets..... +47
+0340  FPUpdats... +9

+0500  DBSection.......... Database section
+0504  DBName..... ©CUSTDB  ©  DBBlksUpd.......... +2

+0528  DBSection.......... Database section
+052C  DBName..... ©ATMDB   ©  DBBlksUpd.......... +14

+0550  DBSection.......... Database section
+0554  DBName..... ©CLIENTDB©  DBBlksUpd.......... +2
```

Figure 6. Panel: Excerpt of an IMS transaction index record displayed in the Transaction Analysis Workbench log browser

Related concepts

Transaction tracking
Transaction tracking identifies the significant events in the lifecycle of a single transaction across subsystems and logs.

Event time stamps versus log record time stamps
Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Related tasks

Creating transaction indexes
Creating transaction indexes from original log files is a useful preliminary step for problem analysis. Where original log files typically contain many types of record, a transaction index contains only a single type of record. Analyzing transaction index records is a useful starting point for analyzing problems with transactions.

Related reference

System and file types
System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

IMSINDEX command

Consolidates IMS log records into IMS transaction index records (log type IMS, log code CA01).

Exception analysis

Transaction Analysis Workbench exception analysis identifies log records that meet your exception criteria, such as long response time or abnormal conditions, and creates a transaction index of those log records for further analysis.

The resulting transaction index is much smaller than the original instrumentation data, and contains only the exceptions that we are interested in analyzing.

To further refine the results, you can also specify filtering criteria for any performance-related field.

Specialized exception analysis for specific systems

Transaction Analysis Workbench provides specialized exception analysis for the following systems:

CICS-DBCTL
: Transaction Analysis Workbench CICS-DBCTL reports combine exception analysis for CICS (CMF) performance records and IMS-DBCTL log records into a consolidated view of CICS-DBCTL transaction performance that spans CICS and IMS.

DB2
: Transaction Analysis Workbench DB2 accounting exception reports support a wide variety of exception criteria for performance-related values.

Related concepts

CICS-DBCTL transactions
You can analyze response time problems in CICS-DBCTL transactions using CICS monitoring facility (CMF) performance class records (SMF type 110 records), IMS log records, or a combined view of both.

Related reference

DB2 accounting exception criteria

Exception criteria specify thresholds for various performance-related accounting values, such as CPU time. Only records that trigger an exception (match the exception criteria) are included in the DB2 accounting exception extracts and reports.

Transaction Analysis Workbench components

Transaction Analysis Workbench consists of the following components: an ISPF dialog user interface; a report and extract batch utility; an automated file selection batch utility; knowledge modules to interpret and format log records; repositories for various types of data; and an Eclipse plug-in.

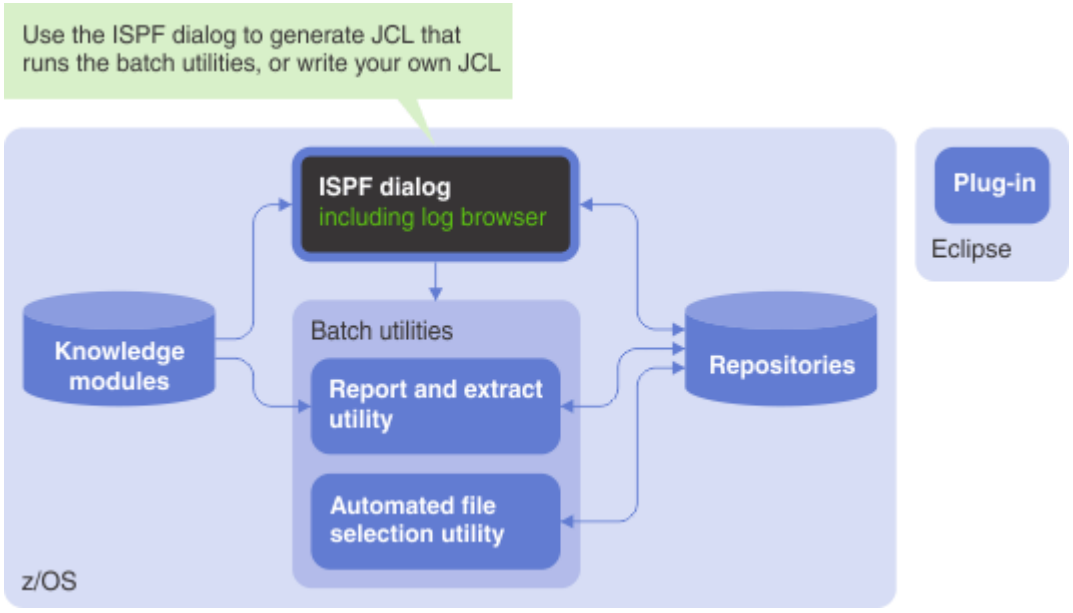


Figure 7. Transaction Analysis Workbench components

ISPF dialog

The Transaction Analysis Workbench ISPF dialog user interface presents a hierarchy of options for interactively browsing logs, managing sessions, generating JCL to run batch jobs, and defining various repository records related to analysis, such as system definitions and filters.

The following two figures show a map of the ISPF dialog and its options.

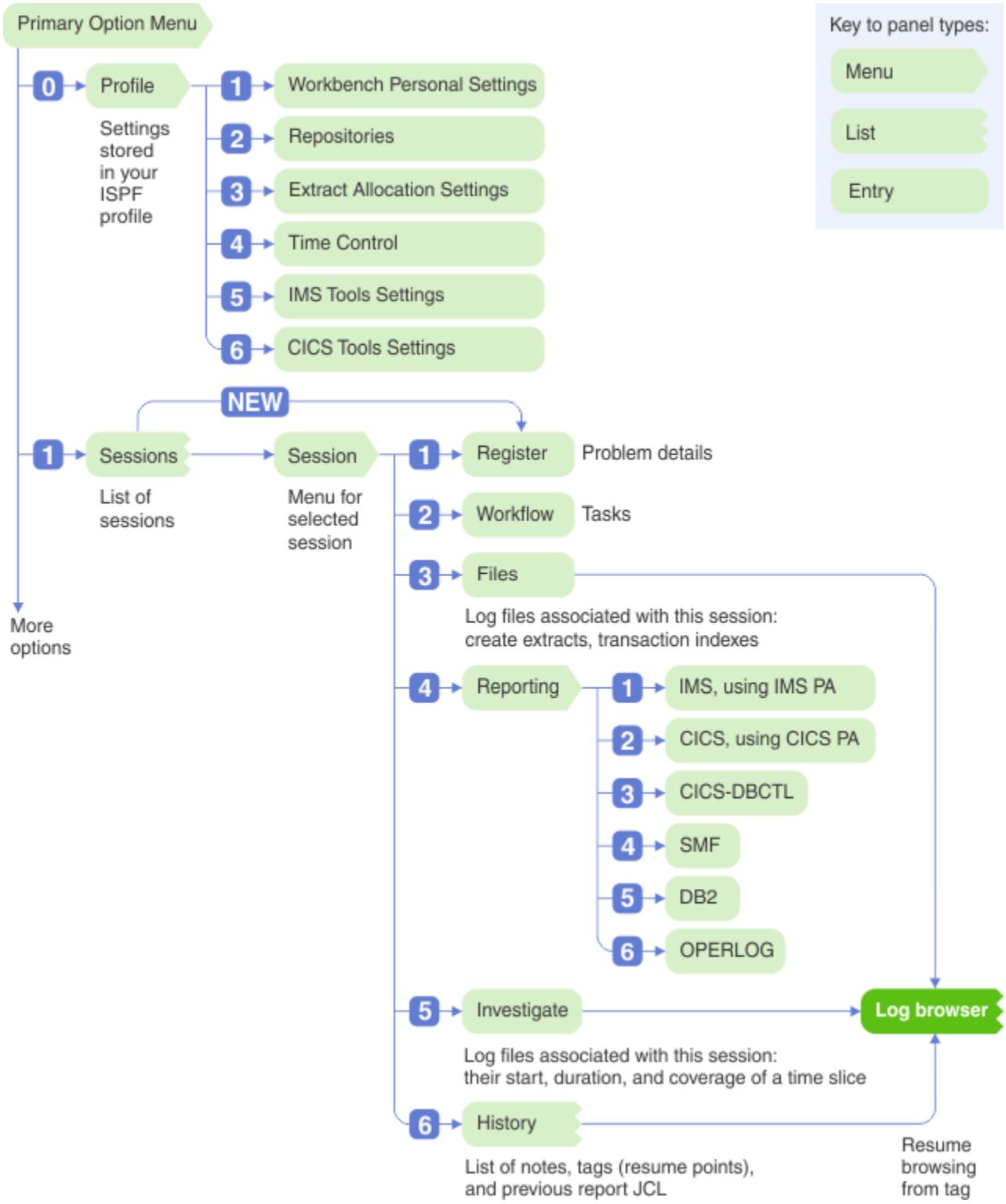


Figure 8. Map of the Transaction Analysis Workbench ISPF dialog (1 of 2)

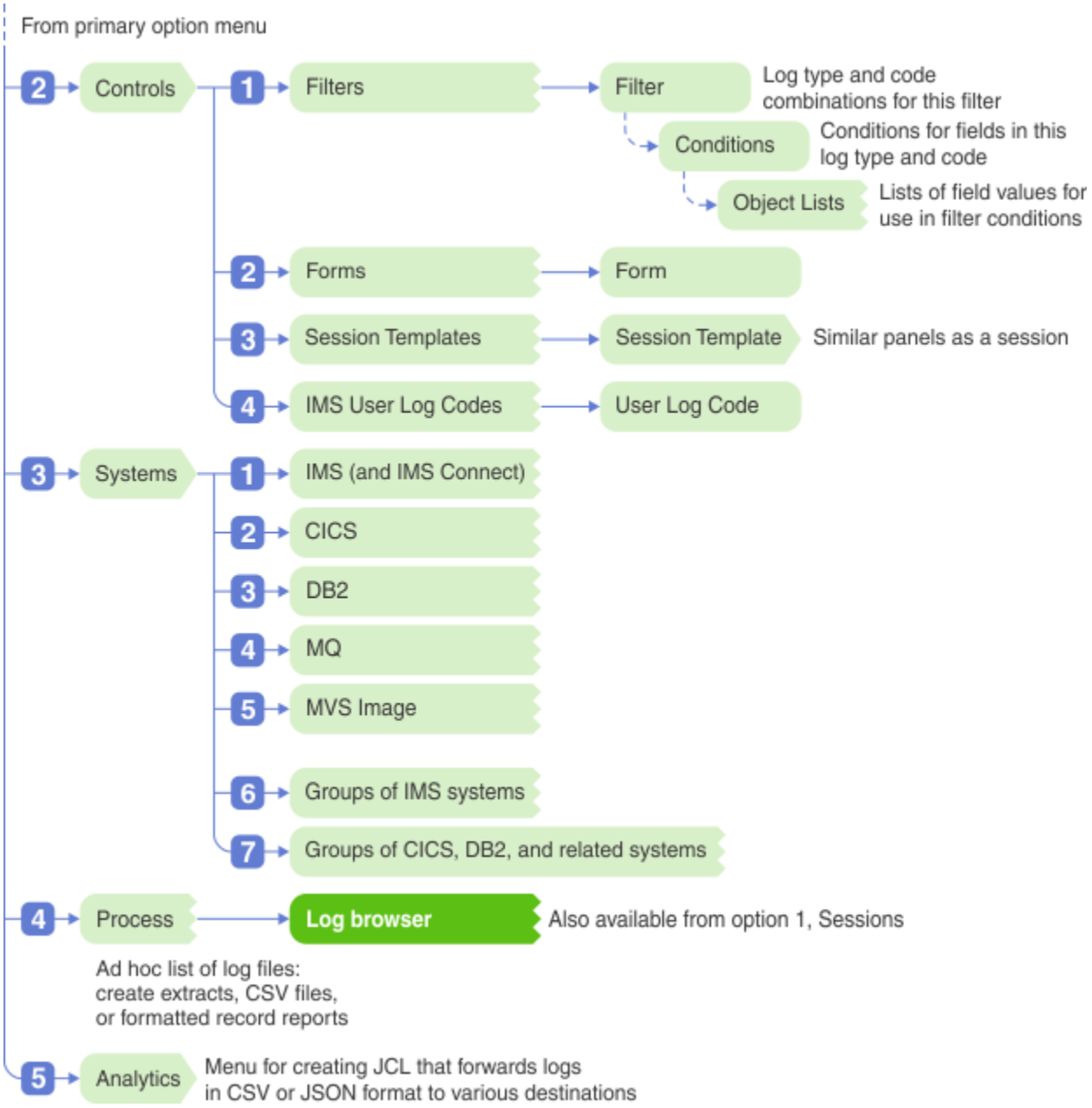


Figure 9. Map of the Transaction Analysis Workbench ISPF dialog (2 of 2)

Log browser

The log browser does not have its own primary menu option. You can start the log browser from the following options:

Option 1 Sessions
: This is the recommended option for typical use; sessions offer a structured, team-oriented framework for problem analysis that encapsulates the details of each problem separately.

Option 4 Process
: This option allows you to browse logs that you have added to your personal ad hoc list. This option offers a more direct way to browse logs than using sessions: add a log to your list, and then select it for browsing. However, this option lacks the structure and other features offered by sessions; it's just an ad hoc list of logs.

Option 0 Profile

Each Transaction Analysis Workbench user has a *profile* that consists of personal settings that affect the behavior of the ISPF dialog for that user. Each user can have different profile settings.

Option 1 Sessions

A *session* is a collection of information about a particular problem, such as the time it occurred, the systems involved, and a list of associated log files. Transaction Analysis Workbench stores sessions in a repository that can be shared between users. Sessions offer a structured, team-oriented framework for problem analysis, enabling you to manage each problem separately, while sharing problem analysis between users.

Option 2 Controls

The following controls allow you to customize how Transaction Analysis Workbench presents log data. Transaction Analysis Workbench stores these controls in a repository that can be shared between users and with other products.

Filters
Select only the log records that are relevant to you.

Object lists
Enable you to define a set of related field values once, and then refer to that set of values in many filter conditions.

Forms
Select only the fields, from a particular log record type, that are relevant to you.

IMS user log records
Format IMS log records written by IMS applications (log code X'A0' or greater).

Session templates are another type of control. Session templates are stored in the session repository.

Option 3 Systems

A *system definition* is a collection of information that associates the name of a system (such as an MVS image, an IMS region, a CICS region, or a DB2 system) with details about that system, including the location of the log files to which it writes log records. You can use the Transaction Analysis Workbench ISPF dialog to create and edit system definitions.

When you register a session for a problem, you can specify various problem details, including the names of the systems involved. If you have defined these systems to Transaction Analysis Workbench, then you can use the automated file selection utility to locate the corresponding log files, based on the system names and a time period, rather than having to manually locate and specify the log files. If you do not use automated file selection, system definitions are for reference only, to associate a system name with its log files.

Transaction Analysis Workbench can share system definitions with other products. The repository to which Transaction Analysis Workbench saves each system definition depends on the system type.

Option 4 Process

Dialog option 4 **Process** offers an alternative to option 1 **Sessions** for browsing and processing log files. Under option 1 **Sessions**, you register a session, add associated log files to that session, and then work with those log files. Option 4 **Process** enables you to work with a personal ad hoc list of log files, independent of sessions.

Using sessions is typically a better choice. Among other benefits, using sessions allows you to manage problems individually and share problem analysis with other users.

Option 5 Analytics

Dialog option 5 **Analytics** creates JCL that forwards logs in CSV or JSON format to various destinations.

Related tasks

Analyzing problems using sessions
Using sessions offers a structured, team-oriented approach to analyzing problems. Sessions encapsulate information about a problem and its analysis that you can share with other users.

Browsing logs interactively
You can use the Transaction Analysis Workbench ISPF dialog to browse records from different logs merged in a single view. You can select a record to browse its fields, and then select a field to browse a detailed description of its contents.

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Defining filters
To select only the log records that are of interest to you, define and then use a filter. A filter specifies which log record codes to select or exclude and, optionally, more detailed conditions based on field values.

Defining forms
To hide log record fields that are not of interest to you, define and then use a form.

Analyzing IMS user log records
IMS applications can use the DL/I **LOG** call to write records to the IMS system log. Such records are known as IMS user log records; they have a log code of X'A0' or greater, which is outside the range of log codes used by IMS itself. To enable Transaction Analysis Workbench to process and format IMS user log records, you create knowledge modules for the corresponding log codes.

Forwarding logs to analytics platforms
You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

Starting the ISPF dialog
To start the Transaction Analysis Workbench ISPF dialog, you run the FUWOREXX REXX exec supplied in the SFUWEXEC library.

Related reference

Required Transaction Analysis Workbench ISPF dialog profile settings
Transaction Analysis Workbench ISPF dialog option 0 **Profile** contains settings that are specific to each user. Some of these settings are required, in the sense that they must have appropriate values for Transaction Analysis Workbench to perform typical functions correctly. Other settings are required only for more specific functions, or depend on your personal preferences.

Repositories

Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

The following figure illustrates the repositories that Transaction Analysis Workbench can use and the products that can share them.

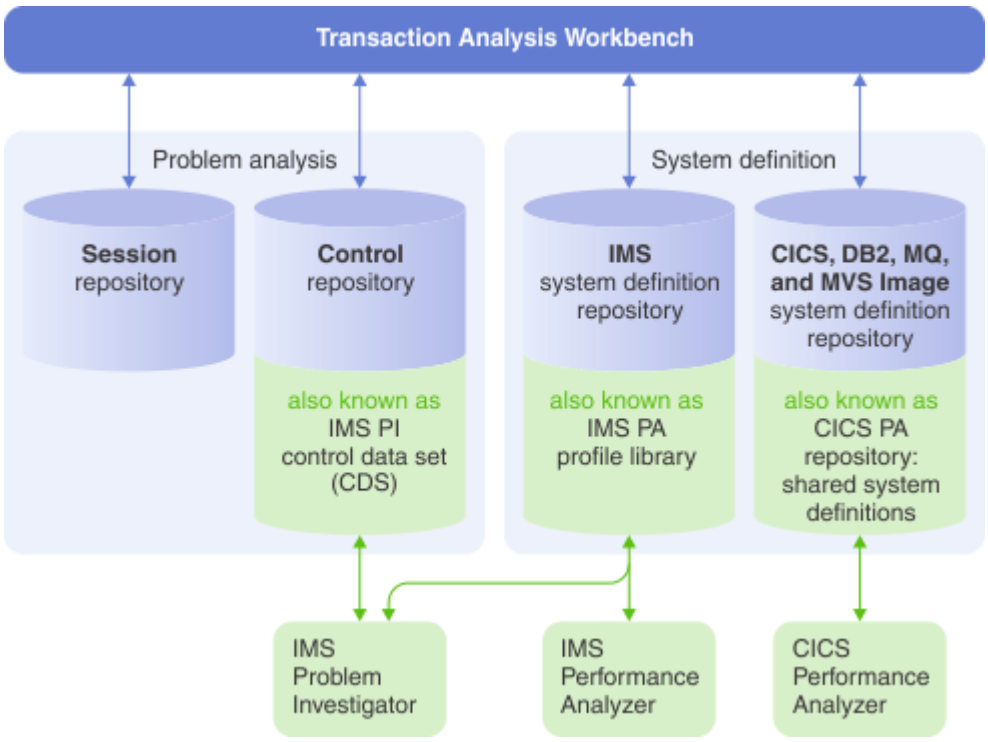


Figure 10. Transaction Analysis Workbench repositories

Transaction Analysis Workbench can create, write to, and read from all of these repositories.

All repositories are optional, depending on which features of Transaction Analysis Workbench you use.

The repositories are divided into two categories: problem analysis and system definition.

Problem analysis repositories

Problem analysis repositories contain information that Transaction Analysis Workbench uses to help you analyze problems. There are two problem analysis repositories:

Session repository
Contains information about each problem session. Also contains session templates.

This repository is required if you use sessions (ISPF dialog option 1 **Sessions**). Using sessions is the recommended method for analyzing transactions. If you always use your personal ad hoc list of log files (ISPF dialog option 4 **Process**) instead of sessions, then you do not need this repository.

This is the only repository that is exclusive to Transaction Analysis Workbench.

Control repository
Contains controls: filters, forms, and object lists. This repository can be shared with IMS Problem Investigator, where it is known as the *control data set (CDS)*.

System definition repositories

System definition repositories contain system definitions that Transaction Analysis Workbench uses for automated log file selection. For example, an IMS system definition specifies the names of the RECON data sets for that IMS system.

You need these repositories if you want to use the Transaction Analysis Workbench ISPF dialog to generate JCL for the automated file selection utility.

There are two system definition repositories:

IMS system definition repository
Can be shared with IMS Performance Analyzer, where it is known as the *permanent ISPF table library*, and IMS Problem Investigator, where it is known as the *IMS PA profile library*.

You need this repository to automate selection of IMS logs and IMS Connect Extensions journals.

CICS, DB2, IBM MQ, and MVS Image system definition repository
Can be shared with CICS Performance Analyzer, where it is known as the CICS Performance Analyzer repository.

Transaction Analysis Workbench only uses CICS Performance Analyzer *shared* system definitions, which are saved in HDB registers, not *personal* system definitions, which are saved in each CICS Performance Analyzer user's personal profile library. You can use CICS Performance Analyzer to take-up (copy) personal system definitions into shared system definitions.

You need this repository to automate selection of DB2 logs and SMF files.

Sharing repositories between products and users

Sharing repositories between products is optional. When you use the Transaction Analysis Workbench ISPF dialog, you can either refer to existing repositories created by other products, or you can specify new data set names. The first time that you attempt to access a new repository, the ISPF dialog displays a panel that enables you to allocate the data set. Alternatively, to allocate these data sets without using the ISPF dialog, use the JCL member FUWALOCR in the sample library SFUWSAMP.

Sharing repositories between users is also optional. Each Transaction Analysis Workbench ISPF dialog user can specify a different set of repositories.

Sharing repositories has advantages. For example:

- Sharing a session repository between users enables collaboration. Users can take advantage of the fact that another user has already registered a session for a problem, selected the corresponding log files, and written notes about their analysis.
- Sharing system definition repositories between users and products avoids duplication of data entry. For example, if an IMS Performance Analyzer user has already defined an IMS system, including details of its related RECON data sets, then a Transaction Analysis Workbench user can reuse that definition to perform automated file selection of the IMS logs.

Your organization must decide how and whether to share repositories. Each user of the Transaction Analysis Workbench ISPF dialog must specify appropriate repository data set names under option 0.2 **Repositories**.

The first time each new user starts the Transaction Analysis Workbench ISPF dialog, the dialog searches the user's ISPF profile library for members created by the dialogs of other products that can share repositories. The Transaction Analysis Workbench ISPF dialog uses the information in these members to fill in the repository data set names under option 0.2 **Repositories**.

Related concepts

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Planning to share repositories between users and products
Before rolling out Transaction Analysis Workbench to users in your organization, you should decide how or whether you want to share its repositories between users and other products.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Tutorial: Automating selection of IMS logs

This tutorial shows you how to define an IMS system to Transaction Analysis Workbench, and then use that system definition to locate the related IMS log files (SLDS or OLDS) for a particular time interval.

Tutorial: Automating selection of DB2 logs
This tutorial shows you how to define a DB2 system to Transaction Analysis Workbench, and then use that system definition to locate the related DB2 log files for a particular time interval.

Related reference

Administrative commands
Administrative commands copy filters, forms, or object lists (collectively known as *controls*) between Transaction Analysis Workbench control repositories.

Report and extract utility

The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

The utility can create the following types of report:

Formatted record reports
Reproduce the formatting that the ISPF dialog displays when you browse an individual log record; or when you browse a list panel of log records and you press the Right function key (F11) to expand the display so that each record occupies up to a few lines (known as "brief" format).

SMF reports
Display information that is specific to a particular SMF record type and, if applicable, subtype. The report layout is also specific to the record type and subtype. SMF reports are available only for some SMF record types and subtypes.

OPERLOG reports
Display formatted listings of z/OS operations log (log stream) records.

CICS-DBCTL reports
Process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.edThe utility can convert logs to JSON or CSV format, and create related metadata files for forwarding logs to various analytics platforms.

The utility can create CSV and related files specifically for the Mobile Workload Reporting Tool (MWRT).

Related concepts

Reporting
Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

Extracts
You can use Transaction Analysis Workbench to create extracts of original log files, containing the subset of records that you need to analyze a problem, within the relevant time period.

Mobile Workload Pricing
Mobile Workload Pricing (MWP) for z/OS is a pricing model that reduces the cost of running mobile workloads on z/OS. To take advantage of MWP, you use the Mobile Workload Reporting Tool (MWRT) to submit monthly reports to IBM. MWRT requires two types of input files: CSV and SMF. You can use Transaction Analysis Workbench to create these files.

Related tasks

Creating reports
You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Forwarding logs to analytics platforms

You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

Creating extracts of log files in a session
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Related reference

Report and extract utility
The report and extract utility processes logs to create reports and extracts; convert logs to comma-separated values (CSV) or or JavaScript Object Notation (JSON); runs REXX execs that process log files; and exports or imports controls (filters, forms, and object lists) between control repositories.

SMF reports
Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

Automated file selection utility

The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

When you register a session in the Transaction Analysis Workbench ISPF dialog, you can choose to either manually specify the data set names of the associated log files or, for the following types of log file, run the automated file selection utility:

DB2 log
To select DB2 log files, the utility uses output from the DB2-supplied print log map utility (DSNJU004). This output lists the locations of DB2 log files, based on information in the DB2 bootstrap data set (BSDS).

IMS log
To select IMS log files, the utility uses the IMS database recovery control (DBRC) API to get the locations of IMS system log data sets (SLDS) or online data sets (OLDS) from the RECON data sets.

IMS Connect Extensions journal
To select IMS Connect Extensions journals, the utility reads the IMS Connect Extensions definition repository.

SMF file or log stream
To select SMF files or log streams, the utility reads the system definition repository. You can automate SMF file selection for the following types of system definition: CICS, DB2, MVS image, and IBM MQ.

Tivoli OMEGAMON XE for DB2 Performance Expert near-term history
To select DB2 near-term history (NTH) files, the utility reads the SEQDATASET dsname pattern in the DB2 system definition in the system definition repository, and then reads the first record of each matching NTH file to identify the files for the specified time period.

The utility saves the details of the located log files to the session, which is stored in the session repository.

You can use the Transaction Analysis Workbench ISPF dialog to generate JCL to run the utility.

Related concepts

Automated file selection
Automated file selection eliminates the tedious process of manually locating the data required for analysis. You specify a time period and the systems that you are interested in; Transaction Analysis Workbench locates the corresponding log files.

Repositories

Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Related tasks

Adding log files to a session
If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Related reference

Automated file selection utility
The JCL statements to invoke the automated file selection utility, FUWFILES, depend on the type of log file that you want to select.

System and file types
System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

Hardware and software prerequisites
Before you install and configure Transaction Analysis Workbench, make sure that your environment meets the following minimum hardware and software requirements.

Knowledge modules

A *knowledge module* is an executable load module that Transaction Analysis Workbench uses to interpret and process a particular type of log record.

Transaction Analysis Workbench is supplied with many knowledge modules. Each knowledge module contains the following information, sometimes referred to as metadata, about a particular type of log record:

- Field structure, data types, and names. Field names in knowledge modules typically match the names in the IBM-supplied z/OS® assembler data sections (DSECTs) that map each record type.
- Field descriptions. Transaction Analysis Workbench displays these descriptions when you zoom on a field while browsing logs in the ISPF dialog.
- Which fields in the record type map to Transaction Analysis Workbench global fields.

You can create your own knowledge modules for IMS log records written by IMS applications, known as IMS user log records, enabling you to analyze these log records in exactly the same way as log records written by IMS itself.

Related tasks

Analyzing IMS user log records
IMS applications can use the DL/I **LOG** call to write records to the IMS system log. Such records are known as IMS user log records; they have a log code of X'A0' or greater, which is outside the range of log codes used by IMS itself. To enable Transaction Analysis Workbench to process and format IMS user log records, you create knowledge modules for the corresponding log codes.

Related reference

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

z/OS Explorer plug-in

You can use the Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) to view or create sessions, submit z/OS batch jobs for session workflow tasks, and view the output of those batch jobs.

The plug-in is aimed at "first responders", such as help desk staff, who record the details of problems reported by users of z/OS-based transactions.

Here is a typical scenario involving the plug-in:

1. A user of a z/OS-based application experiences a problem with a transaction, such as an abend or a long response time.
2. The user reports the problem to their help desk.
3. A help desk team member uses the Transaction Analysis Workbench plug-in to create a session for the problem, based on a session template.

   Session templates are created by subject-matter experts using the Transaction Analysis Workbench ISPF dialog.
4. The help desk team member uses the plug-in to submit the batch jobs defined by the workflow tasks and view the output of any reports.
5. Based on reports created by batch job tasks and instructions in any note tasks, the help desk team member uses the plug-in to update the session details, assigning the session to the appropriate subject-matter expert.

   Precisely how the help desk team member informs the expert that the session is assigned to them depends on the practices at their organization.

   Session details include status and assignee to help you to track problem analysis, but the plug-in is not intended to be a replacement for a comprehensive problem tracking system. Organizations with a help desk typically already have a problem tracking system. Session details also include a reference field for referring to an entry in your organization's problem tracking system.
6. The subject-matter expert continues analyzing the problem.

   If the problem can be analyzed using the reports created by workflow tasks, then the expert might continue to use the plug-in to view those reports. However, if the analysis requires functions that are not available in the plug-in, such as interactively browsing logs, the expert will use the Transaction Analysis Workbench ISPF dialog.

Related tasks

Installing the z/OS Explorer plug-in
The Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) provides a graphical user interface (GUI) to some of the functions provided by the Transaction Analysis Workbench ISPF dialog. The plug-in communicates with Transaction Analysis Workbench via Common Services Library server.

Tutorial: Using the plug-in to create a session and perform a workflow
This tutorial shows you how to use the Transaction Analysis Workbench Eclipse plug-in to create a session based on a template, and then perform the workflow tasks inherited from the template. The tutorial also covers the prerequisite steps that need to be performed in the ISPF dialog by a subject-matter expert: defining systems, and then creating a session template that contains workflow tasks for those systems.

Transaction Analysis Workbench terminology

These terms have specific meanings when used in the context of Transaction Analysis Workbench.

Extract
A file that contains log records copied from one or more log files.

Log browser
Transaction Analysis Workbench ISPF dialog for browsing log files.

Log file
Sometimes abbreviated to "log". A general term for any data set or log stream containing performance, statistics, or other data related to transaction analysis that Transaction Analysis Workbench can read. This term includes files that are not typically characterized as "log" files, such as SMF data sets.

Log record
A record in a log file. This term includes records in SMF data sets.

Log type and code
The two identifiers that Transaction Analysis Workbench uses to classify log records.

Session
A set of information about a problem and the analysis of that problem that is stored in a session repository. Typically, session repositories are shared between users, enabling collaboration.

Time slice
A set of log records for a specified time period, from one or more log files.

Tracking
A function of Transaction Analysis Workbench that shows the log records that are related to a particular transaction, from all available log files, providing an end-to-end view of the transaction lifecycle spanning all subsystems involved. The set of log records for a particular transaction is known as a *tracking result set*.

Transaction index
A specialized type of extract that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Property
When used in the context of JSON, the terms *property* and *field* are sometimes used interchangeably.

Workflow
A sequence of tasks defined in a session template.

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Related reference

Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

System and file types

System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use a software product successfully.

The major accessibility features in this product enable users to perform the following activities:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.
- Customize display attributes such as color, contrast, and font size.
- Operate specific or equivalent features by using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:

- *z/OS ISPF User's Guide, Volume 1*
- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*

These guides describe how to use the ISPF interface, including the use of keyboard shortcuts or function keys (PF keys), include the default settings for the PF keys, and explain how to modify their functions.

Chapter 2. Installing Transaction Analysis Workbench

To install Transaction Analysis Workbench, follow the instructions in the *Transaction Analysis Workbench Program Directory*, GI10-8825. Then follow the instructions here to start the Transaction Analysis Workbench ISPF dialog and verify the installation.

These installation tasks are for use only by the person who installs Transaction Analysis Workbench; they are not intended for every user.

However, after installation is complete, every new user should configure their personal Transaction Analysis Workbench ISPF profile settings.

Hardware and software prerequisites

Before you install and configure Transaction Analysis Workbench, make sure that your environment meets the following minimum hardware and software requirements.

To install Transaction Analysis Workbench, you use SMP/E and standard RECEIVE, APPLY, and ACCEPT processing. For complete information about installation requirements, prerequisites, and procedures for Transaction Analysis Workbench, see the *Transaction Analysis Workbench Program Directory*.

Hardware prerequisites

Transaction Analysis Workbench operates on any hardware configuration that supports the required software.

Software prerequisites: mandatory

Before installing and configuring Transaction Analysis Workbench, ensure that the following software requirements are fulfilled:

- z/OS, V2.1, or later

Software prerequisites: conditional

Transaction Analysis Workbench can read the log records from many types of system. Except under some specific conditions relating to automated file selection, Transaction Analysis Workbench does not require the systems that wrote the records: just the log files that contain the records.

For example, suppose you work for the information systems support department of an enterprise that uses CICS, and you want to use Transaction Analysis Workbench, which is installed on your local z/OS system, to analyze the performance data from a CICS region that is running on a z/OS system in another country. To analyze the performance data, you do not require CICS to be installed on your local z/OS system: you just need copies of the dumped SMF data sets (containing the CMF records written by that region) for the time period that you want to analyze.

Under the following conditions, Transaction Analysis Workbench has additional software requirements:

Analyzing logs
You can use Transaction Analysis Workbench to analyze logs that have been created by the following product versions:

Table 2. Prerequisite product versions for supported log records

Log records	**Must have been created by one of the following product versions...**
IMS log	IMS, V12.1, or later

Table 2. Prerequisite product versions for supported log records (continued)	
Log records	**Must have been created by one of the following product versions...**
DB2 log, or SMF records written by DB2	DB2 for z/OS, V10.1, or later DB2 for z/OS Value Unit Edition, V10.1, or later
IBM MQ log extract	WebSphere MQ for z/OS, V7.1 IBM MQ for z/OS, V8.0, or later
IMS Connect Extensions journal (contains IMS Connect event data)	IMS Connect Extensions for z/OS, V2.4 IMS Performance Solution Pack for z/OS, V1.3
CICS monitoring facility (CMF) records (SMF type 110, subtype 1)	CICS Transaction Server for z/OS, V4.1, or later
Tivoli OMEGAMON XE for IMS Application Trace Facility (ATF) journal	Tivoli OMEGAMON XE for IMS on z/OS, V4.2, or later

Submitting IMS Performance Analyzer for z/OS or CICS Performance Analyzer for z/OS batch reports

The Transaction Analysis Workbench ISPF dialog allows you to submit batch jobs that request reports from the following products:

- IMS Performance Analyzer for z/OS, V4.4
- CICS Performance Analyzer for z/OS, V5.2, or later

If those products are not installed, the batch jobs will fail.

Using the automated file selection utility

When you use the Transaction Analysis Workbench automated file selection utility for the following types of log file, Transaction Analysis Workbench requires the systems that manage the archiving of those log files:

DB2 logs

Automated file selection of DB2 logs requires DB2. Specifically, the automated file selection utility uses output from the DB2-supplied print log map utility, DSNJU004. The print log map utility generates a list of DB2 log files and their time spans from information in the DB2 bootstrap data set (BSDS). When you use the Transaction Analysis Workbench ISPF dialog to run automated file selection, the dialog generates JCL that includes a step that calls DSNJU004; this requires DB2 to be installed on the system.

However, even in this situation, you can remove the requirement for DB2: perform the DSNJU004 job step on the system where DB2 is installed; save the results to a data set; and then adjust the JCL for the Transaction Analysis Workbench automated file selection utility to refer to that data set, rather than calling DSNJU004. Remember that, if you copy the DSNJU004-generated data set and the DB2 log files from their original z/OS system to another system where Transaction Analysis Workbench is installed, the Transaction Analysis Workbench automated file selection utility will expect the DB2 log files to have the same data set names that they had on the original system.

IMS logs

Automated file selection of IMS logs requires IMS. Specifically, the automated file selection utility uses the IMS DBRC API to access the IMS RECON data sets, containing information about IMS log (SLDS) files and their time spans.

IMS Connect Extensions journals

Automated file selection of IMS Connect Extensions journals requires the IMS Connect Extensions definition repository (a VSAM KSDS), containing information about journals and their time spans. This is not quite the same type of requirement as for DB2 or IMS logs, because it does not involve using any IMS Connect Extensions-supplied programs or programming interfaces. Just remember

that, if you want to use Transaction Analysis Workbench to analyze IMS Connect Extensions journals, and you want to use the Transaction Analysis Workbench automated file selection utility to select those journals, then, in additional to the journals, you also need the corresponding IMS Connect Extensions definition repository.

Related concepts

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Related tasks

Creating reports
You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Related reference

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Planning to share repositories between users and products

Before rolling out Transaction Analysis Workbench to users in your organization, you should decide how or whether you want to share its repositories between users and other products.

For example, you might decide that you want all users in your organization to share the same session repository. Or you might choose to have separate session repositories for problems in development and production environments.

Each Transaction Analysis Workbench ISPF dialog user can specify their own set of repositories, under option 0.2 **Repositories**. If a user specifies a new data set name for a repository, the dialog allocates the data set the first time the user attempts to access that repository.

You might prefer to allocate the data sets before rolling out Transaction Analysis Workbench to users, and then direct users to specify those existing data set names. To allocate the data sets without using the dialog, use the JCL member FUWALOCR in the sample library SFUWSAMP.

Related concepts

Repositories
Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Transaction Analysis Workbench libraries

The *Transaction Analysis Workbench Program Directory* describes how to install the Transaction Analysis Workbench libraries on your system.

Transaction Analysis Workbench is supplied in the following libraries. The library names in the following list are the default library low-level qualifiers. You can choose to install these libraries using different low-level qualifiers.

SFUWEXEC
REXX EXECs.

SFUWGENU
GUI files.

SFUWLINK
Executable load modules.

SFUWMENU
ISPF messages.

SFUWPENU
ISPF panels.

SFUWSAMP
Samples, such as JCL for creating reports and extracts.

SFUWSENU
ISPF skeleton JCL.

SFUWTENU
ISPF input tables.

To use Transaction Analysis Workbench, you need to know where these libraries have been installed on your system. You need to know the high-level qualifier for these libraries on your system and, if you did not use the defaults, their low-level qualifiers.

Starting the ISPF dialog

To start the Transaction Analysis Workbench ISPF dialog, you run the FUWOREXX REXX exec supplied in the SFUWEXEC library.

This is a simple, quick method for starting the Transaction Analysis Workbench ISPF dialog. Use this method to verify that Transaction Analysis Workbench has been installed correctly.

1. On the ISPF primary option menu, select option 6 **Command**.
2. Enter the following command:

   ```
   EX 'prefix.SFUWEXEC(FUWOREXX)' 'prefix'
   ```

   where *prefix* is the high-level qualifier of the Transaction Analysis Workbench libraries. For example:

   ```
   EX 'FUW.V1R3M0.SFUWEXEC(FUWOREXX)' 'FUW.V1R3M0'
   ```

   This command assumes that you have installed Transaction Analysis Workbench using the default library low-level qualifiers. If you used different low-level qualifiers, then you must specify the library that contains FUWOREXX instead of the default SFUWEXEC, and you must also specify the complete list of low-level qualifiers on the command line. For details, see “ISPF dialog initialization parameters” on page 53.

Decide whether you want to allow the Transaction Analysis Workbench libraries to be set up dynamically each time you start the dialog (as shown in the example command in the previous procedure), or whether you want to add the libraries statically to the relevant concatenations. The decision to use dynamic or static setup depends on the standards at your site. Dynamic setup is the simplest and quickest approach.

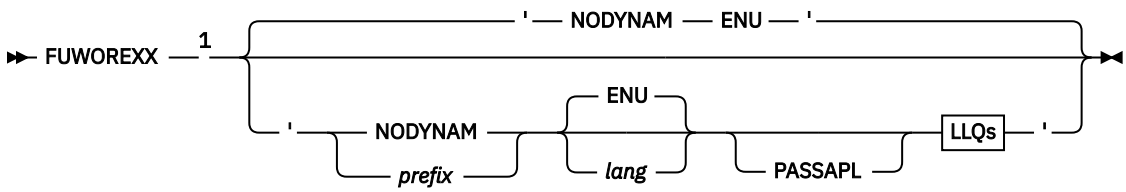
After deciding on dynamic or static setup, consider adding Transaction Analysis Workbench to an ISPF menu, so that users do not have to explicitly enter a command to start the dialog.

Related concepts
ISPF dialog

The Transaction Analysis Workbench ISPF dialog user interface presents a hierarchy of options for interactively browsing logs, managing sessions, generating JCL to run batch jobs, and defining various repository records related to analysis, such as system definitions and filters.

ISPF dialog initialization parameters

FUWOREXX, the REXX exec that starts the Transaction Analysis Workbench ISPF dialog, accepts a parameter string that specifies various initialization values for the ISPF dialog.



Library low-level qualifiers (LLQs)

Notes:

[1] The library low-level qualifiers are only relevant if you also specify *prefix* (dynamic setup). If you specify NODYNAM (static setup), these qualifiers are ignored.

NODYNAM
Instructs the startup REXX exec to not dynamically allocate the Transaction Analysis Workbench libraries. This is the default behavior, and is known as *static setup*. If you choose this option, you must have added the Transaction Analysis Workbench libraries to the appropriate TSO procedure concatenations. Otherwise, you will get various "not found" errors. For details, see “Statically adding libraries to concatenations” on page 54

lang
Identifies the national language. Currently, the only national language that Transaction Analysis Workbench supports is **ENU** (U.S. English).

prefix
Instructs the startup REXX exec to dynamically allocate the Transaction Analysis Workbench libraries, using *prefix* as the high-level qualifier. This is known as *dynamic setup*. For example:

```
EX 'FUW.V1R3M0.SFUWEXEC(FUWOREXX)' 'FUW.V1R3M0'
```

The startup REXX exec uses *prefix* with the default low-level qualifiers of the Transaction Analysis Workbench libraries.

If you have installed the Transaction Analysis Workbench libraries with different low-level qualifiers, you must explicitly specify all of the low-level qualifiers, in the correct order, as shown in the syntax diagram.

PASSAPPL
Allows you to set your own application ID for the Transaction Analysis Workbench ISPF dialog. The default application ID is FUWO.

To set your own application ID, specify PASSAPPL in the FUWOREXX parameter string, and then specify the ISPF command parameter NEWAPPL(*your_application_id*). For example, to set the application ID to TAW, when adding the Transaction Analysis Workbench ISPF dialog to an ISPF menu, set &ZSEL to the following value:

```
'CMD(''prefix.SFUWEXEC(FUWOREXX)'' ''prefix PASSAPPL'') NEWAPPL(TAW)'
```

Low-level qualifiers
If you specify *prefix*, and you installed the Transaction Analysis Workbench libraries using your own low-level qualifiers instead of the defaults, then you must explicitly specify the low-level qualifiers here. Otherwise, the startup REXX exec will be unable to dynamically allocate the libraries, and the Transaction Analysis Workbench ISPF dialog will not start correctly.

Dynamically allocating libraries at ISPF dialog startup

The simplest way to start the Transaction Analysis Workbench ISPF dialog is to allow the startup REXX exec, FUWOREXX, to dynamically allocate the Transaction Analysis Workbench libraries. This is known as *dynamic setup*.

The following procedure assumes that you have installed the Transaction Analysis Workbench libraries using the default low-level qualifiers, listed in "Transaction Analysis Workbench libraries" on page 51. If you used different low-level qualifiers, then you must specify those qualifiers in the parameter string of the startup REXX exec. For details, see "ISPF dialog initialization parameters" on page 53.

Start the Transaction Analysis Workbench ISPF dialog by running the FUWOREXX REXX exec, either directly from a command line or from an ISPF menu.

To start the Transaction Analysis Workbench ISPF dialog from the ISPF Command Shell panel (ISPF primary menu option 6, Command), enter the following command:

```
'prefix.SFUWEXEC(FUWOREXX)' 'prefix'
```

where *prefix* is the high-level qualifier of the Transaction Analysis Workbench libraries installed on your system.

To add the Transaction Analysis Workbench ISPF dialog to an ISPF menu, set &ZSEL to the following value:

```
'CMD(''prefix.SFUWEXEC(FUWOREXX)'' ''prefix'') NOCHECK'
```

Specifying NOCHECK supports users entering concatenated commands via the direct option (trail). Also specify the following statement on the calling panel:

```
&ZTRAIL=.TRAIL
```

Statically adding libraries to concatenations

If you prefer not to have the Transaction Analysis Workbench libraries dynamically allocated each time you start the ISPF dialog, you can instead add the libraries to the appropriate ISPF concatenations in your TSO logon procedure. This is known as *static setup*.

1. Edit your TSO logon procedure, and add the Transaction Analysis Workbench library *prefix*.SFUWEXEC to either your SYSEXEC (recommended) or SYSPROC concatenation.

   The SFUWEXEC library contains the REXX execs required to start the Transaction Analysis Workbench ISPF dialog. It is allocated with fixed-block 80 record format during installation.

   If you add this library to your SYSPROC, then your SYSPROC must have a record length of 80 bytes.
2. Add the remaining Transaction Analysis Workbench libraries to the following ISPF library concatenations.

Library	Contains	Add to this ISPF concatenation
SFUW**L**INK	Executable load modules	ISP**L**LIB
SFUW**M**ENU	ISPF messages	ISP**M**LIB
SFUW**P**ENU	ISPF panels	ISP**P**LIB
SFUW**S**ENU	ISPF skeleton JCL	ISP**S**LIB

Library	Contains	Add to this ISPF concatenation
SFUW**T**ENU	ISPF input tables	ISP**T**LIB

3. Logon to TSO using the updated logon procedure.
4. Start the Transaction Analysis Workbench ISPF dialog by running the FUWOREXX REXX exec, either directly from a command line or from an ISPF menu.

   To start the Transaction Analysis Workbench ISPF dialog from the **ISPF Command Shell** panel (ISPF option 6 **Command**), enter the following command:

   ```
   %FUWOREXX
   ```

   To add the Transaction Analysis Workbench ISPF dialog to an ISPF menu, set &ZSEL to the following value:

   ```
   'CMD(%FUWOREXX) NOCHECK'
   ```

Verifying installation

Starting the Transaction Analysis Workbench ISPF dialog, and displaying the primary option menu, verifies that you have successfully installed Transaction Analysis Workbench. Next, each user must configure personal profile settings, and then verify that configuration.

Related concepts

Configuring Transaction Analysis Workbench

Every new Transaction Analysis Workbench user should follow the instructions here to configure their ISPF settings and their Transaction Analysis Workbench ISPF dialog profile settings.

Installing Common Services Library server

If you want to use the Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer), you need to install Common Services Library server on z/OS. The plug-in uses Common Services Library server to communicate with Transaction Analysis Workbench.

Common Services Library server is a component of IBM Common Services Library for z/OS, V1.1 (Common Services Library), a no-charge product.

Other products also use Common Services Library server. If you have already installed the same release of Common Services Library server to support another product, you do not need to install the server again: skip the following procedure, and configure your existing server to support Transaction Analysis Workbench. You can either start separate instances of the server configured for each product, or you can configure the server to support more than one product.

To install Common Services Library server:

1. Get Common Services Library from IBM.
2. Follow the instructions in the *Common Services Library Program Directory*.

   Common Services Library server consists of members in the following two target libraries:

   SFUNLINK
   Contains Common Services Library server load modules. Must be APF-authorized.

   SFUNSAMP
   Contains sample Common Services Library server startup JCL and configuration files.

   Common Services Library server introduces no installation prerequisites beyond those required by Transaction Analysis Workbench.

Verify that you have successfully installed Common Services Library server.

Related tasks

Installing the z/OS Explorer plug-in

The Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) provides a graphical user interface (GUI) to some of the functions provided by the Transaction Analysis Workbench ISPF dialog. The plug-in communicates with Transaction Analysis Workbench via Common Services Library server.

Verifying installation of Common Services Library server

You should verify that Common Services Library server starts before you configure it to support the Transaction Analysis Workbench plug-in.

The following procedure describes how to start Common Services Library server. This procedure is independent of Transaction Analysis Workbench. Later procedures describe how to configure the server to work with Transaction Analysis Workbench.

1. Copy the following three members from the Common Services Library server sample library SFUNSAMP to a data set of your choice, and then edit the copies according to the comments inside each member.

   FUNSRVST
   Startup JCL:

   ```
   //SERVER   EXEC PGM=FUNSRV,
   //    PARM=('BPECFG=FUNBPECF,FUNCFG=FUNCONFG') 1
   //*
   //STEPLIB  DD  DISP=SHR,DSN=FUNHLQ.SFUNLINK 2
   //PROCLIB  DD  DISP=SHR,DSN=MY.FUN.PROCLIB 3
   ```

   1
   The **PARM** parameter specifies the member names of the two configuration files required by Common Services Library server. These members must be in the concatenation specified by the PROCLIB DD statement (3). In this example, both members belong to the same data set, MY.FUN.PROCLIB.

   2
   In this example, FUNHLQ is the high-level qualifier of the data set where you have installed the Common Services Library server load module library, SFUNLINK.

   3
   The PROCLIB DD statement specifies the location of the Common Services Library server configuration files.

   FUNBPECF
   IMS Base Primitive Environment (BPE) configuration parameter file. Common Services Library server uses BPE services. The BPE configuration file defines the BPE execution environment settings for the server.

   Unless you have a specific requirement to set different tracing options, use the sample member as supplied.

   FUNCONFG
   Common Services Library server configuration file.

   You must edit the sample member to specify your own site-specific values for the following parameters:

   SERVER_NAME
   The name of this Common Services Library server.

   TCP_PORT
   The TCP/IP port number on which the server listens for messages from the Transaction Analysis Workbench plug-in.

   For the other parameters, you can either use the default values or specify values according to your site-specific requirements.
2. Submit the startup JCL.
3. View the JESMSGLG job output data set.

If Common Services Library server started successfully, the JESMSGLG data set contains the following message:

```
FUN3226I Server start completed
```

4. Stop the server.

   For example, enter the following MVS operator command:

   F *jobname*,SHUTDOWN

Configure the Common Services Library server startup JCL and Common Services Library server configuration file to support the Transaction Analysis Workbench plug-in.

BPECFG: Common Services Library server BPE configuration file

You need to configure the Common Services Library server by setting options in the BPECFG file.

The BPECFG file can specify the following parameters. For an example file, see member FUNBPECF of the Common Services Library server sample library SFUNSAMP.

Unless you have a specific requirement to set different tracing options, use the sample member as supplied.

LANG=ENU
The language of BPE and IMS component message text. ENU is for US English, which is currently the only supported language. This parameter is required.

TRCLEV=(*type*,*level*,*component*,PAGES=*num_pages*)
The trace level for a trace table and, optionally, the number of storage pages allocated for the trace table.

The supported values of *type* are:

BPE
Sets tracing options for the BPE.

FUN
Sets tracing options for Common Services Library server. It is recommended that you leave these trace levels at high.

FUNCFG: Common Services Library server configuration file

You need to configure the Common Services Library server by setting parameters in the FUNCFG file.

The FUNCFG file can contain the following parameters. For an example configuration file, see member FUNCONFG in the Common Services Library server sample library SFUNSAMP.

SERVER_NAME=*name*
1 - 8 alphanumeric character server name. The name must be unique across the sysplex. This is a required parameter.

PRODUCT=*prd*
A 3-character product code representing a product to be supported by the server. For example, FUW for Transaction Analysis Workbench. A server can support multiple products. Specify a **PRODUCT** parameter for each product. If you do not specify any products, then you will only have access to basic server administration functions.

TCP_NAME=*name*
A 1 - 8 character name of the TCP/IP stack. If this parameter is omitted or blanks are specified, the server uses the default TCP/IP stack.

TCP_PORT=*port*
The TCP/IP port number that the server listens on: 1 - 65535. This parameter is required. Consult your network administrator to identify a suitable (not in use) port.

TCP_THREADS=*threads*
The maximum number of threads that can accept client connections concurrently: 0 - 64. The default is 16.

TCP_MAXSOC=*sockets*
The maximum number of TCP sockets available for concurrent client connections: 50 - 2048. The default is 50.

TCP_IPV6=Y|N
Whether the server supports IPv6 clients. Specify Y to allow IPv6 clients to connect to the server. Your TCP/IP stack must be configured for IPv6; if it is configured to also allow IPv4 clients, then the server will support both. The default is N: the server supports only IPv4 clients, regardless of the stack configuration.

CCSID=*ccsid*
Specifies the coded character set identifier (CCSID) for the server: 1 - 65533. The CCSID must specify a single-byte character set (SBCS) that is supported by z/OS Unicode Services. The special identifiers 0, 65534, and 65535 are not supported. The default is 37.

SAF_CLASS=*class*
The 1 - 8 character SAF security class name, used for product access authorization. If this parameter is omitted or explicitly set to blanks, then product access authorization is not performed.

SDA_BARLIM=*kilobytes*
The Session Data Area (SDA) bar limit size in kilobytes: 64 - 4096. An SDA is used to hold any incoming client request data and subsequently any outgoing client response data generated for the request. An SDA of a length that exceeds the SDA_BARLIM will reside above the bar. If this parameter is omitted, the default is 2048 kilobytes.

SDA_MAXLEN=*megabytes*
The Session Data Area (SDA) maximum length in megabytes: 4 - 100. An SDA is used to hold incoming client request and outgoing client response data. A client request with data that exceeds the SDA_MAXLEN will fail. If this parameter is omitted the default is 32 megabytes.

Common Services Library server security

Common Services Library server can check whether users are authorized to use a product. Common Services Library server performs actions according to the authority of the client user ID.

Access authorization for basic server functions

In addition to the products that are specified by the **PRODUCT** parameter in the Common Services Library server configuration file, Common Services Library server starts its own default product, with product code FUD, that provides basic functions such as verifying connections with clients. If the Common Services Library server configuration file specifies a **SAF_CLASS** parameter, the server performs a security check for that default product using the following general resource profile:

FUNPRD.FUD

If the user has at least READ access for this resource profile, Common Services Library server allows access to the basic functions.

Users of the FUW plug-in must have at least READ access to this resource profile.

Product access authorization

Products running under Common Services Library server manage authorization internally, within the constraints of the Common Services Library server environment.

Optionally, Common Services Library server can restrict access to each product. If the Common Services Library server configuration file specifies a **SAF_CLASS** parameter, the server performs a security check using the following general resource profile:

FUNPRD.*product*

where *product* is one of the 3-character product codes specified by the **PRODUCT** parameter in the Common Services Library server configuration file. For example, FUW is the product code for Transaction Analysis Workbench.

If the user has at least READ access for this resource profile, Common Services Library server allows access to that product. Otherwise, Common Services Library server denies access to that product.

Users of the FUW plug-in must have at least READ access to this resource profile.

Client user ID authentication

Common Services Library server authenticates the client user ID when a client establishes a connection with the server. Client request threads running in the target product are associated with the user ID of the connected client.

Starting Common Services Library server

To start Common Services Library server, you submit an MVS batch job.

1. Customize the JCL in the FUNSRVST member of the Common Services Library server sample library SFUNSAMP.
2. Submit the batch job.

Stopping Common Services Library server

To stop an instance of Common Services Library server, you stop the corresponding MVS batch job.

Enter one of the following MVS operator **MODIFY** (**F**) or **STOP** (**P**) commands:

Option	Description
F *jobname*,SHUTDOWN	Quiesce the server before shutting down. The server rejects new client request threads and shuts down when all active client request threads have completed.
F *jobname*,SHUTDOWN FORCE	Force the server to shut down immediately, cancelling any active client request threads. You can upgrade a quiesce shut down to a forced shut down; see the following command **SHUTDOWN FORCE** command.
P *jobname*	Quiesce the server before shutting down. This **STOP** command is a shorthand alternative to F *jobname*,SHUTDOWN, with the following difference: the server will not respond to subsequent **MODIFY** commands, so you cannot upgrade this request to a forced shut down.

where *jobname* refers to the batch job for the instance of the server that you want to stop.

Common Services Library server administrative functions

Common Services Library server provides administrative functions that allow you to control the server and the products it runs.

The Common Services Library server accepts operator commands to perform many operations. The format of the command is:

```
F servername, command
```

Where the commands include the following

DISPLAY PRODUCT ***product_code***
Displays information about a particular product.

RESTARTIP
Restarts the TCP/IP layer.

SHUTDOWN
Shuts down the server, waiting for any products to complete their functions.

SHUTDOWN FORCE
Forces shut down even if some products have not responded.

START PRODUCT ***product_code***
Starts a product with the given code.

STOP PRODUCT ***product_code***
Stops a product with the given code.

Installing the z/OS Explorer plug-in

The Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) provides a graphical user interface (GUI) to some of the functions provided by the Transaction Analysis Workbench ISPF dialog. The plug-in communicates with Transaction Analysis Workbench via Common Services Library server.

- Install Common Services Library server on z/OS.
- Install z/OS Explorer on your PC.

The following figure shows how the environment required to run the plug-in spans systems.

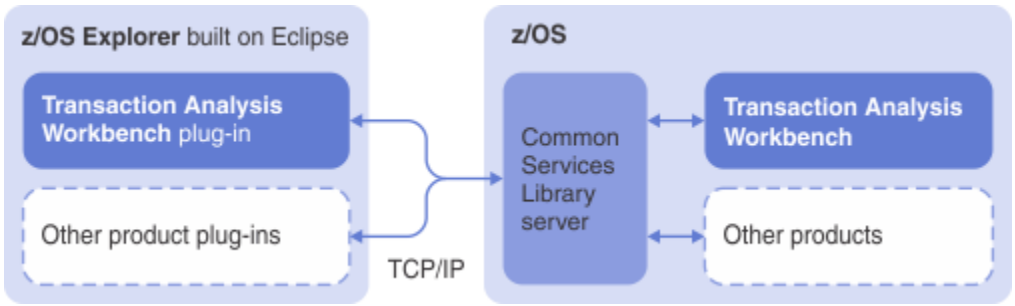


Figure 11. Transaction Analysis Workbench plug-in environment

1. Copy the Transaction Analysis Workbench load modules required for use with Common Services Library server from the SFUWLINK library to an APF-authorized library of your choice.

   To perform Transaction Analysis Workbench functions, Common Services Library server requires some Transaction Analysis Workbench load modules. All libraries in the Common Services Library server STEPLIB concatenation must be APF-authorized.

   To copy the load modules, edit and submit the JCL in member FUWAPF of the sample library SFUWSAMP.
2. Configure Common Services Library server to support Transaction Analysis Workbench:
   a) Add the parameter PRODUCT=FUW to the Common Services Library server configuration file.
   b) Modify the Common Services Library server startup JCL for Transaction Analysis Workbench.
   c) Create the Transaction Analysis Workbench control library members.
3. Follow the instructions in the readme supplied with Transaction Analysis Workbench in member FUWREAD of library SFUWGENU.

Related concepts
z/OS Explorer plug-in

You can use the Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) to view or create sessions, submit z/OS batch jobs for session workflow tasks, and view the output of those batch jobs.

Related tasks

Installing Common Services Library server

If you want to use the Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer), you need to install Common Services Library server on z/OS. The plug-in uses Common Services Library server to communicate with Transaction Analysis Workbench.

Common Services Library server startup JCL to support Transaction Analysis Workbench

To use the Transaction Analysis Workbench plug-in, you need to modify the Common Services Library server startup JCL to support Transaction Analysis Workbench.

The following Common Services Library server startup JCL shows the modifications required to support Transaction Analysis Workbench:

```
//FUNSRV   JOB ,CLASS=A,MSGCLASS=A,REGION=0M
//ZSERVER  EXEC PGM=FUNSRV,
//    PARM=('BPECFG=<BPECONFG>,FUNCFG=<FUNCONFG>')
//STEPLIB  DD  DISP=SHR,DSN=<FUNHLQ>.SFUNLINK  1
//PROCLIB  DD  DISP=SHR,DSN=<FUNHLQ>.SFUNSAMP  2
//FUNPRINT DD  SYSOUT=*
//FUWPRINT DD  SYSOUT=*  3
//FUWCNTL  DD  DISP=SHR,DSN=MY.FUW.PROCLIB  4
```

Figure 12. Common Services Library server startup JCL to support Transaction Analysis Workbench

1

Ensure that the STEPLIB concatenation includes the library where you copied the Transaction Analysis Workbench load modules required by Common Services Library server. In this example JCL, rather than adding another library to the STEPLIB concatenation, the Transaction Analysis Workbench load modules have been copied to the Common Services Library server load module library, SFUNLINK. All libraries in the STEPLIB concatenation must be APF-authorized.

2

Add the parameter PRODUCT=FUW to the Common Services Library server configuration file.

The Common Services Library server configuration file is a member of the partitioned data set specified by the PROCLIB DD statement. The member name is specified by the value of the **FUNCFG** parameter in the **PARM** parameter of the EXEC statement for the FUNSRV program.

PRODUCT=FUW instructs Common Services Library server to start Transaction Analysis Workbench and make it available to users. The configuration file might also contain PRODUCT parameters for other products.

3

Add a DD statement for FUWPRINT, the output data set for the Transaction Analysis Workbench event log. This event log contains messages from Transaction Analysis Workbench that are about running with Common Services Library server.

4

Add a DD statement for FUWCNTL, the Transaction Analysis Workbench control library. This library contains members that control how Transaction Analysis Workbench operates with Common Services Library server.

Either allocate a new data set or reuse an existing control or PROCLIB library on your system. There is no control library supplied with Transaction Analysis Workbench. The control library must be a PDS or library allocated with the attributes RECFM=FB and LRECL=80.

Create the following members in the control library: FUWCNTL, FUWVARS, and, optionally, JOBCARD.

Transaction Analysis Workbench control library for Common Services Library server

The Common Services Library server startup JCL must contain an FUWCNTL DD statement that identifies the Transaction Analysis Workbench control library. This library contains members that control how Transaction Analysis Workbench operates with Common Services Library server.

You must create this library and its members. Transaction Analysis Workbench does not provide a sample control library or members.

Either allocate a new data set or reuse an existing control or PROCLIB library on your system.

The control library must be a PDS or a library. The control library must have the attributes `RECFM=FB` and `LRECL=80`.

FUWCNTL: Transaction Analysis Workbench startup member

The startup member of the Transaction Analysis Workbench control library lists the session repositories that will be visible in the plug-in.

FUWCNTL must contain one or more **REPOSITORY** commands. Each **REPOSITORY** command identifies a session repository that you want to be visible in the plug-in.

Format

name
The name that you want the plug-in to use for this session repository. 1 - 8 national or uppercase alphanumeric characters without spaces: @, #, $, A - Z, 0 - 9.

description
The description that you want the plug-in to use for this session repository. 1 - 40 mixed-case characters with spaces. If the description includes an apostrophe, then enclose the description in double quotation marks.

data_set_name
The fully qualified data set name of the session repository, without enclosing quotation marks.

The session repository must already exist. To create a session repository, use the ISPF dialog: on the Transaction Analysis Workbench Primary Option Menu, type a new session repository data set name in the **Session Repository** field, and then select option 1 **Sessions**.

REPOSITORY commands must be specified within columns 1 - 71.

You can split a **REPOSITORY** command over multiple lines: you can specify the **DESC** and **DSN** parameters on separate consecutive lines immediately following the line that contains the **REPOSITORY** command and the **NAME** parameter.

Blank lines and comment lines can occur between **REPOSITORY** commands. A comment line begins with an asterisk (*) in column one.

Example

The following example startup member identifies two session repositories:

```
* Session repositories to be displayed in the plug-in
REPOSITORY NAME=HELPDESK,
           DESC=(Production helpdesk),
           DSN=FUW.HELPDESK.SESSIONS

REPOSITORY NAME=JOHN#1,
           DESC=("John's #1 repository"),
           DSN=JCH.FUW.SESSIONS
```

Figure 13. FUWCNTL: Example Transaction Analysis Workbench startup member for Common Services Library server

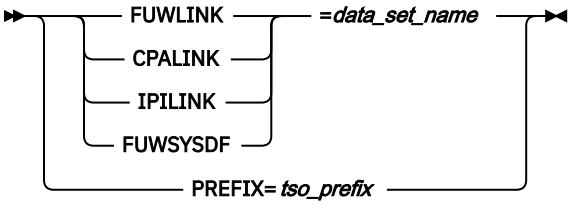
FUWVARS: Workflow task JCL substitution variables

The FUWVARS member of the Transaction Analysis Workbench control library defines substitution variables that the plug-in uses to resolve workflow task JCL.

FUWVARS defines *user profile* substitution variables: if you submit a workflow task from the Transaction Analysis Workbench ISPF dialog rather than the plug-in, the ISPF dialog gets the values for these variables from the profile settings of the TSO user.

Format

Each line of FUWVARS can contain one of the following variable definitions:



FUWLINK
Transaction Analysis Workbench load module library

CPALINK
CICS Performance Analyzer load module library

IPILINK
IMS Performance Analyzer load module library

FUWSYSDF
Transaction Analysis Workbench CICS and related system definition repository

PREFIX
TSO PROFILE PREFIX for new data sets

Variable definitions must start in column 1. Spaces are not allowed before or after the equal sign.

FUWVARS can also contain comment lines, inline comments, and blank lines. A comment line begins with an asterisk (*) in column 1. An inline comment begins with a slash followed by an asterisk (/*), after column 1 and after any variable definition on the line.

Example

```
* Product libraries
FUWLINK=FUWVRM.DEVT.SFUWLINK  /* Transaction Analysis Workbench
CPALINK=CPAVRM.DEVT.SCPALINK  /* CICS Performance Analyzer
IPILINK=IPIVRM.DEVT.SIPILINK  /* IMS Performance Analyzer

FUWSYSDF=JCH.CICSPA.SYSDEF    /* System definitions for SMF file selection

PREFIX=FUWVRM                 /* TSO PROFILE PREFIX for new data sets
```

Related reference

Workflow task JCL substitution variables

The JCL for workflow tasks can contain substitution variables that are specific to Transaction Analysis Workbench.

JOBCARD: Job card for workflow tasks

The JOBCARD member of the Transaction Analysis Workbench control library specifies the job card that the plug-in uses to submit jobs for workflow tasks.

The job card can contain substitution variables.

The JOBCARD member is optional. If the JOBCARD member does not exist, the plug-in uses the following default job card:

```
//%USERID1 JOB ,NOTIFY=&SYSUID
```

where the job name is derived from the plug-in user ID; job and message classes assume system defaults.

Example

If the plug-in user name is JAMES, then the following JOBCARD member:

```
//%USERID1  JOB (HELPDESK),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
```

generates the following job card:

```
//JAMES1    JOB (HELPDESK),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
```

Chapter 3. Configuring Transaction Analysis Workbench

Every new Transaction Analysis Workbench user should follow the instructions here to configure their ISPF settings and their Transaction Analysis Workbench ISPF dialog profile settings.

To configure Transaction Analysis Workbench, you need to know how to start the Transaction Analysis Workbench ISPF dialog. Contact the person at your organization who installed Transaction Analysis Workbench.

Related tasks

Verifying installation

Starting the Transaction Analysis Workbench ISPF dialog, and displaying the primary option menu, verifies that you have successfully installed Transaction Analysis Workbench. Next, each user must configure personal profile settings, and then verify that configuration.

Recommended ISPF settings

The Transaction Analysis Workbench ISPF dialog follows CUA conventions. These recommendations describe how to set up your ISPF environment to use the Transaction Analysis Workbench ISPF dialog efficiently.

CUA attributes: point-and-shoot fields

The Transaction Analysis Workbench ISPF dialog is designed to use the default CUA attributes, with one exception: it is recommended that you set distinctive display attributes for point-and-shoot fields (for example, yellow with high intensity). To change the attributes of point-and-shoot fields, enter the ISPF command **PSCOLOR**, as shown in the following figure:

```
                    CUA Attribute Change Utility
Command ===> ____________________________________________________ Defaults

Change colors, intensities, or highlights for panel attribute elements.
Enter the EXIT command to save changes or enter the CANCEL command to exit
without saving. To restore the defaults for a type, clear the field and
press Enter or select the Defaults point-and-shoot field to restore all
default settings for all types.

Panel Element                Color         Intensity  Highlight
                                                            More:   -
Point-and-Shoot  . . . . . . YELLOW        HIGH       NONE

...
```

Figure 14. Panel: Example CUA attribute settings to differentiate point-and-shoot fields

Function keys

The Transaction Analysis Workbench ISPF dialog uses function keys extensively. To show or hide the function key assignments at the bottom of each panel, enter the ISPF command **PFSHOW ON** or **PFSHOW OFF**. Show the assignments until you are familiar with them.

To assign alternative functions to the keys, use the ISPF commands **KEYS** and **KEYLIST**. To display the Transaction Analysis Workbench default settings for the function keys, enter the **KEYSHELP** command in the Transaction Analysis Workbench ISPF dialog, or select **Help** > **Keys Help** in the action bar.

Prompt fields

Some entry fields have a Prompt action that allows you to fill in the field by selecting a value from a pop-up list of valid values. Prompt fields are indicated by a plus sign (**+**) at the end of the field. To display the pop-up list, move the cursor to the field and press the Prompt function key (F4).

Tabbing to point-and-shoot fields

The Transaction Analysis Workbench ISPF dialog uses point-and-shoot fields. To be able to tab to point-and-shoot fields, enter the ISPF command **SETTINGS**, and then select **Tab to point-and-shoot fields**, as shown in the following figure:

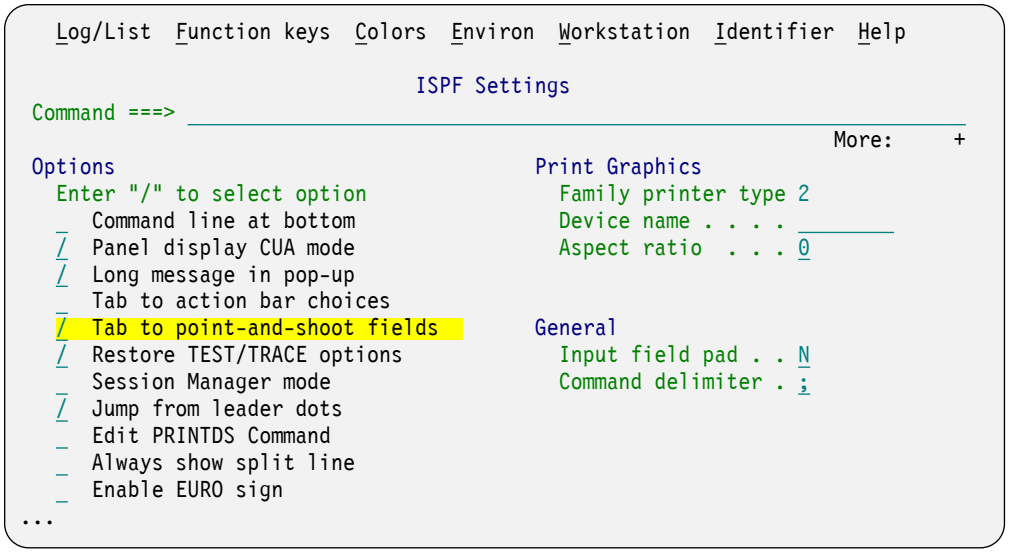


Figure 15. Panel: Set ISPF to tab to point-and-shoot fields

Mouse options

If your terminal emulation allows, it is recommended that you configure your mouse options to assign the Enter key to left mouse button double-click. This allows you to left button double-click on point-and-shoot fields and makes functions such as field zoom (where the log browser displays log record field names as point-and-shoot fields) quicker and easier to use.

Panel size and scrolling

Transaction Analysis Workbench ISPF dialog panels are optimized for 32 lines, but accommodate 24 lines using scrolling with the Backward function key (F7) and the Forward function key (F8).

When Transaction Analysis Workbench displays formatted records, the amount of data in the viewing window is optimized according to the width of the screen, either 80 or 132 characters wide.

Displaying messages

The Transaction Analysis Workbench ISPF dialog uses both long and short messages. Short messages are displayed on the same line as the panel title, in the top right of the panel. Long messages are designed to display in a pop-up window when you press the Help function key (F1) after the short message is displayed. However, long messages of less than the panel width can be customized to display just below or above the command line rather than in a window. To always display long messages in a pop-up window, enter the **SETTINGS** command to display the **ISPF Settings** panel, and then select **Long message in pop-up**.

To move messages displayed in a window to another location on the panel:

1. Move the cursor to the top or bottom border of the message window, and then press Enter.

2. Move the cursor to the location on the panel where you wish to move the message, and then press Enter.

Required Transaction Analysis Workbench ISPF dialog profile settings

Transaction Analysis Workbench ISPF dialog option 0 **Profile** contains settings that are specific to each user. Some of these settings are required, in the sense that they must have appropriate values for Transaction Analysis Workbench to perform typical functions correctly. Other settings are required only for more specific functions, or depend on your personal preferences.

The first time that a new user starts the Transaction Analysis Workbench ISPF dialog, the dialog sets the initial value of various profile settings. These initial values enable new users to start work immediately without having to set the values manually. However, the first time that you start the ISPF dialog, you should review these values to confirm that they are correct. The required fields are described here under their panel titles. For details of other fields, see the ISPF dialog online help.

Option 0.1 Workbench Personal Settings

Workbench Load Library

The data set name of the library that contains the Transaction Analysis Workbench executable load modules.

Transaction Analysis Workbench supplies this library with the default low-level qualifier `SFUWLINK`. Your site might have installed this library with a different low-level qualifier.

The ISPF dialog uses this setting for the following purposes:

- To locate knowledge modules, when browsing logs.
- To generate the STEPLIB DD statement, when generating JCL for the batch report and extract utility or the automated file selection utility.

A blank value is allowed for this field, and has the following effects:

- When browsing logs, the dialog searches for the knowledge modules in the standard ISPF search order, including the ISPLLIB, STEPLIB, and LNKLST concatenations.
- When generating JCL, the dialog omits the STEPLIB DD statement. If the batch program is not in the standard z/OS search order, such as the LNKLST concatenation, the job fails.

  If a new user starts the dialog for the first time via static setup, rather than via dynamic setup, then the dialog does not set the initial value of this field. Many sites typically do not add application libraries such as this to their LNKLST concatenations. For this reason, if a new Transaction Analysis Workbench user starts the ISPF dialog for the first time via static setup, then before using the dialog to generate JCL, they should explicitly specify the correct value for this field.

If you select a dialog option to browse logs (such as session menu option 5 **Investigate**), but the dialog cannot locate knowledge modules, then the dialog displays the error message number FUW0050E (LOAD failed), indicating that you have not specified an appropriate value in this field.

Option 0.2 Repositories

Session

The data set name of the repository where Transaction Analysis Workbench saves session information.

For details on how Transaction Analysis Workbench uses repositories, see “Repositories” on page 39.

Option 0.4 Time Control

The following field provides the default time zone setting for new sessions:

Time Zone
The time zone of the system that created the logs that you want to analyze. Each session can specify a different time zone. When you create a session, this field provides the default value of the **Zone** field for the new session.

For logs that were created locally (on the same system on which you are running Transaction Analysis Workbench, or on a system in the same time zone), specify LOCAL. Otherwise, you must specify a time zone offset, such as -0400 (New York), or GMT (no offset).

Option 0.5 IMS Tools Settings

If you want to use IMS logs with Transaction Analysis Workbench, then you must specify an appropriate value for the following field:

IMS Release
The IMS release that generated the IMS logs you want to use. This determines the knowledge modules that Transaction Analysis Workbench uses to format IMS log records. The format of IMS log records is release-specific.

Verifying your configuration

To verify your configuration of Transaction Analysis Workbench, complete the first two tutorials.
Related concepts
Tutorials
Follow these step-by-step tutorials to help you get started with Transaction Analysis Workbench.

Defining systems for log file selection

Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

To use automated file selection, you specify one or more system names and a time period, and then you run the automated file selection utility to identify the data set names of the corresponding log files. You can use automated file selection for DB2 logs, IMS logs, IMS Connect Extensions journals, and SMF files.

System definitions enable the automated file selection utility to locate the appropriate log files.

If you do not use automated file selection, you do not need to define any systems.

You can define systems to Transaction Analysis Workbench either in a batch job using the system take-up utility, or interactively in the ISPF dialog.

To define a system to Transaction Analysis Workbench using the ISPF dialog:

1. On the Transaction Analysis Workbench Primary Option Menu, select option 3 **Systems**.
2. Select the system type.
3. On the command line, enter **NEW** to define a new system.

   Specify the details required to locate the log files for this system. For example:

   DB2 logs
   The DB2 bootstrap data set (BSDS) name.

   IMS logs
   The RECON data set names.

   SMF files
   The generation data group (GDG) conventions that your site uses to keep historical SMF data.

   Tip: Rather than specifying details of cyclic SMF files and SMF log streams in each CICS, DB2, or MQ system definition, define an MVS Image system definition with those details, and, when you define CICS, DB2, or MQ system definitions, refer to the MVS Image name. If you use the automated file

selection utility to locate the SMF records for CICS, DB2, or MQ system definitions, then, if those definitions do not contain their own details for cyclic SMF files and SMF log streams, the automated file selection utility uses the details from the associated MVS Image system definition.

For more information on defining each system type, see the online help.

Often, a problem can involve a group of related systems. To reflect this, you can arrange system definitions into groups. Groups are useful when you are creating a session and adding the systems involved. Rather than adding each system individually, you can add a group of systems in a single step.

Restriction: A group can only contain systems that are stored in the same system definition repository. For example, a single group can contain MVS images, CICS systems, and DB2 systems, because those types of system definition are stored in the same repository; but that group cannot contain an IMS system. Similarly, a single group can contain IMS systems and IMS Connect systems; but that group cannot contain DB2 systems.

Related concepts

Repositories
Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Related tasks

Adding log files to a session
If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Tutorial: Using the plug-in to create a session and perform a workflow
This tutorial shows you how to use the Transaction Analysis Workbench Eclipse plug-in to create a session based on a template, and then perform the workflow tasks inherited from the template. The tutorial also covers the prerequisite steps that need to be performed in the ISPF dialog by a subject-matter expert: defining systems, and then creating a session template that contains workflow tasks for those systems.

Tutorial: Automating selection of IMS logs
This tutorial shows you how to define an IMS system to Transaction Analysis Workbench, and then use that system definition to locate the related IMS log files (SLDS or OLDS) for a particular time interval.

Tutorial: Automating selection of DB2 logs
This tutorial shows you how to define a DB2 system to Transaction Analysis Workbench, and then use that system definition to locate the related DB2 log files for a particular time interval.

Registering sessions
Registering (creating) a session is the recommended first step for using Transaction Analysis Workbench to analyze a problem.

Related reference

System and file types
System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

System take-up utility
The system take-up utility reads a plain-text file of system definitions, and then defines the corresponding systems in Transaction Analysis Workbench system definition repositories.

Chapter 4. Tutorials

Follow these step-by-step tutorials to help you get started with Transaction Analysis Workbench.

Related concepts

Scenarios

These step-by-step scenarios demonstrate using Transaction Analysis Workbench to analyze specific types of transaction.

Related tasks

Verifying your configuration

To verify your configuration of Transaction Analysis Workbench, complete the first two tutorials.

Tutorial: Creating a session and browsing a log

This tutorial shows you how to create a session for a problem, manually specify a log file for the problem, and then browse the log file.

- If you have not already done so, follow the instructions in Chapter 3, "Configuring Transaction Analysis Workbench," on page 65.
- Determine the data set name or the log stream name of a log file that you want to analyze. This log file can be any of the types that are supported by Transaction Analysis Workbench. This tutorial uses an SMF file.

This tutorial is intended for first-time users. You can use it to verify that you have correctly configured your Transaction Analysis Workbench ISPF dialog profile settings (under option 0 **Profile**).

1. Start the Transaction Analysis Workbench ISPF dialog.

   For example:

   a. On the ISPF primary option menu, select option 6 **Command**.

   b. Enter the following command. Replace `FUW.V1R3M0` with the high-level qualifiers of the Transaction Analysis Workbench libraries installed on your system.

   ```
   EX 'FUW.V1R3M0.SFUWEXEC(FUWOREXX)' 'FUW.V1R3M0'
   ```

   The following steps register (create) a session.

2. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**.

```
  File  Help

            Transaction Analysis Workbench - Primary Option Menu
Option ===> 1

0 Profile     Customize your dialog profile
1 Sessions    Analyze problems using the session manager
2 Controls    Define record filtering and formatting controls
3 Systems     Define the systems where transactions are processed
4 Process     Analyze ad hoc log files
5 Big Data    Extract, transform, and load logs to a big data platform
X Exit        Quit the workbench

Session Repository  . . . FUW.SESSIONS                              +
```

Figure 16. Panel: Transaction Analysis Workbench Primary Option Menu

If the session repository does not yet exist, a **Define Repository Data Set** panel is displayed; press Enter to create the repository.

The **Session Manager** panel is displayed. This panel shows the list of existing sessions. To scroll the list horizontally and see more columns, press the Right function key (F11).

3. Enter **NEW** on the command line.

```
  File  View  Help

                              Session Manager           Row 1 of 16 More: < >
Command ===> NEW                                         Scroll ===> PAGE

NEW Register a new Session

/  Key       Status  Description
_  00000001  OPEN    Installation verification
****************************** Bottom of data ********************************
```

*Figure 17. Panel: **Session Manager** (list of existing sessions)*

The **New Session** window opens, prompting for an optional template. In this example, we will create a session without using a template.

4. Press Enter without specifying a template.

The **Session Details** panel for the new session is displayed.

5. Specify a description such as `Getting started`. Leave the other details blank or accept their default values.

```
  File  Help

                              Session Details                Row 1 to 1 of 1
Command ===> _                                           Scroll ===> PAGE

Key . . . . : 00000002
Description . Getting started
Severity  . . _
Reference . . ________________________________     When problem occurred
Reported by . ________________________________       YYYY-MM-DD  HH.MM.SS.TH
Assigned to . ________________________________  From ___________  ___________
Status  . . . OPEN                             To   ___________  ___________
Template  . . ________  +                       Zone LOCAL

Systems involved:

/  System +   Type +

__ ________   ________
****************************** Bottom of data ********************************
```

*Figure 18. Panel: **Session Details***

6. Press the Exit function key (F3) to save the session.

The menu for the new session is displayed. The menu title contains the unique session key identifier (for example, **Session 00000002**).

The following steps select a log file for the new session.

7. On the session menu, select option 3 **Files**.

```
  File  Help

                          Session 00000002          Session 00000002 saved
Option ===> 3

Description : Getting started

1  Register      Update the problem registration details
2  Workflow      Perform the diagnostic tasks
3  Files         Locate and manage the log files required for diagnosis
4  Reporting     Run batch reports
5  Investigate   Perform interactive log file analysis
6  History       Review the problem history
```

*Figure 19. Panel: **Session** menu*

The **Locate and Manage Log Files** panel is displayed.

```
  File  Help

                       Locate and Manage Log Files
Command ===> NEW                                          Scroll ===> PAGE

NEW  Insert a new log file.
AUTO Run automated file selection to locate log files.

Log Files:
                                                          System      File
/   Exc Data Set Name                                    Name     Type    Type
******************************** Bottom of data ********************************
```

*Figure 20. Panel: **Locate and Manage Log Files** panel*

8. Manually specify your SMF file:
   a) Enter **NEW** on the command line.

      The **Specify File Details** window opens.

```
  File  Help

                          Specify File Details              Row 1 to 5 of 5
Command ===>

Specify file details then press EXIT to save.

System Name . . . MVSA     +                  When the system is IMS:
System Type . . . IMAGE    +                    IMS Release ___   +
File Type . . . . SMF      +  _  Log stream
Source  . . . . . MANUAL   +

/  Data Set Name                                  UNIT +    VOLSER
_  ©CPPX.DEMO.SMF©                                ________  ______
_  ____________________________________________  ________  ______
_  ____________________________________________  ________  ______
_  ____________________________________________  ________  ______
_  ____________________________________________  ________  ______
****************************** Bottom of data ******************************
```

Figure 21. Panel: Manually specifying an SMF file for a session (1 of 2)

   b) Specify the following file details:

Field	Value	Notes
System Name	The SYSID of the MVS system that created the SMF file	The system name does not have to match an MVS system definition. That is, you do not need to have previously defined this MVS system to Transaction Analysis Workbench under primary menu option 3 **Systems**. If you do want to select from the list of systems defined to Transaction Analysis Workbench, tab to the **System Name** field, and then press the Prompt function key (F4).
System Type	IMAGE	IMAGE is an abbreviation of "MVS image".
IMS Release	Blank	Required only for system type IMS.
File Type	SMF	
Log stream	Unselected	In this example, we are using a data set, not a log stream.
Source	MANUAL	Indicates that the file that has been manually entered into the list of session files.
Data Set Name	The fully qualified data set name of the SMF file, enclosed in single quotes	If you omit the enclosing single quotes and your TSO user profile specifies a prefix to be used as the first qualifier of all non-fully-qualified data set names, that prefix is added to the data set name.

Tips:

- A plus sign (+) next to a field indicates that you can press the Prompt function key (F4) to open a window that lists the values that you can select for that field.
- For a list of compatible combinations of system type and file type, see Chapter 81, “System and file types,” on page 629.

c) Press the Exit function key (F3).

The window closes.

The list at the bottom of the **Locate and Manage Log Files** panel shows the details of the file that you have just specified.

```
  File  Help

                       Locate and Manage Log Files      Row 1 of 1 More: < >
Command ===> ____________________________________________ Scroll ===> PAGE

NEW  Insert a new log file.
AUTO Run automated file selection to locate log files.

Log Files:
                                                        System      File
/   Exc Data Set Name                                  Name     Type    Type
___     CPPX.DEMO.SMF                                  MVSA     IMAGE   SMF
******************************** Bottom of data ********************************
```

Figure 22. Panel: Manually specifying an SMF file for a session (2 of 2)

9. Press the the Exit function key (F3) to return to the session menu.
10. On the session menu, select option 5 **Investigate**.

The **Investigate** panel is displayed. Use this panel to select which log files you want to browse, and whether you want to browse entire files starting at their first record or a *time slice*: a specific time period across one or more log files. Time slicing improves performance when browsing large log files.

The **Time Slice** heading shows the start and duration of the time slice, and whether time slicing is on or off.

The **Coverage** column indicates how much of the time slice each file covers.

11. In the **Duration** field under the **Time Slice** heading, enter `00.10.00` (10 minutes).
12. If the **Time Slice** heading shows **(OFF)**, enter `SLICE` on the command line to switch time slicing on.
13. Enter S on the first line under the **/** heading.

    Entering S in this top line shows a merged view of all log files for the session. In this example, we have only one file, so there is no difference between entering S on this line, or the line next to our single file.

    Tip: To set the time slice to the start and duration of a particular log file, enter T next to the file.

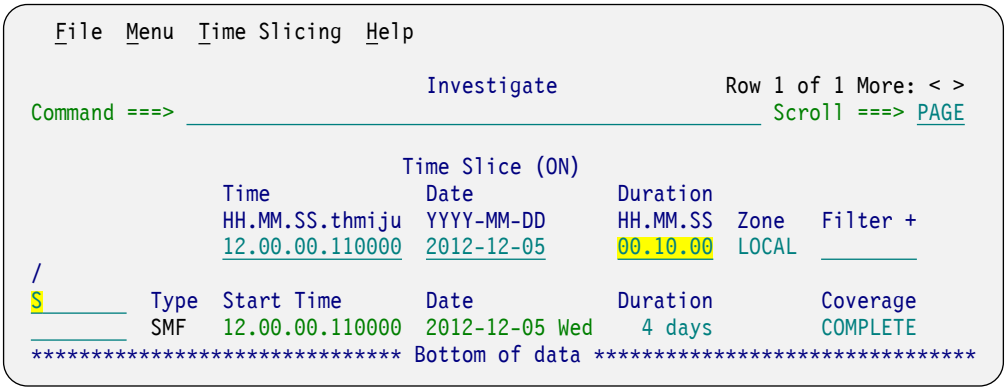


*Figure 23. Panel: **Investigate (Time Slicing)***

The log records are displayed in the log browser. Your screen should look similar to the following figure, although the details will depend on your particular log file.

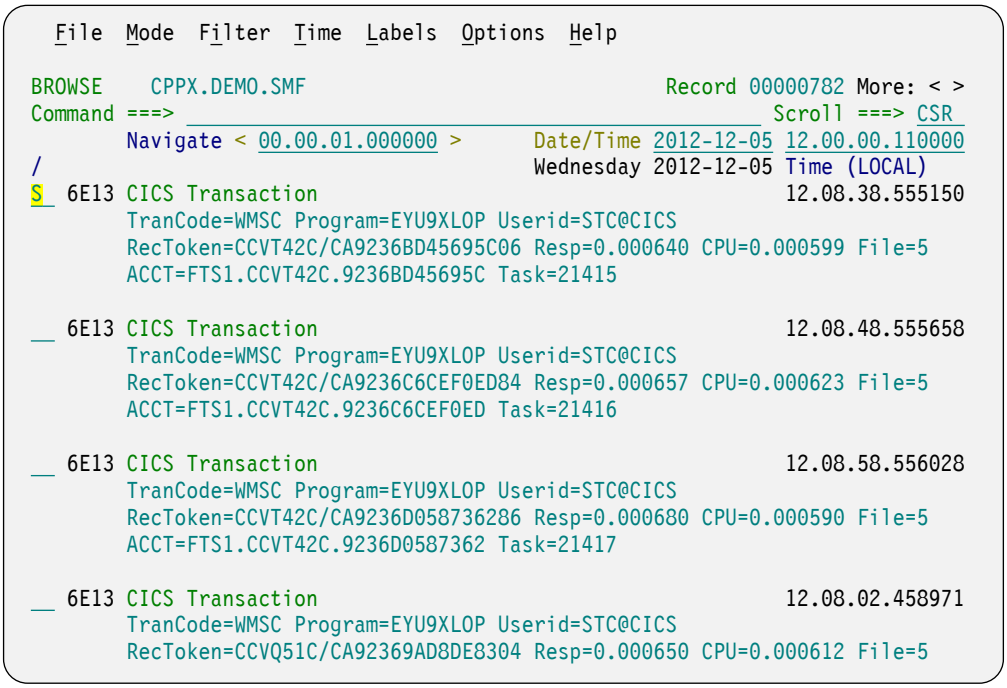


Figure 24. Panel: Browsing an SMF file

14. Browse the log.

- To scroll forward or backward through the records, press the Forward function key (F8) or the Backward function key (F7).
- To cycle through the four views on this log browser panel, press the Left function key (F10) or the Right function key (F11). The different views show or hide global fields, and show either the **Time** column or the **LSN** (log record sequence number) column.
- To view all the fields of a particular record, enter S next to the record.

Related tasks

Browsing logs interactively
You can use the Transaction Analysis Workbench ISPF dialog to browse records from different logs merged in a single view. You can select a record to browse its fields, and then select a field to browse a detailed description of its contents.

Adding log files to a session
If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Tutorial: Submitting a batch report

This tutorial shows you how to create a batch report for a session.

You must already have created a session and added an SMF file to the session. For details, see "Tutorial: Creating a session and browsing a log" on page 71.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**.
2. Select the session that you created previously.

```
  File  Help

                              Session Manager                 Row 1 to 2 of 2
Command ===> ____________________________________________ Scroll ===> PAGE

NEW Register a new Session

/  Key       Status  Description
_  00000001  OPEN    CPU delay in DBCTL
s  00000002  OPEN    Getting started
****************************** Bottom of data ********************************
```

*Figure 25. Panel: **Session Manager**: selecting an existing session*

3. On the session menu, select option 4 **Reporting**.

   The **Reporting** menu is displayed.
4. Select option 4 **SMF**.

   This option uses SMF files for input.

```
  File  Help

                                 Reporting
Option ===> 4 ____________________________________________________________

Select a reporting option then press Enter.

1  IMS         Transaction and system analysis using IMS PA
2  CICS        Transaction and system analysis using CICS PA
3  CICS-DBCTL  Combined CICS and IMS analysis of transactions
4  SMF         z/OS and subsystem analysis
5  DB2         DB2 accounting exception analysis
6  OPERLOG     Sysplex operations log (SYSLOG)
```

*Figure 26. Panel: **Reporting** menu*

5. On the **Reporting - z/OS and Subsystem Analysis** panel, select the **Address space accounting** report by entering a slash (/) next to it. Do not press Enter yet.

   For this tutorial, we will leave the **Report Interval** blank to process all of the records in the file that we are about to select, regardless of date.

6. Select the SMF file:

   a) Select option 2 **SMF File**.

   b) Tab to the **SMF File** field.

   c) Press the Prompt function key (F4).

      A pop-up window opens, which lists the SMF files that are associated with the session.

   d) Tab to an SMF file, and then press Enter.

      The window closes. The name of the SMF file that you selected is displayed in the **SMF File** field.

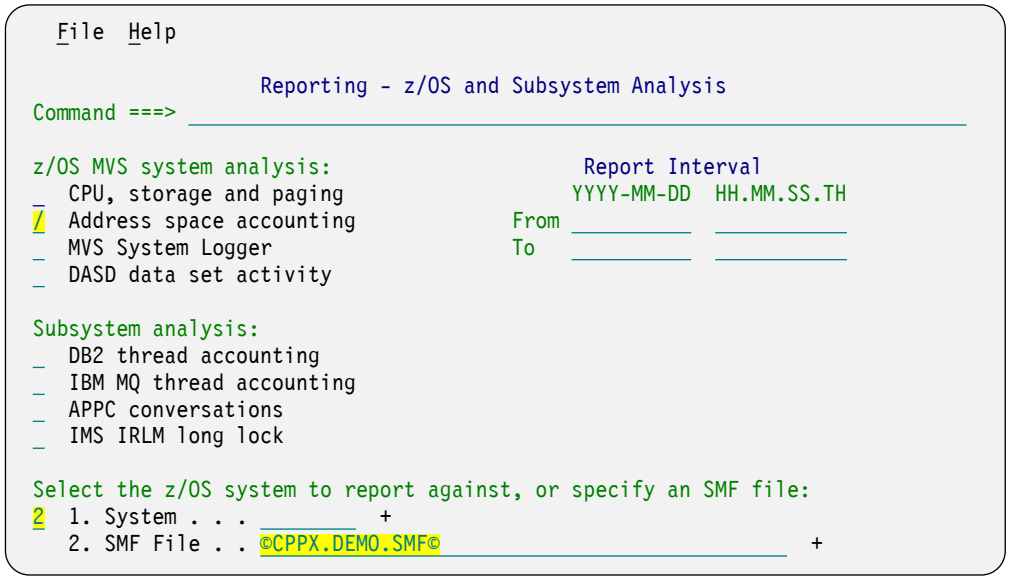


Figure 27. Panel: Selecting a report for an SMF file

7. Press Enter twice to generate the JCL for the report.

   The JCL is displayed in a **Notepad** panel.

   Tips:

   - Look at the **REPORT** command in the in-stream SYSIN data set. The **OUTPUT** parameter specifies the ddname of the data set in the job output that will contain the report.
   - To generate the STEPLIB DD statement, the dialog uses the value of the **Workbench Load Library** field from your dialog profile, option 0.1 **Workbench Personal Settings**. If the **Workbench Load Library** field is blank, the dialog omits the STEPLIB DD statement, and the FUWBATCH program must be in the standard z/OS search order, such as the LNKLST concatenation; otherwise, the job will fail.

     If the STEPLIB DD statement is missing, and you want to generate new JCL with this statement:

     a. Press the Cancel function key (F12) to exit the **Notepad** panel.

     b. Enter PROFILE on the command line of the **Reporting - z/OS and Subsystem Analysis** panel.

        The **Workbench Profile Settings** menu opens.

     c. Select option 1 **Personal**.

     d. In the **Workbench Load Library** field, enter the fully qualified data set name of the Transaction Analysis Workbench load library (supplied with the default low-level qualifier SFUWLINK), enclosed in single quotes. For example, 'FUW.V1R3M0.SFUWLINK'.

     e. Press the Exit function key (F3) to exit the **Workbench Personal Settings** panel.

f. Press the Exit function key (F3) to close the **Workbench Profile Settings** menu and return to the **Reporting - z/OS and Subsystem Analysis** panel.

g. Press Enter twice to generate the JCL.

8. Submit the job: enter **SUB** on the command line.

```
  File  Edit  Edit_Settings  Help

EDIT                            Notepad                        New task created
Command ===> SUB                                               Scroll ===> PAGE

Description z/OS and subsystem analysis report

****** ***************************** Top of Data ******************************
000001 //MIDFUW JOB ,NOTIFY=&SYSUID
000002 //*
000003 //FUWBATCH EXEC PGM=FUWBATCH
000004 //STEPLIB  DD  DSN=PRODUCTS.FUW.SFUWLINK,
000005 //             DISP=SHR
000006 //FUWPROBR DD  DSN=MID.FUW.SESSIONS,
000007 //             DISP=SHR
000008 //SYSPRINT DD  SYSOUT=*                   Messages
000009 //JOBSTATS DD  SYSOUT=*                   Address Space accounting
000010 //SMFIN001 DD  DSN=CPPX.DEMO.SMF,
000011 //             DISP=SHR
000012 //SYSIN    DD  *                          Command input
000013 SESSION=00000002
000014 REPORT SMF(30)     OUTPUT(JOBSTATS)     /* Address Space accounting
000015 CODE(SMF,30.)                          /* Include...
000016 COND PROGRAM EQ ©DFH*©                 /* ...CICS address spaces
000017 COND PROGRAM EQ ©DFS*©                 /* ...IMS address spaces
000018 COND PROGRAM EQ ©BPE*©                 /* ...IMS address spaces
000019 COND PROGRAM EQ ©DSNYASCP©             /* ...DB2 address spaces
000020 COND PROGRAM EQ ©DXRRLM00©             /* ...IRLM address spaces
000021 /*
```

Figure 28. Panel: Submitting generated JCL

9. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Example report

The following figure shows an example report.

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                      SMF Type=30 Address Space Activity

------------Interval-------------   System                              ----------- CPU -----------  EXCPs  ----Storage-----
Start Date/Time      Duration Typ    Name   Jobname  Stepname Comp        TCB        SRB  %CPU  /Sec   <16M  >16M 64bit

2011-02-25 09:45:00  00:14:59 INT  FTS1     DB2PMSTR MASTER   0000   1.188172   0.284325   0.2      1    1M   25M    0M
2011-02-25 09:45:00  00:14:59 INT  FTS1     DB2PDBM1 DBM      0000   0.059287   0.697935   0.1      5    2M  150M    0M

2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPMPP1 REGION   0000   0.110287   0.003570   0.0      0    0M   11M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPCTL  CONTROL  0000   0.300900   0.068595   0.0      0    2M   21M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPDBRC DBRC     0000          0   0.000255   0.0      0    0M   15M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPDLIS DLISAS   0000   0.028560   0.001147   0.0      0    1M   15M    0M

2011-02-25 11:30:01  00:14:58 INT  FTS1     CICSPAOR CICS     0000  97.986300   1.432845  11.1     21    4M 1369M    0M
⋮
```

Figure 29. Example Address Space Activity report

Related tasks

Creating reports

You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench

report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Related reference

SMF type 30: Address Space Activity report
The Address Space Activity report uses data from SMF type 30 records to show activity in each address space per interval, where the interval depends on the record subtype.

Tutorial: Automating selection of IMS logs

This tutorial shows you how to define an IMS system to Transaction Analysis Workbench, and then use that system definition to locate the related IMS log files (SLDS or OLDS) for a particular time interval.

- The Transaction Analysis Workbench automated file selection utility uses the IMS database recovery (DBRC) API to read RECON data sets. IMS (or at least, the IMS RESLIB library, containing the DBRC API code) must be installed on the system where you want to run the utility.
- Determine the time interval (the "from" and "to" dates and times) of the IMS logs that you want to analyze.
- Obtain the following details about the IMS system that created the logs. If you already have an IMS system definition that specifies these details (for example, your IMS system definition repository is an existing IMS Performance Analyzer profile library), skip to step “7” on page 81.
  - IMS subsystem identifier (IMSID)
  - IMS version
  - RESLIB data set name
  - Either of the following details:
    - Two or three IMS RECON data set names
    - Data set name of an MVS dynamic allocation (MDA) library (containing RECON*n* members that refer to the RECON data set names)

  If you do not know these details, contact your IMS system administrator.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 3 **Systems**.

   The **System Definitions Menu** is displayed.
2. Check the name of the data set for the IMS system definition repository.

   This is where Transaction Analysis Workbench will store the system definition that you are about to create. For this tutorial, you might choose to use a private data set rather than a data set that is shared with others.
3. Select **IMS**.

   If the repository does not yet exist, a **Define Repository Data Set** panel is displayed; press Enter to create the repository.

```
  File  Help

                        System Definitions Menu
Command ===> ____________________________________________________________

Select the type of system then press Enter.

   Systems:            Groups of Systems:
1  1. IMS              6. IMS systems (IMSplexes)
   2. CICS             7. CICS, DB2 and related
   3. DB2
   4. MQ
   5. MVS Image

System definition repositories:
  IMS  . . . . . . . . FUW.ISYSDEFS                                  +
  CICS, DB2, more  . . FUW.CSYSDEFS                                  +
```

*Figure 30. Panel: **System Definitions Menu***

The **System Definitions** panel is displayed. This panel lists any existing IMS and IMS Connect system definitions in the repository.

4. Enter **NEW** on the command line.

```
  File  Menu  Edit  Help

                              System Definitions          Row 1 of 1 More: < >
Command ===> new                                          Scroll ===> PAGE

Specify IMS and Connect systems.

/ System   Type     VRM +              Description
_ IADJ     IMS      131   IMS demonstration system
****************************** Bottom of data ********************************
```

*Figure 31. Panel: **System Definitions** (IMS and IMS Connect)*

The **IMS Subsystem** panel is displayed.

5. Enter the details that you have obtained for the IMS system:
   a) In the fields at the top of the panel, specify the following details:
      - IMS subsystem ID
      - IMS version
      - A description of the IMS subsystem (this description is used only for selection lists in the dialog)
      - RESLIB data set name
   b) Select the **DBRC Settings** view.
   c) Specify either of the following details:
      - Two or three IMS RECON data set names
      - Data set name of an MVS dynamic allocation (MDA) library (containing RECON*n* members that refer to the RECON data set names)

      Note:

      - If the RECON data sets that you want to use belong to a running IMSplex:
        – Specify the IMSplex name.
        – The structured call interface (SCI) address space must be running on the system on which you want to run the automated file selection utility.

- If the RECON data sets are used by more than one IMSplex, also specify the DBRC sharing group ID.

This step assumes that you want to use cataloged, primary system log data sets (SLDS). If the SLDS are uncataloged, if you want to use secondary SLDS, or if you want to use online log data sets (OLDS) instead of SLDS, see the online help by pressing the Help function key (F1) on the related options.

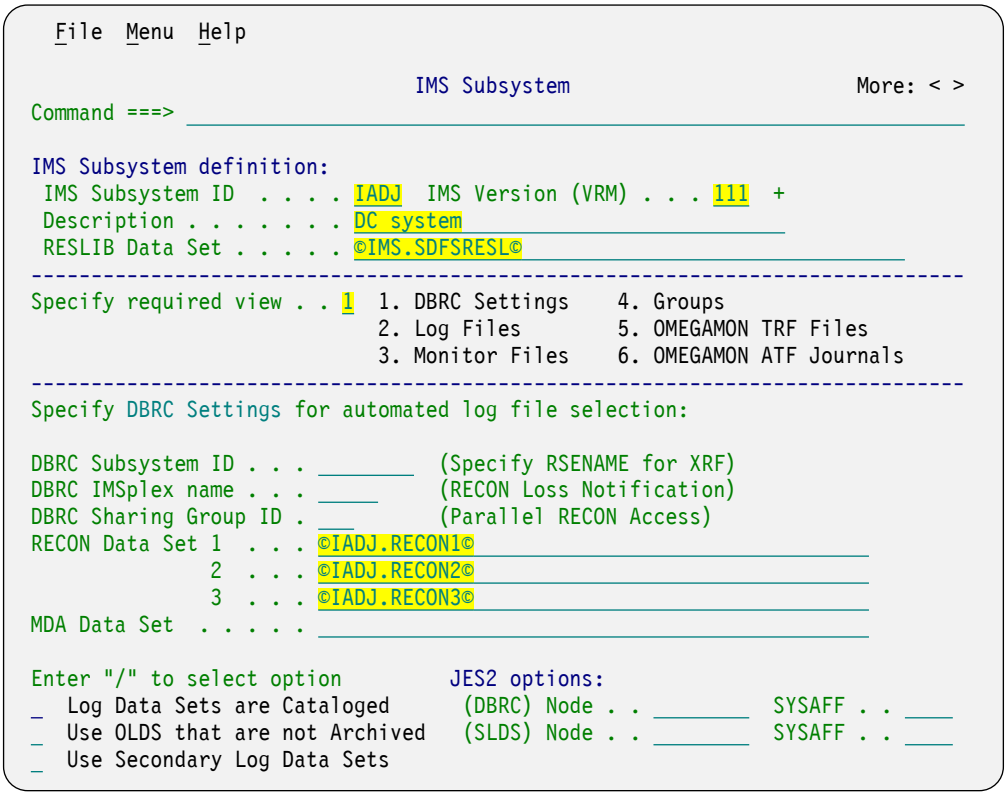


*Figure 32. Panel: **IMS Subsystem**: definition with DBRC settings for automated file selection*

6. Press the Exit function key (F3) to exit the **IMS Subsystem** panel. Press the Exit function key (F3) repeatedly until you return to the Transaction Analysis Workbench Primary Option Menu.

   The following steps use the IMS system definition that you have just created to automate selection of the related log files for a particular time interval, and then add those selected files to a problem session.
7. On the primary option menu, select option 1 **Sessions**.
8. On the command line of the **Session Manager** panel, enter **NEW** to create a new session.
9. On the **Problem Details** panel for the new session, specify the following details:
   - A summary (for example, `IMS log selection`)
   - The from and to dates and times of the period when the problem occurred
   - The IMS system that you have just defined

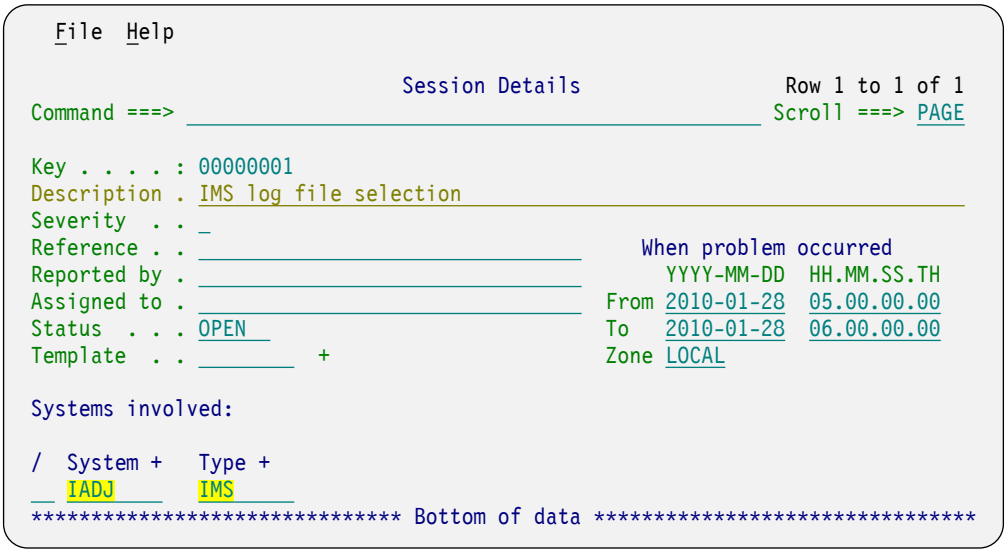


*Figure 33. Panel: **Session Details**: session that refers to an IMS system*

10. Press the Exit function key (F3) to save the new session.

    The session menu is displayed.
11. Select option 3 **Files**.
12. On the **Locate and Manage Log Files** panel, run automated file selection:
    a) Enter **AUTO** on the command line.

       The **Automated File Selection** window opens.
    b) Specify the IMS system that you defined previously.

       By default, automated file selection spans the same date and time interval as the session; however, you can override these default values.

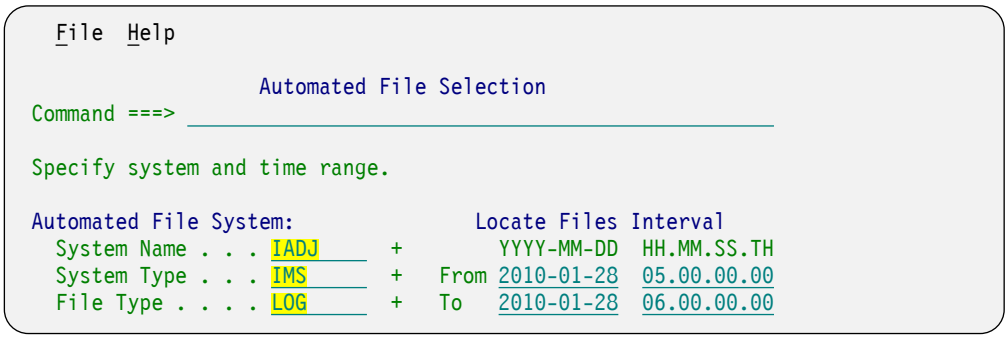


Figure 34. Panel: Automated IMS log file selection

    c) Press Enter to generate the JCL for the automated file selection.

       The JCL is displayed in a **Notepad** (ISPF edit) panel.
    d) Submit the job: enter **SUB** on the command line.
13. Press the Exit function key (F3) to exit the **Notepad** panel and return to the **Locate and Manage Log Files** panel.
14. When the job is complete, enter **REFRESH** on the command line to refresh the list of log files and to see any files added by the job.

    You can now browse these log files, or use them to generate batch reports or extracts.

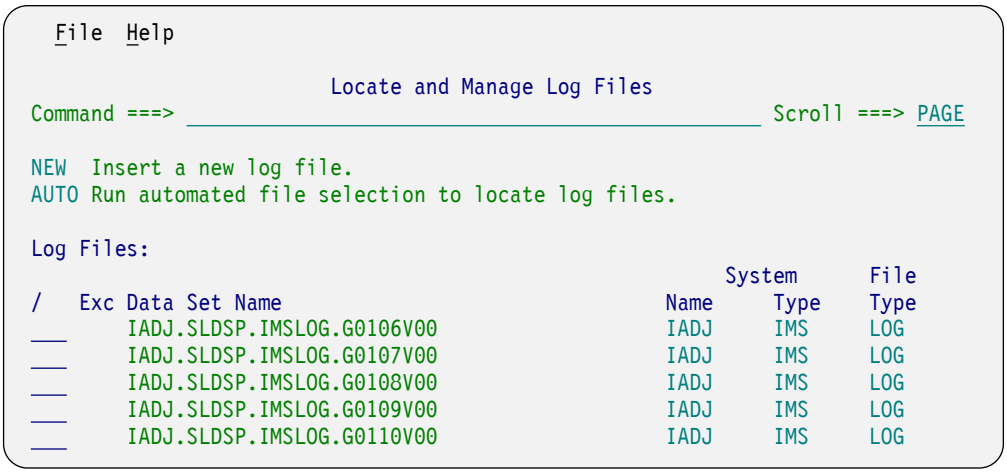


Figure 35. Panel: IMS log files located by automated file selection

Related concepts

Repositories

Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Related tasks

Adding log files to a session

If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Defining systems for log file selection

Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Related reference

JCL for automated IMS log file selection (using DBRC)

To select IMS log files, the Transaction Analysis Workbench automated file selection utility uses the IMS database recovery (DBRC) API to read RECON data sets.

Tutorial: Automating selection of DB2 logs

This tutorial shows you how to define a DB2 system to Transaction Analysis Workbench, and then use that system definition to locate the related DB2 log files for a particular time interval.

- To locate DB2 log files, the Transaction Analysis Workbench automated file selection utility uses output from the DB2-supplied print log map utility, DSNJU004. DB2 (in particular, DSNJU004) must be installed on the system where you want to run the Transaction Analysis Workbench automated file selection utility.
- Determine the time interval (the "from" and "to" dates and times) of the DB2 logs that you want to analyze.
- Obtain the following details about the DB2 system that created the logs. If you already have a DB2 system definition that specifies these details (for example, your DB2 system definition repository is an existing CICS Performance Analyzer HDB register that contains shared system definitions), skip to step “8” on page 85.
  - DB2 subsystem identifier (SSID)
  - Optionally, the SYSID of the MVS image on which the DB2 system runs
  - DB2 version

– Whether the DB2 system is in a data sharing group (yes or no)
– DSNLOAD library data set name
– DB2 bootstrap data set (BSDS) name

If you do not know these details, contact your DB2 system administrator.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 3 **Systems**.

   The **System Definitions Menu** is displayed.
2. Check the name of the data set for the "CICS, DB2, more" system definition repository.

   This is where Transaction Analysis Workbench will store the system definition that you are about to create. For this tutorial, you might choose to use a private data set rather than a data set that is shared with others.
3. Select **DB2**.

   If the repository does not yet exist, a **Define Repository Data Set** panel is displayed; press Enter to create the repository.

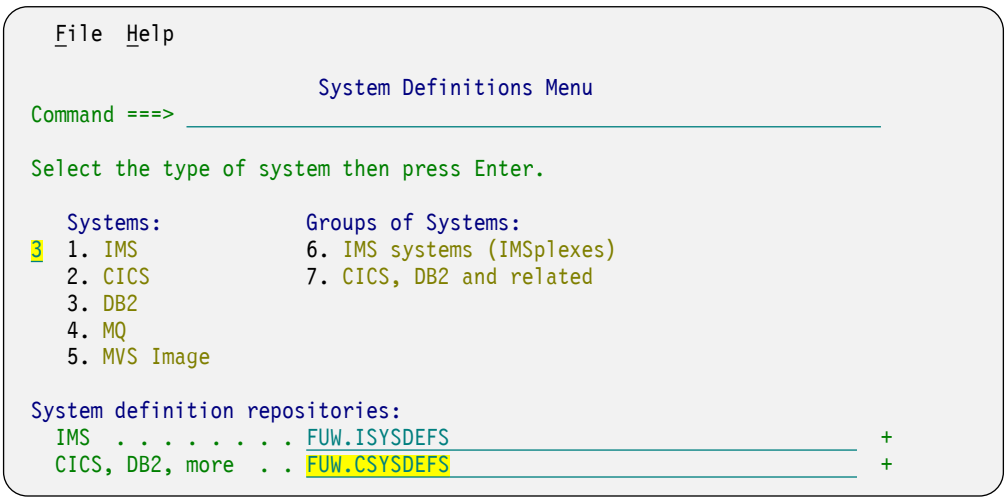


*Figure 36. Panel: **System Definitions Menu***

   The **System Definitions** panel is displayed. This panel lists any existing system definitions in the repository.
4. Enter **NEW** on the command line.

   A **DB2 Subsystem** panel for the new system is displayed.
5. Enter the DB2 subsystem ID (SSID) and description.
6. Specify the remaining details that you have obtained for the DB2 system:
   a) Select the **Definition** system view (this is the default view, so it might already be selected).
   b) Specify the following details:
      - Whether the DB2 system is in a data sharing group (yes or no)
      - DSNLOAD library data set name
      - DB2 bootstrap data set (BSDS) name

```
  File  Help

EDIT                              DB2 Subsystem                        More: < >
Command ===> ____________________________________________________________________

DB2 System definition:
 DB2 SSID  . . . . . . . DB2P  MVS Image . . . ________
 Description . . . . . . Production DB2 ____________________

System View:
 1  1. Definition   2. Cyclic SMF Files

Specify DB2 Subsystem Definition:
 Data sharing  . . . . . NO_  (YES or NO)
 DSNLOAD library . . . . ©DB2.PROD.SDSNLOAD© ____________________
 DB2 bootstrap . . . . . ©DB2P.BSDS01© ____________________

Data sources
 SMF log stream  . . . . ____________________________________  RETPD _____
 Near-term history . . . ____________________________________
```

*Figure 37. Panel: **DB2 Subsystem** definition*

7. Press the Exit function key (F3) to save the DB2 system definition. Press the Exit function key (F3) repeatedly until you return to the Transaction Analysis Workbench Primary Option Menu.

   The following steps use the DB2 system definition that you have just created to automate selection of the related log files for a particular time interval, and then add those selected files to a problem session.
8. On the primary option menu, select option 1 **Sessions**.
9. On the command line of the **Session Manager** panel, enter **NEW** to create a new session.
10. On the **Problem Details** panel for the new session, specify the following details:
    - A summary (for example, `DB2 log selection`)
    - The from and to dates and times of the log records that you want to analyze
    - The DB2 system that you have just defined

```
  File  Help

                              Session Details                 Row 1 to 1 of 1
Command ===> ____________________________________________ Scroll ===> PAGE

Key . . . . : 00000012
Description . DB2 log selection ____________________________________________
Severity  . . _
Reference . . ____________________________     When problem occurred
Reported by . ____________________________       YYYY-MM-DD  HH.MM.SS.TH
Assigned to . ____________________________  From 2010-06-24  15.20.00.00
Status  . . . OPEN__                        To   2010-06-24  16.50.00.00
Template  . . ________  +                   Zone LOCAL

Systems involved:

/  System +   Type +
__ DB2P____   DB2 ____
******************************** Bottom of data ********************************
```

*Figure 38. Panel: **Session Details**: session that refers to a DB2 system*

11. Press the Exit function key (F3) to save the new session.

    The session menu is displayed.
12. Select option 3 **Files**.
13. On the **Locate and Manage Log Files** panel, run automated file selection:

a) Enter **AUTO** on the command line.

The **Automated File Selection** window opens.

b) Specify the DB2 system that you defined previously.

By default, automated file selection spans the same date and time interval as the problem session; however, you can override these default values.

```
  File  Help

                  Automated File Selection
Command ===> ____________________________________________

Specify system and time range.

Automated File System:              Locate Files Interval
  System Name . . . DB2P     +         YYYY-MM-DD  HH.MM.SS.TH
  System Type . . . DB2      +    From 2010-06-24  15.20.00.00
  File Type . . . . LOG      +    To   2010-06-24  16.50.00.00
```

Figure 39. Panel: Automated DB2 log file selection

c) Press Enter to generate the JCL for the automated file selection.

The JCL is displayed in a **Notepad** (ISPF edit) panel.

d) Submit the job: enter **SUB** on the command line.

14. Press the Exit function key (F3) to exit the **Notepad** panel and return to the **Locate and Manage Log Files** panel.
15. When the job is complete, enter **REFRESH** on the command line to refresh the list of log files and to see any files added by the job.

```
  File  Help

                        Locate and Manage Log Files          Row 1 to 3 of 3
Command ===> ____________________________________________ Scroll ===> PAGE

NEW  Insert a new log file.
AUTO Run automated file selection to locate log files.

Log Files:
                                                        System      File
/   Exc Data Set Name                                  Name     Type    Type
___     DB2P.ARCHLOG1.A0003778                         DB2P     DB2     LOG
___     DB2P.ARCHLOG1.A0003780                         DB2P     DB2     LOG
___     DB2P.ARCHLOG1.A0003779                         DB2P     DB2     LOG
****************************** Bottom of data ********************************
```

Figure 40. Panel: DB2 log files located by automated file selection

Related concepts

Repositories

Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Related tasks

Adding log files to a session

If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Defining systems for log file selection

Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for

documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Related reference

JCL for automated DB2 log file selection
To select DB2 log files, the Transaction Analysis Workbench automated file selection utility uses output from the DB2-supplied print log map utility, DSNJU004.

Tutorial: Using the plug-in to create a session and perform a workflow

This tutorial shows you how to use the Transaction Analysis Workbench Eclipse plug-in to create a session based on a template, and then perform the workflow tasks inherited from the template. The tutorial also covers the prerequisite steps that need to be performed in the ISPF dialog by a subject-matter expert: defining systems, and then creating a session template that contains workflow tasks for those systems.

You can use the plug-in to create a session that is not based on a template. However, such a session has no workflow tasks, and so has limited use in the plug-in. At best, such a session is a placeholder containing a few details about the problem for further analysis in the ISPF dialog.

To fully exploit the plug-in, you need to base new sessions on templates. A session template refers to the systems involved in a particular type of problem and contains workflow tasks for those systems. For example, a session template for a problem with a CICS-DB2 transaction refers to the corresponding CICS system, DB2 system, and host z/OS system; the template contains workflow tasks to select the log files for those systems, such as SMF files, and perhaps also to create extracts of those original log files, and to create batch reports from the extracts. You can use the plug-in to perform workflow tasks. So, rather than simply creating a session and entering a few descriptive details about the problem for further attention by a subject-matter expert, plug-in users, who might have no knowledge of the systems involved, can perform useful preliminary analysis steps before assigning the problem to an expert.

This tutorial consists of steps to be performed in the ISPF dialog by a subject-matter expert followed by steps to be performed by the plug-in user. The plug-in user might be a help desk team member with no knowledge of the systems involved, or the same expert who created the session template. Here is a summary of the steps:

- Steps for the expert, using the ISPF dialog:
  1. Create system definitions that instruct Transaction Analysis Workbench how to locate log files.
  2. Create a session template; a blueprint for starting problem analysis.
- Steps for the plug-in user:
  1. Register the reported problem in a new session, referring to the template created by the expert
  2. Schedule (run) the workflow tasks inherited from the template

1. On the Transaction Analysis Workbench ISPF dialog primary option menu, select option 3 **Systems**.

   The **System Definitions Menu** is displayed.

   In this tutorial we are going to create a session for a CICS-DB2 transaction problem, so we will define the following systems:

   - MVS image; the z/OS system running the CICS and DB2 systems
   - CICS system
   - DB2 system
2. Define the MVS image.
   a) Select **MVS Image**.

      The **System Definitions** panel is displayed.

   b) Enter **NEW** on the command line.

      An **MVS Image** panel for the new system is displayed.

c) Enter a system name and description. For the system name, consider using the SMF ID; for example, MVS1.

```
EDIT                              MVS Image                          More: < >
Command ===> ____________________________________________________________________

MVS Image System definition:
 MVS Image . . . . MVS1____
 Description . . . Production MVS image_____________

System View:
 1  1. Definition   2. Cyclic SMF Files

Log Streams:
 OPERLOG . . . _____________________________
 SMF . . . . . _____________________________  RETPD _____
```

Figure 41. Panel: Defining an MVS image: system name and description

d) Press the Right function key (F11) or select **System View** option 2 **Cyclic SMF Files**.

The fields for cyclic SMF files scroll into view.

Each site has its own policy for the retention and accessibility of SMF data. In this example, we will specify a generation data group (GDG) base for weekly SMF data. Each new generation starts on Sunday (the **Origin**), contains one week of data (the **Interval**), and is accumulated over the course of the week (**DISP**=MOD).

e) Enter the cyclic SMF file details.

```
EDIT                              MVS Image                Row 1 of 1 More: < >
Command ===> ____________________________________________ Scroll ===> PAGE

MVS Image System definition:
 MVS Image . . . . MVS1____
 Description . . . Production MVS image_____________

System View:
 2  1. Definition   2. Cyclic SMF Files

/ Exc Cyclic SMF File GDG Base or Data Set Name       Origin     Interval DISP
_     ©MVS1.SMF.WEEKLY©_____________________________ SUNDAY___ WEEK     MOD
****************************** Bottom of data ********************************
```

Figure 42. Panel: Defining an MVS image: cyclic SMF files

f) Press the Exit function key (F3) to exit the **MVS Image** panel and save the definition.

3. Press the Exit function key (F3) to return to the **System Definitions Menu**.
4. Define the CICS system.
   a) Select **CICS**.
   b) Enter **NEW** on the command line.

   A **CICS System** panel for the new system is displayed.

   c) Enter the CICS APPLID and description.
   d) Enter the name of the MVS image that we recently defined.

   The CICS system writes to the same SMF files as its parent MVS image. Rather than specifying the SMF file details again in this system definition, you can refer to the parent MVS image. When selecting SMF files for this CICS system, Transaction Analysis Workbench uses the details specified for the MVS image.

   Leave all other fields blank.

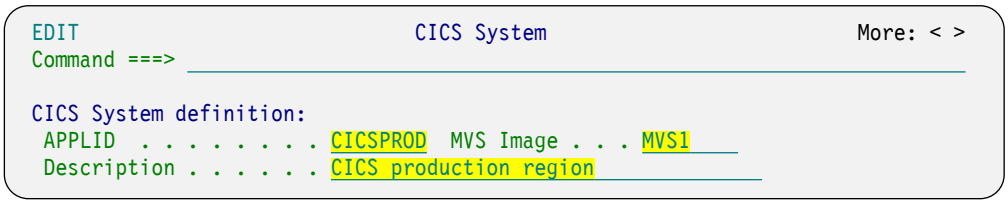


Figure 43. Panel: Defining a CICS system: referring to the parent MVS image

e) Press the Exit function key (F3) to exit the **CICS System** panel and save the definition.

5. Press the Exit function key (F3) to return to the **System Definitions Menu**.
6. Define the DB2 system.
   a) Select **DB2**.
   b) Enter **NEW** on the command line.

      A **DB2 Subsystem** panel for the new system is displayed.
   c) Enter the DB2 subsystem ID (SSID) and description.
   d) Enter the name of the MVS image that we recently defined.

      In addition to SMF file details, DB2 system definitions can also contain details for automating the selection of DB2 log files.
   e) Enter values in the **Data sharing**, **DSNLOAD library**, and **DB2 bootstrap** fields.

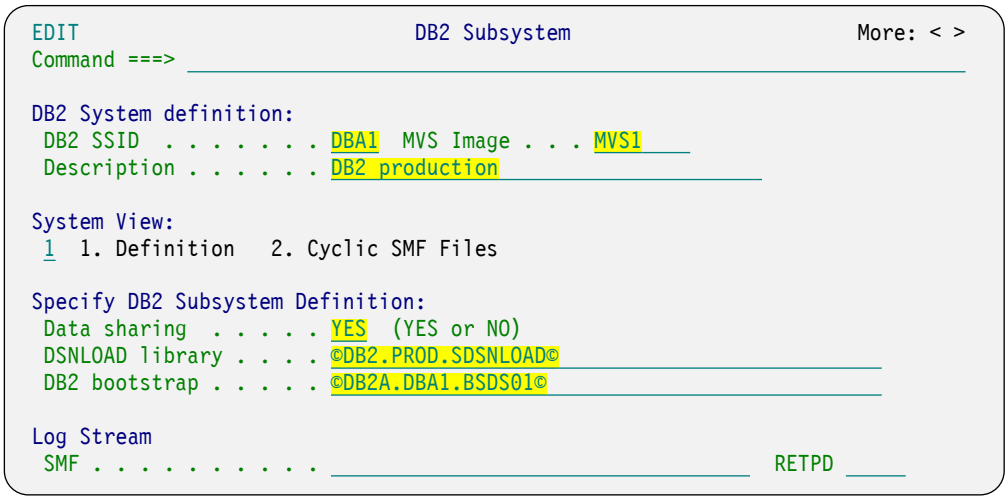


Figure 44. Panel: Defining a DB2 system

   f) Press the Exit function key (F3) to exit the **DB2 Subsystem** panel and save the definition.

      We have created three system definitions: an MVS image, a CICS system, and a DB2 system. Next, we will create a session template that contains workflow tasks to automate the selection of log files for those systems, and create extracts and transaction indexes from those log files.
7. Return to the Transaction Analysis Workbench Primary Option Menu.
8. Select option 2 **Controls**.

   The **Controls Menu** is displayed.
9. Select option 3 **Session Templates**.

   The **Template Manager** panel is displayed.
10. Enter **NEW** on the command line.

    The **New Session Template** window opens.
11. Enter a name for the session template and then press Enter.

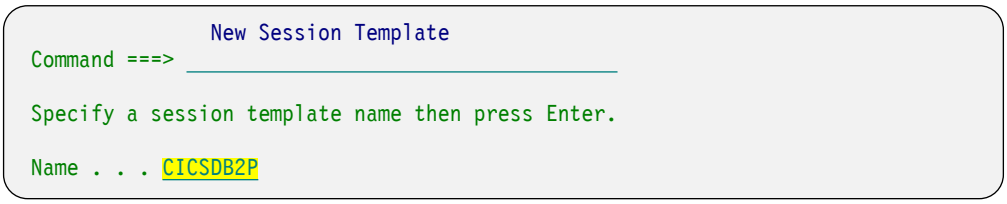


Figure 45. Panel: Creating a session template: specifying the new template name

The **Template Details** panel for the new template is displayed.

12. Enter a description for the template.
13. Add the systems that we defined earlier.
    a) Tab to the first field under the **System** column heading.
    b) Press the Prompt function key (F4).
    c) Enter S next to the MVS image.
    d) Add the CICS and DB2 systems.

    Tip: To insert a new blank item in the list of systems, enter line action `I` under the **/** (slash) column heading.

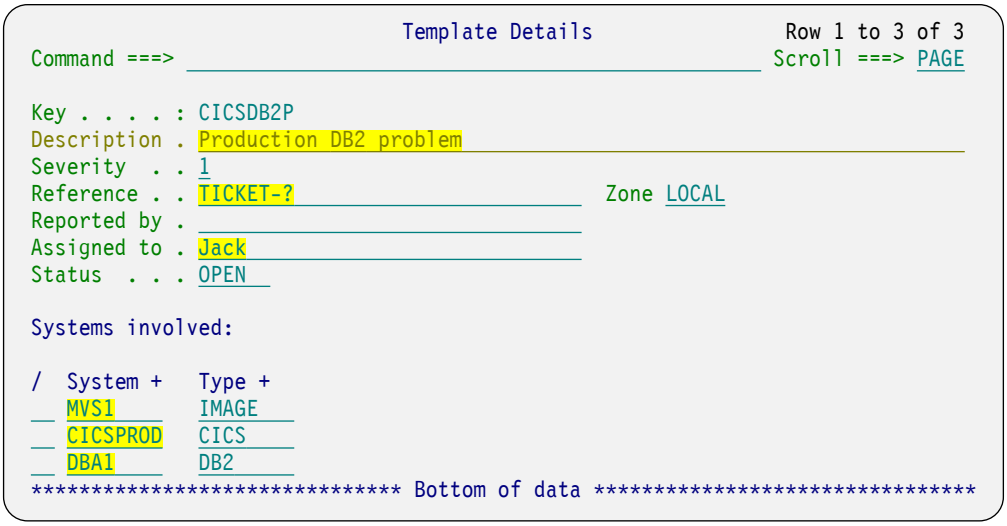


Figure 46. Panel: Creating a session template: specifying the description, the systems involved, and other optional details

Consider specifying other optional details. For example:

- If your organization has a problem tracking system, consider entering a placeholder value for the problem ID in the **Reference** field. In the following figure, we have specified the placeholder value `TICKET-?`.

  When users create a session based on this template, the **Reference** field in the new session inherits the value from the template, acting as a prompt to the user to replace the value with an actual problem ID from your problem tracking system.
- If you know the subject-matter expert at your organization for the particular combination of systems involved, consider referring to that user in the **Assigned to** field.

  When users create a session based on this template, the **Assigned to** field in the new session inherits the value from the template; by default, production CICS-DB2 problems are assigned to Jack.

14. Press the Exit function key (F3).

    Because you added systems to the template, the **Auto Task Creation** window opens, prompting you to create workflow tasks for those systems.

```
                          Auto Task Creation                Row 1 to 4 of 4
Command ===> ____________________________________________ Scroll ===> PAGE

Create workflow tasks to:
o   Locate the log files require for diagnosis
o   Create extracts of the original log files
o   Create transaction indexes (for CICS, DB2 and IMS)

Select the systems that you want to create tasks for then press EXIT

   ---- System ------  File
S  Name      Type      Type      Eligible?
.  MVS1      IMAGE     SMF       YES
.  DBA1      DB2       SMF       YES
.  DBA1      DB2       LOG       YES
.  CICSPROD  CICS      SMF       YES
****************************** Bottom of data ******************************
```

Figure 47. Panel: Creating a session template: auto task creation

The **Auto Task Creation** window lists the combinations of systems and file types for which Transaction Analysis Workbench can perform automated file selection. The **Eligible?** column indicates whether the system definitions contain the information required for automated file selection.

When we defined the MVS image, we specified cyclic SMF file details. When we defined the CICS and DB2 systems, we referred to that parent MVS image. So, all three systems are eligible for SMF file selection. In addition, when we defined the DB2 system, we specified the details required to select DB2 logs.

15. Select all systems: enter S next to the column headings, above the first system in the list.
16. Press the Exit function key (F3).

    Transaction Analysis Workbench creates the workflow tasks.

    The menu for the new template is displayed.

    The menu for a session template shows the same options as the menu for a session. However, some of the options are available only in sessions, not in templates; on the menu for a session template, these options are disabled.

```
                            Template CICSDB2P
Option ===> 2 ____________________________________________________________

Description : Production CICS-DB2 problem

1  Register      Update the problem registration details
2  Workflow      Perform the diagnostic tasks
3  Files         Locate and manage the log files required for diagnosis
4  Reporting     Run batch reports
5  Investigate   Perform interactive log file analysis
6  History       Review the problem history
```

Figure 48. Panel: Session template menu

Next, we will review the workflow tasks that we have just created.

17. Select option 2 **Workflow**.

    The **Tasks** panel is displayed, listing the workflow tasks.

```
                                    Tasks                        Row 1 to 8 of 8
Command ===> NEW NOTE                                           Scroll ===> PAGE

 NEW   Create a new task
 AUTO  Create file selection and extract tasks
 SCHED Schedule all the tasks (or select required tasks only)

/  Task Status      Description
__    1  NOT DONE    Select SMF files for IMAGE system MVS1
__    2  NOT DONE    Select SMF files for CICS system CICSPROD
__    3  NOT DONE    Select SMF files for DB2 system DBA1
__    4  NOT DONE    Select log files for DB2 system DBA1
__    5  NOT DONE    Create SMF extract for CICS system CICSPROD
__    6  NOT DONE    Create SMF extract for DB2 system DBA1
__    7  NOT DONE    Create log extract for DB2 system DBA1
__    8  NOT DONE    Create index for CICS systems
****************************** Bottom of data ********************************
```

Figure 49. Panel: Session workflow tasks

18. Enter S next to one of the tasks.

    The JCL for the task is displayed.

    Batch job task JCL in a session template is "unresolved" JCL: it contains references to substitution variable names that are prefixed by a percent sign (%) or a plus sign (+). Transaction Analysis Workbench resolves the JCL when the workflow task is performed in a session.

19. Press the Cancel function key (F12) to return to the **Tasks** panel.

    When you create a session template, consider adding a note as the final task in the workflow, to offer guidance to users about what to do next. This note might also include instructions for further analysis.

20. Enter `NEW NOTE` on the command line.

    A blank note is displayed.

21. Enter text for the note.

```
EDIT                            Notepad                   Columns 00001 00072
Command ===>                                                Scroll ===> PAGE
****** ***************************** Top of Data ****************************
000001 Assign the problem to Jack, our CICS-DB2 expert
****** **************************** Bottom of Data ***************************
```

Figure 50. Panel: Creating a note task

22. Press the Exit function key (F3) to save the note and return to the **Tasks** panel.

    The new note task is displayed at the bottom of the list of tasks.

23. Press the Exit function key (F3) until you return to the Transaction Analysis Workbench Primary Option Menu.

    We have now completed the prerequisite steps for using the plug-in to create a session based on a template.

24. Start the Eclipse environment where you have installed the plug-in.

25. Open the session repository where you saved the session template: in the **Navigation** view, expand **All Source** > **Analysis Sessions**, and then double-click the session repository.

    In this example, the session repository name is HELPDESK.

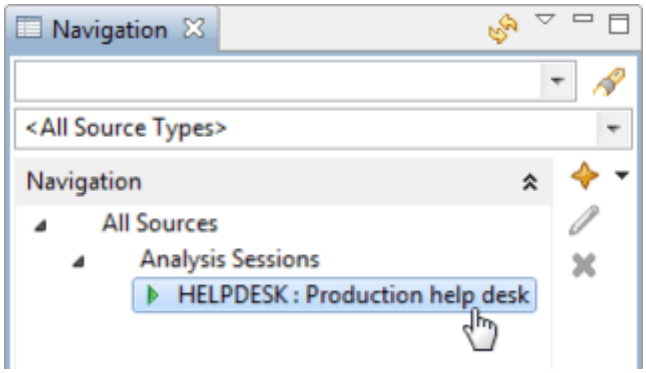


Figure 51. Opening a session repository in the plug-in

A **Sessions** tab opens, showing the list of sessions in the repository.

26. Click **New Session**.
27. Select the session template that we have just created.

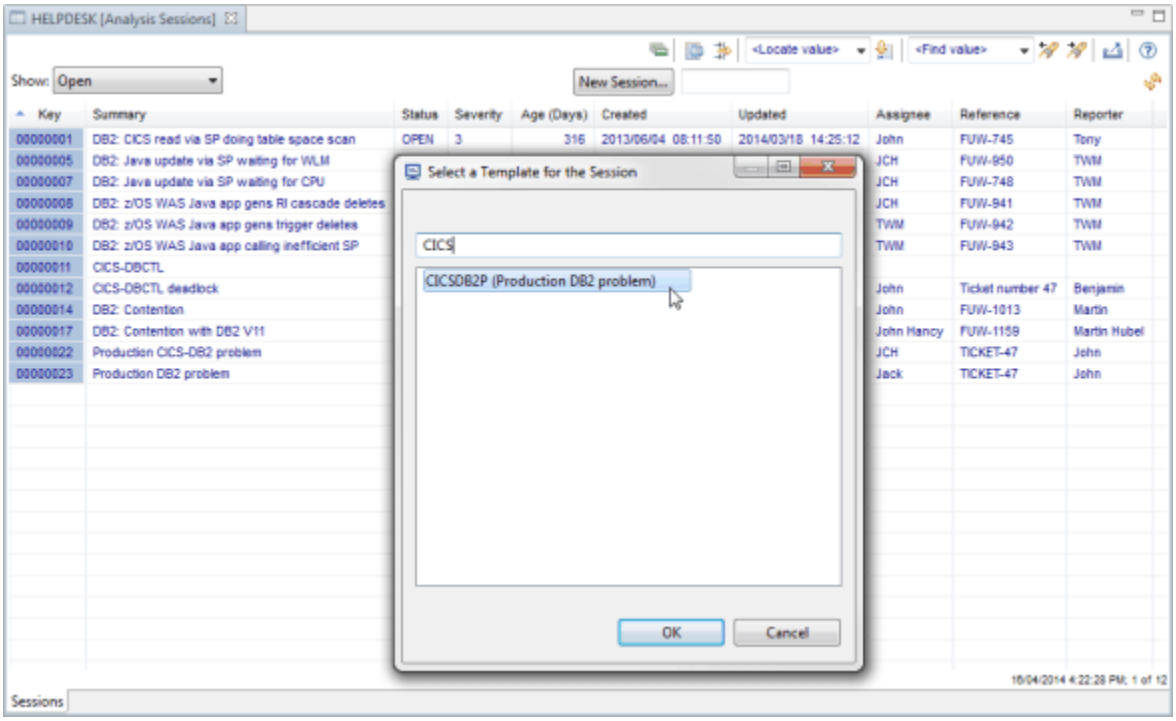
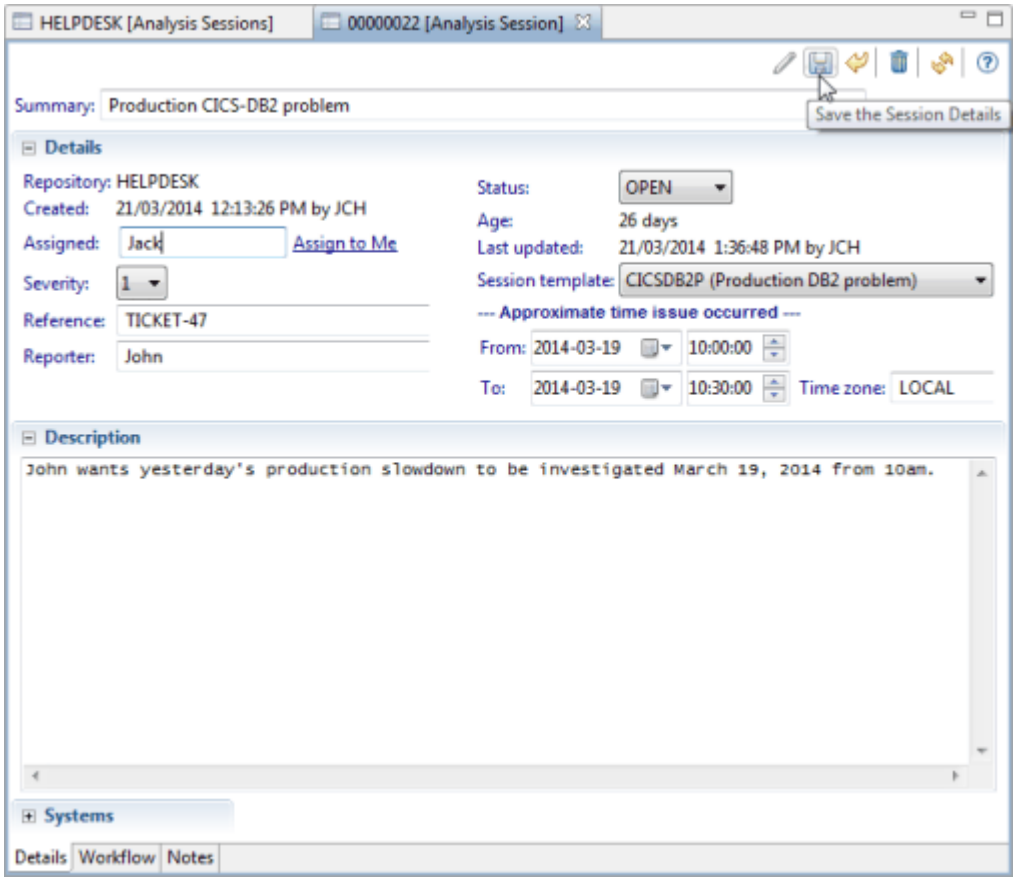


Figure 52. Using the plug-in to create a session: selecting a template

An **Analysis Session** editor opens for the new session.

28. Enter the session details, including the time that the problem occurred.



You must specify the interval of time when the problem occurred; otherwise, the workflow will fail. The automated file selection tasks requires this interval to select the corresponding log files.

Figure 53. Using the plug-in to create a session: specifying session details

29. Click the **Save** icon at the top of the **Details** tab.
30. Click the **Workflow** tab.
31. Click **Submit** > **All Tasks**.

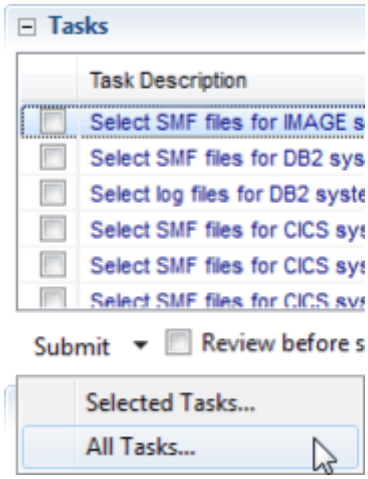


Figure 54. Using the plug-in to perform a workflow: submitting all tasks

The plug-in submits a task scheduler batch job on z/OS that submits each task in sequence.

When each task completes, you can view the corresponding job output in the **Workflow** tab.

For more information on using the plug-in, see the plug-in help.

The workflow tasks in this tutorial select log files, and then create extracts and transaction indexes from those log files. These are useful preliminary steps for analyzing a problem. The subject-matter expert can enhance session templates by adding workflow tasks that create batch reports. In the final note task, the expert can write instructions for analyzing the reports. Plug-in users can view the reports. Depending on the nature of the problem, the results of the report, and the instructions written by the expert, the plug-in user might have enough information to solve the problem without assigning it to an expert.

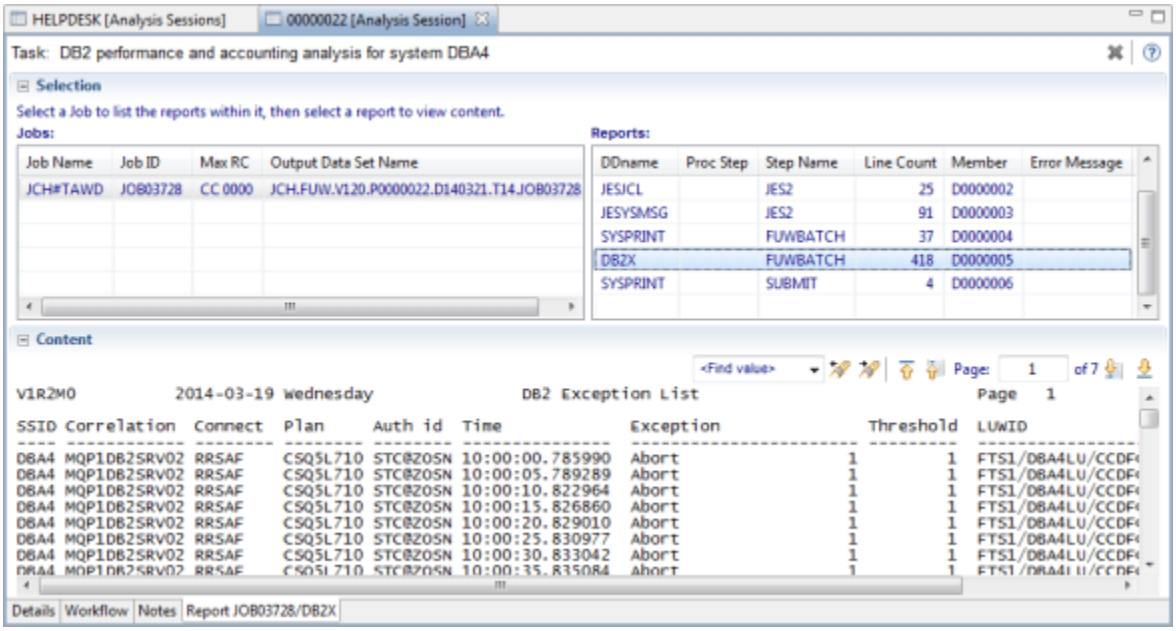


Figure 55. Viewing a batch report in the plug-in

Related concepts

z/OS Explorer plug-in

You can use the Transaction Analysis Workbench plug-in for IBM Explorer for z/OS (z/OS Explorer) to view or create sessions, submit z/OS batch jobs for session workflow tasks, and view the output of those batch jobs.

Related tasks

Defining systems for log file selection

Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Part 2. Configuring subsystems to collect data for analysis

Some subsystems, such as DB2 and IMS, write log records without requiring additional configuration. However, for other subsystems, or for specific types of log record such as performance, accounting, or trace-related data, you need to configure the subsystems to write log records.

Typically, each subsystem provides its own documentation that describes how to collect data for analysis. For example, the documentation provided with CICS describes how to collect CICS performance data. The instructions presented here reproduce some of that existing documentation, with a focus on collecting the specific data required for transaction analysis.

Related concepts

Analyzing system and subsystem instrumentation
These topics provide tips for using Transaction Analysis Workbench to analyze some types of log records from specific systems and subsystems.

Related reference

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Chapter 5. Collecting CICS performance data

To collect CICS performance data (SMF type 110, subtype 1, class 3 records; known in Transaction Analysis Workbench as log type CMF, log code 6E13), you must switch on CICS monitoring and activate the performance monitoring class. You can do this dynamically or at CICS startup.

Switching on CICS performance monitoring dynamically

To switch on collection of CICS performance data while CICS is running, use the CICS-supplied transaction **CEMT**.

The following procedure assumes that you know how to operate CICS from a console device and that you have the authority to use **CEMT**.

CICS offers several methods for dynamically switching on collection of CICS performance data. The following procedure uses **CEMT**. For other methods, see the CICS documentation.

Enter the following command on a CICS console device:

```
CEMT SET MONITOR ON PERF
```

Related tasks

Switching on CICS performance monitoring at CICS startup
To switch on collection of CICS performance class data when CICS starts, specify the CICS system initialization parameters MN=ON and MNPER=ON.

Switching on CICS performance monitoring at CICS startup

To switch on collection of CICS performance class data when CICS starts, specify the CICS system initialization parameters MN=ON and MNPER=ON.

CICS offers several methods for specifying system initialization parameters. For example, you can specify these parameters in the SYSIN data set of the CICS startup program DFHSIP, or in a CICS system initialization table (SIT) that you refer to in the PARM parameter of DFHSIP. You must know the method that your organization uses.

The following procedure involves restarting each CICS region for which you want to collect performance data.

1. Specify the following system initialization parameters:

   MN=ON
   Switches on CICS monitoring.

   MNPER=ON
   Activates the performance monitoring class.
2. Restart the CICS regions.

Related tasks

Switching on CICS performance monitoring dynamically
To switch on collection of CICS performance data while CICS is running, use the CICS-supplied transaction **CEMT**.

Chapter 6. Collecting CICS-DBCTL performance data

IMS DBCTL returns monitoring data to CICS that you can collect in CICS performance data (SMF type 110, subtype 1, class 3) records.

The following procedure assumes that you know how to assemble and link-edit the CICS monitoring control table (MCT).

The following procedure involves restarting each CICS region for which you want to collect performance data.

1. Edit the MCT source member, and add the event monitoring points (EMPs) defined in the CICS-supplied SDFHSAMP library member DFH$MCTD.
2. Assemble and link-edit the MCT.
3. Restart the CICS regions.

Chapter 7. Collecting CICS RMI performance data

The CICS resource manager interface (RMI), also known as the task-related user exit (TRUE) interface, enables CICS applications to use non-CICS resources, such as DB2 and IMS databases. RMI performance data consists of the elapsed time spent in the RMI, including specific fields for requests to DB2 and DBCTL.

The following procedure assumes that you know how to assemble and link-edit the CICS monitoring control table (MCT).

The following procedure involves restarting each CICS region for which you want to collect performance data.

1. Edit the MCT source member, and specify the parameter `RMI=YES` on the `DFHMCT TYPE=INITIAL` macro.
2. Assemble and link-edit the MCT.
3. Restart the CICS regions.

The collected data appears as the DFHRMI group of fields in CICS performance data (SMF type 110, subtype 1, class 3 records).

Chapter 8. Collecting CICS trace data

To collect CICS trace data (known in Transaction Analysis Workbench as log type CTR), either use the **CETR** transaction or the corresponding **SPCTRxx** SIT parameters.

To generate the trace records that correspond to Transaction Analysis Workbench trace levels 0 - 3, set the CICS tracing status according to the following table.

Domain	CICS trace levels
AP	1
EI	1 - 2
LD	1
PG	1
RA	1
RI	1
XM	1

The trace levels cited in the previous table are the CICS trace levels that you specify to set the CICS tracing status. CICS trace levels do not match Transaction Analysis Workbench trace levels.

Related concepts

Trace levels versus filters

Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Chapter 9. Collecting DB2 trace data

To collect DB2 trace data, including accounting data, use the DB2 command **START TRACE**.

You can record DB2 trace output to various destinations. Transaction Analysis Workbench can read DB2 trace output that has been recorded to the following destinations:

- z/OS generalized trace facility (GTF) data set
- z/OS system management facility (SMF); either an SMF log stream or a dumped SMF data set

There are several types of DB2 trace. The performance and accounting trace types are particularly useful for analyzing performance issues with DB2 threads and transactions that use DB2, such as CICS and IMS transactions.

Related concepts

Trace levels versus filters

Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Collecting DB2 performance trace data

To collect DB2 performance trace data, you must use the DB2 command START TRACE (PERFM). Collecting DB2 performance trace data can consume a large amount of processing resources and generate a large amount of data. You should activate only the IFCIDs that you need.

Trace levels

To help you decide which IFCIDs to activate, so that you can use the resulting data with Transaction Analysis Workbench to analyze DB2 behavior and performance, Transaction Analysis Workbench introduces the concept of *trace levels*. For DB2 performance traces, a trace level defines a set of IFCIDs. Each trace level, from 1 to 4, offers progressively more detail about DB2 activity, but with a higher cost of collection. In practice, for many DB2 behavior and performance issues, collecting trace levels 1 and 2 typically offers enough detail for useful analysis in Transaction Analysis Workbench.

Trace level 1: Program invocation

DB2 performance trace level 1 shows signon, thread allocation, and the flow of control between packages, stored procedures, and user-defined functions (UDFs).

Example DB2 **START TRACE** command for trace level 1:

```
-START TRACE(PERFM) DEST(SMF) CLASS(30)
  IFCID(86,87,112,162,163,177,233,325,380,381)
```

Trace level 2: SQL

DB2 performance trace level 2 shows SQL activity, including the start and completion of individual SQL statements.

Example DB2 **START TRACE** command for trace level 2, including IFCIDs from trace level 1:

```
-START TRACE(PERFM) DEST(SMF) CLASS(30)
  IFCID(86,87,112,162,163,177,233,325,380,381,
  22,53,55,58,59,60,61,65,66,68,69,70,71,72,73,74,75,82,83,84,85,88,89,121,122,316,318,350,499)
```

Trace level 3: I/O

DB2 performance trace level 3 shows input/output activity.

Example DB2 **START TRACE** command for trace level 3, including IFCIDs from trace levels 1 and 2:

```
-START TRACE(PERFM) DEST(SMF) CLASS(30)
  IFCID(86,87,112,162,163,177,233,325,380,381,
  22,53,55,58,59,60,61,65,66,68,69,70,71,72,73,74,75,82,83,84,85,88,89,121,122,316,318,350,499,
  6,7,8,9,10,15,16,17,18.20,26,27,28,44,45,95,96,105,107,125,127,128,218,223)
```

Trace level 4: All

Activating all IFCIDs will consume a large amount of processing resources and generate a large amount of data.

Trace levels and the Transaction Analysis Workbench TRACE command

The DB2 performance trace levels correspond to the filtering performed by the **TRACE** command that is available in the Transaction Analysis Workbench ISPF dialog log browser and in the Transaction Analysis Workbench batch report and extract utility. The **TRACE** command selects DB2 trace records (log type DTR) in the specified trace level and any preceding levels. For example, suppose you have used the DB2 **START TRACE** command to collect trace levels 1, 2, and 3 in an SMF file. If you display the SMF file in the log browser, then entering the command TRACE 2 displays the DB2 trace records for IFCIDs in trace levels 1 and 2, hiding IFCIDs that belong to trace level 3.

Collecting DB2 accounting data

To collect DB2 accounting data, you must use the DB2 command START TRACE (ACCTG).

If you want to use DB2 accounting records to analyze the behavior and performance of individual DDF or RRSAF DB2 threads, then set the DB2 subsystem parameter ACCUMACC to NO, to prevent DB2 from accumulating data for multiple DDF or RRSAF threads in a single accounting record. With ACCUMACC set to NO, each DB2 accounting record contains data for a single DDF or RRSAF thread.

Part 3. Analyzing problems using sessions

Using sessions offers a structured, team-oriented approach to analyzing problems. Sessions encapsulate information about a problem and its analysis that you can share with other users.

After registering a session and locating its associated log files, you can begin analyzing the problem by browsing logs and creating reports.

When you use a session, any option you select that generates new JCL, such as creating reports or extracts, creates a corresponding batch job task in the session workflow. If you submit the JCL, then you can view the job output in the session: on the session menu, select 2 **Workflow**, and then enter ? next to the task.

Alternatively, if you do not wish to use sessions, you can analyze problems using a personal ad hoc list of log files. However, using sessions has many advantages, such as allowing you to more easily share your analysis with other users.

1. Register (create) a session for the problem.
2. If you created the session from a template: perform the workflow tasks.

   Ideally, the final task in a workflow describes what to do next, such as interactively browsing extracts for records with field values that indicate specific problems. Otherwise, the following steps offer a general approach to problem analysis.
3. If you created the session without a template, or if you need to specify log files that cannot be automatically selected: add log files to the session.
4. Browse the logs or use the logs to create batch reports.

   While browsing logs, you can create tags that you and other users can later select to resume browsing at a particular position.
5. Write notes in the session history about your analysis of the problem.

Related concepts

Sessions, workflows, and templates
Transaction Analysis Workbench offers an optional framework for analyzing problems in *sessions*. A session encapsulates information about a problem and the analysis of that problem. This information can include the time period in which a problem occurred, the systems involved, and the corresponding log files.

Related tasks

Browsing logs interactively
You can use the Transaction Analysis Workbench ISPF dialog to browse records from different logs merged in a single view. You can select a record to browse its fields, and then select a field to browse a detailed description of its contents.

Chapter 10. Registering sessions

Registering (creating) a session is the recommended first step for using Transaction Analysis Workbench to analyze a problem.

If you want to use the Transaction Analysis Workbench automated file selection utility to select the log files associated with a problem, rather than manually specifying the log files yourself, then, before registering a session, you must perform the following tasks:

- Define to Transaction Analysis Workbench the systems that write to the log files. For details, see "Defining systems for log file selection" on page 68.
- Specify the following settings under Transaction Analysis Workbench dialog option 0.1 **Workbench Personal Settings**:
  - Workbench load library
  - Job statement information
  - JES2 control statements

  Transaction Analysis Workbench uses these settings to generate the JCL that runs the automated file selection utility.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**.
2. Enter **NEW** on the command line of the **Session Manager** panel.

   The **New Session** window opens.
3. If you want to base the new session on a template, press the Prompt function key (F4) to select the template.
4. On the **New Session** window, press Enter.

   The **Session Details** panel for the new session is displayed.
5. Specify any problem details that you want to.

   To enter a multi-line description, position the cursor on the **Description** label, and then press Enter.

   All details are optional except for the key, which is a unique identifier that Transaction Analysis Workbench automatically assigns to the session.

   By default, the status of a new session is OPEN.

   Ensure that the **Zone** field identifies the time zone of the system that created the logs for this session. Each session can specify a different time zone. The default value of this session **Zone** field is the value of the global **Time Zone** field under option 0.4 **Time Control**. For logs that were created locally, on the same system on which you are running Transaction Analysis Workbench, or on a system in the same time zone, specify `LOCAL`. Otherwise, you must specify either a time zone offset such as `-0400` (New York), or `GMT` to indicate no offset.

   If you want to use the Transaction Analysis Workbench automated file selection utility to select the log files associated with this session, then you need to specify the following details:

   - The from and to date and time when the problem occurred.
   - The systems that the problem involves.

     If you created the session from a template, then the session might have inherited system details from the template.

     You must already have defined these systems to Transaction Analysis Workbench with the details required for automated file selection.

   If you do not want to use automated file selection, then you can specify systems here without defining them first, for documentation only. You can choose to create the corresponding system definitions later, or not at all.

If you select a different template after creating the session, then any systems, tasks, or files that do not already exist in the session are appended to the session.

6. Exit the details panel.

   The session menu is displayed.

If you created the session from a template, then the session will likely have inherited a workflow from the template. Typically, the workflow includes tasks that automate file selection. Select session menu option 2 **Workflow**, and then run the tasks; for example, by entering the **SCHED** command to run the tasks in sequence.

If you created the session without a template, then create and run workflow tasks to automate file selection. Select session menu option 2 **Workflow**, use the **AUTO** command to create the tasks, and then run the tasks.

Some log files cannot be automatically selected; you need to manually add them to the session. To manually add files to a session, select session menu option 3 **Files**.

Related concepts

Sessions, workflows, and templates

Transaction Analysis Workbench offers an optional framework for analyzing problems in *sessions*. A session encapsulates information about a problem and the analysis of that problem. This information can include the time period in which a problem occurred, the systems involved, and the corresponding log files.

Related tasks

Adding log files to a session

If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Performing workflow tasks

You can perform tasks in a session workflow either by selecting individual tasks or by scheduling multiple tasks to run in sequence.

Defining systems for log file selection

Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Chapter 11. Performing workflow tasks

You can perform tasks in a session workflow either by selecting individual tasks or by scheduling multiple tasks to run in sequence.

You must have created a session that contains a workflow. The easiest way to create a session that contains a workflow is to create a session from a template. The new session inherits the workflow from the template.

You can perform tasks in a session workflow at any time. Normally, you perform tasks immediately after creating a session from a template. A typical workflow includes tasks to automate the selection of log files for a problem.

How you perform a task depends on whether the task is a batch job task or a note task:

- Performing a batch job task involves running a batch job. You can either select a single batch job task and submit it, or you can run multiple tasks in sequence by submitting a task scheduler job.
- Performing a note task involves reading the note. A note typically contains tips or instructions from the subject-matter expert who developed the workflow.

1. On the session menu, select option 2 **Workflow**.

   The **Tasks** panel is displayed, showing the list of tasks for the session.
2. Either schedule tasks to run in sequence or select a single task. Typically, if you have just created a session from a template, you would schedule all tasks.

Option	Description
Schedule all tasks	Enter **SCHED** on the command line. The JCL for the task scheduler job is displayed in a **Notepad** panel. Submit the job: enter **SUB** on the command line.
Schedule some tasks	Type line action S next to each task, and then enter **SCHED** on the command line. The JCL for the task scheduler job is displayed in a **Notepad** panel. Submit the job: enter **SUB** on the command line.
Select a single task	Enter line action S next to the task. The JCL or the note, depending on the type of task, is displayed in a **Notepad** panel. If the task is a batch job task, submit it: enter **SUB** on the command line.

   The **Notepad** panel is an ISPF Edit panel: you can edit its contents. If you edit a task, and then press the Exit function key (F3) to return to the **Tasks** panel, the changes to the note or the JCL are saved. If you do not want to save changes, press the Cancel function key (F12) instead.
3. Press the Cancel function key (F12) or the Exit function key (F3) to return to the **Tasks** panel.

   Tip: To refresh the **Tasks** panel, and see any updates to the **Status** column for batch job tasks, enter **REFRESH** on the command line.

   The task scheduler job submits a separate job for each scheduled task. If the job for a task fails to submit or its completion code exceeds the **MAXRC** parameter specified for that task in the task scheduler JCL, then scheduling stops and remaining tasks are not performed. The default is `MAXRC=0`; each task job must end with completion code zero (`CC 0000`).

   By default, the task scheduler skips tasks that have one of the following status values:

   DONE
   CC *nnnn*, where *nnnn* is less than or equal to the **MAXRC** parameter for the task
   IGNORE

If a task scheduler job includes note tasks, the job writes each note to a separate sysout data set.

Transaction Analysis Workbench updates the **Status** column of batch job tasks to reflect the task status. For example, when the job for the task completes, Transaction Analysis Workbench updates the task status to reflect the maximum return code. `CC 0000` indicates that a task completed successfully.

Transaction Analysis Workbench does not change the status of note tasks. To change the status of a note task, enter line action T next to the task. For example, if you have completed the task described in a note, you might set the status to DONE; if you are collaborating with other users, you might set the status to ACTIVE to indicate that you are working on the task.

To view the output of a batch job task, enter line action ? next to the task.

The final task in a workflow is typically a note describing what to do next; in that case, follow those instructions.

Otherwise, if the workflow consists of tasks to select log files, and then create extracts and transaction indexes, then you should analyze those extracts and transaction indexes, either by creating reports or browsing log records, or both. To view the list of files that the tasks have added to the session, select option 3 **Files** from the session menu. To create reports from those files, select option 4 **Reporting** from the session menu. To browse log records, select option 5 **Investigate** from the session menu.

Related tasks

Registering sessions
Registering (creating) a session is the recommended first step for using Transaction Analysis Workbench to analyze a problem.

Related reference

Task scheduler
The task scheduler manages the execution of some or all of the tasks in a session workflow, allowing you to perform a workflow without user intervention. The scheduler runs as a batch job, submitting the tasks in order, stopping if a task fails.

Workflow task status

Each task in a session workflow has a status. Transaction Analysis Workbench updates the status of batch job tasks to reflect whether the corresponding job is active, failed to submit, or completed. You can also change the status of tasks manually.

Table 3. Workflow task status values

Status	Description	Notes
NOT DONE	Not started yet.	NOT DONE is the initial status of all tasks.
ACTIVE	In progress.	When the job for a batch job task starts, Transaction Analysis Workbench updates the task status to ACTIVE.
DONE	Completed successfully.	DONE is a task status set manually by the user. This value is not set by Transaction Analysis Workbench; instead, when a batch job task completes, Transaction Analysis Workbench updates the status to reflect the maximum return code. By default, the task scheduler skips tasks that have a status of DONE. To include tasks that have a status of DONE, specify the parameter **REDOIFDONE** on the **REQUEST** command of the SYSIN data set for the task scheduler job.

Table 3. Workflow task status values (continued)		
Status	**Description**	**Notes**

Max-RC	Maximum return code of batch job.	When a batch job task completes, Transaction Analysis Workbench updates the task status to reflect the maximum return code of the corresponding job as displayed by the **Max-RC** column of the **Status** panel of SDSF.
		For example:
		`CC 0000` (completed successfully)
		`CC 0008`
		`ABEND S622`
		`ABEND U4095`
		`JCL ERROR`

FAILED	Scheduled job for batch job task failed to submit.	A scheduled batch job task did not run because it failed to submit. For example, Transaction Analysis Workbench could not resolve substitution variables in the task JCL.
IGNORE	Not required	If you want the task scheduler to skip a task, set the task status to IGNORE.

Job step to save workflow task output

Session workflow batch job tasks include a "post-processing" step that saves the task sysout data sets as members of a library data set. Saving task output enables you to view the output even when the original job is purged. You can use the Transaction Analysis Workbench ISPF dialog to view the task output.

Job step JCL

You do not need to write the JCL for this step yourself; Transaction Analysis Workbench generates the JCL for you when you create a task. However, you might find it useful to understand how this step works and where the task output is saved.

This step is appended as the last step in the job.

The following listing shows an example of the JCL for this step before any substitution variables have been resolved:

```
//*====================================================================
//* Save job output
//*====================================================================
//SUBMIT   EXEC PGM=FUWBATCH,PARM='SUBMIT',COND=EVEN
//STEPLIB  DD  DSN=%FUWLINK,
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUT2   DD  SYSOUT=(*,INTRDR)
//SYSUT1   DD  DATA,DLM=$$
//%JOBCARD JOB
//*
//SAVE     EXEC PGM=FUWBATCH
//STEPLIB  DD  DSN=%FUWLINK,
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//FUWPROBR DD  DSN=%FUWPROBR,
//             DISP=SHR
//FUWSAVE  DD  DSN=FUW.%PID.+DATE.+TIME.+XTYP,
//             DISP=(NEW,CATLG),DSNTYPE=LIBRARY
//SYSIN    DD  *
REQUEST=OUTPUT
```

```
JOBNAME=+JOBNAME
JOBID=+JOBID
SESSION=%PID
TASK=%TASKID
/*
$$
```

The step submits a new job. On the first line of the step, the PARM='SUBMIT' parameter in the EXEC statement instructs the FUWBATCH program to perform the following actions:

1. Read the JCL specified by the SYSUT1 DD statement.
2. Resolve any runtime JCL substitution variables. Runtime JCL substitution variables are prefixed by a plus sign (+).

   At this point, Transaction Analysis Workbench has already resolved the JCL generation-time substitution variables. Transaction Analysis Workbench resolved these variables when the user selected the task for scheduling, before the job was submitted. JCL generation-time substitution variables are prefixed by a percent sign (%).
3. Submit the resolved JCL to the JES internal reader, the destination specified by SYSUT2.

The new job has the same name as the parent task job that submitted it; the new job does not start until the task job ends.

The new job also runs the FUWBATCH program, but with the command REQUEST=0UTPUT. The REQUEST=0UTPUT command copies the sysout data sets from the task job to the library specified by the FUWSAVE DD statement, and then attaches that library data set name to the task information stored in the session repository.

Location of the saved output

Transaction Analysis Workbench saves task sysout data sets to the location specified in your personal profile by the extract data set name template for sessions. To specify the template, select Transaction Analysis Workbench ISPF dialog option 0.3 **Extract Data Set Allocations**.

In this context, the template substitution variable %XTYP resolves to OUTPUT. For example, if your profile specifies the template 'FUW.%PID.%DATE.%TIME.%XTYP', then, for a session with key 00000042, the sysout data sets might be saved to the library 'FUW.00000042.D140802.T093055.OUTPUT'.

Some of the substitution variables in the extract data set name template cannot be resolved until run time. When generating the JCL for the FUWSAVE DD statement, Transaction Analysis Workbench replaces the % prefix of these variables with +, delaying substitution until the FUWBATCH programs runs with the **SUBMIT** parameter.

Viewing the saved output in the ISPF dialog

To view the save output in the Transaction Analysis Workbench ISPF dialog: on the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Chapter 12. Adding log files to a session

If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Before you can associate log files with a problem, you must register a session for the problem.

The easiest way to add log files to a session is to submit workflow tasks. When you register a session, you can optionally base the session on a session template. A session template can contain workflow tasks that add log files to the session.

However, if your session has no workflow tasks, or you need to manually add log files that cannot be automatically selected, then follow the procedure described here to add log files to a session.

Choose one of the following options:

Option	Description
Create and then submit workflow tasks that run the automated file selection utility	Choose this option if you want Transaction Analysis Workbench to select log files and then automatically create extracts and transaction indexes from those log files. a. On the session menu, select option 2 **Workflow**. The **Tasks** panel is displayed. b. Enter **AUTO** on the command line. c. Submit the tasks.
Run the automated file selection utility without creating workflow tasks first	Choose this option if you want Transaction Analysis Workbench to select log files without automatically creating extracts and transaction indexes from those log files. a. On the session menu, select option 3 **Files**. The **Locate and Manage Log Files** panel is displayed. b. Enter **AUTO** on the command line. c. Specify the criteria of the file that you want to select. d. Press Enter to generate the JCL to run the automated file selection utility. The JCL is displayed in a **Notepad** panel. As for any action you perform in the ISPF dialog that generates JCL for a session, Transaction Analysis Workbench creates a corresponding task for this JCL in the session workflow. e. Enter **SUB** on the command line to submit the job. f. Press the the Exit function key (F3) to exit the **Notepad** panel. g. When the job is complete, enter **REFRESH** on the command line to refresh the list of log files, displaying any files added by the job.
Enter the log file details manually	If you know the data set names or log stream names, enter the details yourself: a. On the session menu, select option 3 **Files**. b. Enter **NEW** on the command line. c. Specify the file details.

The **AUTO** command on the **Locate and Manage Log Files** panel and **AUTO** command on **Tasks** panel have different results:

- On the **Locate and Manage Log Files** panel, the **AUTO** command generates JCL that selects log files.

- On the **Tasks** panel, the **AUTO** command creates tasks: tasks that select log files, and tasks that create extracts and transaction indexes from the selected log files.

After adding log files to a session, you can create reports from the log files (session menu option 4 **Reporting**) or browse the log files (option 5 **Investigate**).

Tip: In some cases, you might want to exclude a log file from the **Investigate** panel, but keep it on the list of files for other uses, such as batch reporting. For example, you might have added a large log file to a session, and then created a smaller extract for interactive browsing. You want to keep the original on the list of files, but you do not want users to inadvertently select that larger file, or both files together, for interactive browsing. To exclude a log file from the **Investigate** panel, enter line action X; to include it later, enter X again. To browse a log file that has been excluded from the **Investigate** panel, select it from the complete list of log files on the **Locate and Manage Log Files** panel: enter S to browse the entire file, or `W` to browse a time slice.

Related concepts

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Registering sessions
Registering (creating) a session is the recommended first step for using Transaction Analysis Workbench to analyze a problem.

Browsing logs in a session
After creating a session and adding log files to the session, you can use the Transaction Analysis Workbench ISPF dialog to browse the logs.

Creating reports
You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Tutorial: Creating a session and browsing a log
This tutorial shows you how to create a session for a problem, manually specify a log file for the problem, and then browse the log file.

Tutorial: Automating selection of IMS logs
This tutorial shows you how to define an IMS system to Transaction Analysis Workbench, and then use that system definition to locate the related IMS log files (SLDS or OLDS) for a particular time interval.

Tutorial: Automating selection of DB2 logs
This tutorial shows you how to define a DB2 system to Transaction Analysis Workbench, and then use that system definition to locate the related DB2 log files for a particular time interval.

Chapter 13. Creating extracts of log files in a session

Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Before using the procedure described here to create extracts, you must register a session and add the log files that contain the records that interest you.

You can use several methods to create extracts:

- The procedure described here: on the **Locate and Manage Log Files** panel for a session, enter CX next to a log file from which you want to create an extract.
- The easiest way: create a session based on a template that contains workflow tasks to select log files and create extracts from those log files. To create extracts, you run the workflow tasks.
- On the **Tasks** panel for a session, you can use the **AUTO** command to create a sequence of workflow tasks: tasks that select log files, followed by tasks that create extracts of those log files.
- If you do not want to use a session, you can create an extract of log files that are in your personal ad hoc list of files.
- While browsing logs, you can use the **EXTRACT** command to create an extract of the displayed log records.

1. On the session menu, select option 3 **Files**.
2. Enter CX next to a log file from which you want to create an extract.
3. Specify the data set name of the extract that you want to create.

   The panel suggests a data set name based on a template that you specify in ISPF dialog option 0.3 **Extract Data Set Allocations**.
4. Optionally, specify filtering criteria and an interval to extract only matching records.
5. Press Enter to generate the request JCL.

   The JCL is displayed in a **Notepad** panel.
6. Submit the job: enter **SUB** on the command line.
7. Press the Exit function key (F3) to return to the list of log files in the session.
8. When the job completes, refresh the panel to show the new extracts in the list of log files: enter **REFRESH** on the command line.

   If the list of files does not contain the expected extracts, check the job output for errors: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task. Check the SYSPRINT data set for error messages.

Related concepts

Report and extract utility
The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Extracts
You can use Transaction Analysis Workbench to create extracts of original log files, containing the subset of records that you need to analyze a problem, within the relevant time period.

Related tasks

Creating extracts of log files in your ad hoc list
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Saving records displayed in the log browser to an extract file

You can create an extract file containing records displayed in the log browser from the current scroll position to the bottom of data marker, or a selected block of records.

Related reference

EXTRACT command
Writes the selected log records to an extract data set.

Chapter 14. Creating transaction indexes

Creating transaction indexes from original log files is a useful preliminary step for problem analysis. Where original log files typically contain many types of record, a transaction index contains only a single type of record. Analyzing transaction index records is a useful starting point for analyzing problems with transactions.

Before using the procedure described here to create transaction indexes, you must register a session and add the log files from which you want to create the transaction indexes.

You can use several methods to create transaction indexes:

- The procedure described here: on the **Locate and Manage Log Files** panel for a session, enter `CI` next to a log file from which you want to create a transaction index.
- The easiest way: create a session based on a template that contains workflow tasks to select log files and create transaction indexes from those log files. To create transaction indexes, you run the workflow tasks.
- On the **Tasks** panel for a session, you can use the **AUTO** command to create a sequence of workflow tasks: tasks that select log files, followed by tasks that create transaction indexes from those log files.
- To create a DB2 accounting index: on the session menu, select option 4 **Reporting**, select reporting option 5 **DB2**, and then request an extract.
- Creating CICS-DBCTL reports from original log files involves creating CICS and IMS transaction indexes: select option 4 **Reporting**, select reporting option 3 **CICS-DBCTL**, and then follow the on-screen prompts to select the appropriate options.

1. On the session menu, select option 3 **Files**.
2. Enter `CI` next to a log file from which you want to create a transaction index.
3. Specify the types and data set names of the transaction indexes that you want to create.
4. Optionally, specify exception criteria and an interval to extract only matching records.
5. Press Enter to generate the request JCL.

   The JCL is displayed in a **Notepad** panel.
6. Submit the job: enter **SUB** on the command line.
7. Press the Exit function key (F3) to return to the list of log files in the session.
8. When the job completes, refresh the panel to show the new transaction indexes in the list of log files: enter **REFRESH** on the command line.

   If the list of files does not display the expected transaction indexes, check the job output for errors: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task. Check the SYSPRINT data set for error messages.

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Related tasks

Creating DB2 accounting exception reports and extracts

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports and extracts of DB2 accounting records that match the exception criteria that you specify.

Creating CICS-DBCTL reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

Related reference

EXTRACT command

Writes the selected log records to an extract data set.

Chapter 15. Creating reports

You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Related concepts

Reporting
Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

Report and extract utility
The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Related tasks

Adding log files to a session
If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Tutorial: Submitting a batch report
This tutorial shows you how to create a batch report for a session.

Related reference

Hardware and software prerequisites
Before you install and configure Transaction Analysis Workbench, make sure that your environment meets the following minimum hardware and software requirements.

REPORT command
Requests a report.

Creating IMS reports using IMS Performance Analyzer

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports that help you to analyze IMS transactions. To create these IMS reports, you must have IMS Performance Analyzer installed on your system.

Before you can use the dialog to create IMS reports, you must create a session, and then add the associated log files.

Check that ISPF dialog option 0.5 **IMS Tools Settings** specifies the IMS Performance Analyzer load library. The default low-level qualifier of this library is SIPILINK. The dialog uses this data set name in the STEPLIB DD statement of the report JCL.

1. On the session menu, select option 4 **Reporting**.
2. Select reporting option 1 **IMS**.
3. Select the types of analysis that you want.

   The following table shows how the type of analysis determines the IMS Performance Analyzer report generated:

Table 4. Creating IMS reports: how the type of analysis determines the IMS Performance Analyzer report generated

Type of analysis	IMS Performance Analyzer report generated	IMSPALOG command operand
Individual transaction detail (see Note)	Transaction Transit List	`LIST`
Transaction statistical summary (see Note)	Transaction Transit Summary	`SUMMARY`
IMS system resources	Internal Resource Usage	`IRUR`
Deadlock analysis	Deadlock	`DEADLOCK(LIST,SUMMARY)`

Note: The options under **Focus of transaction analysis** apply only to two types of analysis: individual transaction detail and transaction statistical summary. Each option under **Focus of transaction analysis** generates a separate form-based report containing fields that are relevant to that focus. For details, see the **FIELDS** operand in the JCL generated later in this procedure.

4. If you selected individual transaction detail or transaction statistical summary, select one or more focus of analysis.
5. Optionally, specify a report interval. If you omit the report interval, the reports use the entire input IMS log files.
6. Select the IMS system or the IMS log file that you want to report against.

   If you select an IMS system, then the reports use all of the IMS log files associated with that system for this session.
7. Press Enter to generate the report JCL.

   The JCL is displayed in a **Notepad** panel.
8. Submit the job: enter **SUB** on the command line.

   If you do not have IMS Performance Analyzer installed on your system, the job will fail.
9. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

   For more information about the reports, see the IMS Performance Analyzer documentation.

Creating CICS reports using CICS Performance Analyzer

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports that help you to analyze CICS transactions. To create these CICS reports, you must have CICS Performance Analyzer installed on your system.

Before you can use the ISPF dialog to create CICS reports, you must create a session, and then add the associated log files.

Check that ISPF dialog option 0.6 **CICS Tools Settings** specifies the CICS Performance Analyzer load library. The default low-level qualifier of this library is SCPALINK. The ISPF dialog uses this data set name in the STEPLIB DD statement of the report JCL.

1. On the session menu, select option 4 **Reporting**.
2. Select reporting option 2 **CICS**.
3. Select the types of analysis that you want.

   The following table shows how the type of analysis determines the CICS Performance Analyzer report generated:

Table 5. Creating CICS reports: how the type of analysis determines the CICS Performance Analyzer report generated		
Type of analysis	**CICS Performance Analyzer report generated**	**CICSPA command operand**
Individual transaction detail	Performance List	**LIST**
Transaction statistical summary	Performance Summary	**SUMMARY**
Transaction suspend time breakdown	Wait Analysis and Statistics Alerts	**WAITANAL** and **STATSALERT**

4. Select one or more focus of analysis.

   Each focus generates a separate form-based report containing fields that are relevant to the focus. For details, see the **FIELDS** operand in the JCL generated later in this procedure.
5. Optionally, specify a report interval. If you omit the report interval, the reports use the entire input SMF file.
6. Select the z/OS system or the SMF file that you want to report against.

   If you select a CICS system, then the reports use all of the SMF files associated with that system for this session.
7. Press Enter to generate the report JCL.

   The JCL is displayed in a **Notepad** panel.
8. Submit the job: enter **SUB** on the command line.

   If you do not have CICS Performance Analyzer installed on your system, the job will fail.
9. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

   For more information about the reports, see the CICS Performance Analyzer documentation.

Creating CICS-DBCTL reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

You will need at least one of the following input files, covering the time period that you want to report:

- An SMF file containing CMF records for CICS-DBCTL transactions.
- An IMS log file containing records of DBCTL activity.

The input files required depend on the reports that you want to create:

- CICS reports require an SMF file.
- IMS-DBCTL reports require an IMS log file.
- Combined CICS and IMS reports require both an SMF file and an IMS log file.

1. On the session menu, select option 4 **Reporting**.
2. Select reporting option 3 **CICS-DBCTL**.
3. Select a type of report request.
4. Optionally, specify a report interval.

   If you omit the report interval, the reports cover the entire input log files.
5. Optionally, specify exception criteria to select only log records for poorly performing or failed transactions.
6. Select the systems or the files that you want to report against.
7. Press Enter to generate the request JCL.

   The JCL is displayed in a **Notepad** panel.

Tip: The **OUTPUT** parameter of the **REPORT** command in the SYSIN data set specifies the ddname of the output data set that will contain the report.

8. Submit the job: enter **SUB** on the command line.
9. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Related reference

CICS-DBCTL reports
CICS-DBCTL reports process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.

REPORT CICS-DBCTL command: CICS reports
Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

REPORT CICS-DBCTL command: combined CICS and IMS reporting
Requests a report that combines output from earlier **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands to show CICS DBCTL transaction response times across CICS and IMS.

REPORT IMS-DBCTL command: IMS reporting
Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

Creating SMF reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of SMF records.

Before you can use the ISPF dialog to create SMF reports, you must create a session, and then add the associated SMF files.

1. On the session menu, select option 4 **Reporting**.
2. Select reporting option 4 **SMF**.
3. Select the types of analysis that you want.

   The following table shows how the type of analysis determines the SMF reports generated.

Table 6. Creating SMF reports: how the type of analysis determines the SMF report generated

Type of analysis	SMF reports generated
CPU, storage, and paging	"SMF type 70-1: RMF Processor Activity report" on page 443 "SMF type 75: RMF Page Data Set Activity report" on page 445 "SMF type 78-2: RMF Virtual Storage Activity report" on page 447
Address space accounting	"SMF type 30: Address Space Activity report" on page 435 This report is filtered to only show data for CICS, IMS, DB2, and IRLM-supplied program names. For details, see the **COND** statements in the generated JCL.
MVS System Logger	"SMF type 88-1: System Logger Log Stream Summary report" on page 450
DASD data set activity	"SMF type 64: VSAM Data Set report" on page 442 "SMF type 42-6: DASD Data Set I/O report" on page 438
DB2 thread accounting	"SMF type 101: DB2 Thread Accounting Summary report" on page 453

Table 6. Creating SMF reports: how the type of analysis determines the SMF report generated (continued)	
Type of analysis	**SMF reports generated**
IBM MQ thread accounting	"SMF type 116-0: IBM MQ Accounting Class 1 reports" on page 456 "SMF type 116-1: IBM MQ Accounting Class 3 reports" on page 458
APPC conversations	"SMF type 33-2: APPC/MVS Conversation List report" on page 437
IMS IRLM long lock	"SMF type 79-15: IRLM Long Lock Detection report" on page 449

4. Optionally, specify a report interval. If you omit the report interval, the reports use the entire SMF file.
5. Select the MVS system (image) or the SMF file that you want to report against.

   If you select a system, then the reports use all of the SMF files associated with that system for this session.
6. Press Enter to generate the report JCL.

   The JCL is displayed in a **Notepad** panel.

   Tip: The **OUTPUT** parameter of the **REPORT** command in the SYSIN data set specifies the ddname of the output data set that will contain the report.
7. Submit the job: enter **SUB** on the command line.
8. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Related reference

REPORT command for SMF reports
Requests either an SMF Recap report that gives an overview of the contents of an SMF file or, for some SMF record types only, a report that processes a particular type of SMF record; these reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

SMF reports
Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

Creating DB2 accounting exception reports and extracts

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports and extracts of DB2 accounting records that match the exception criteria that you specify.

Before you can use the ISPF dialog to create DB2 accounting exception reports and extracts, you must create a session, and then add the SMF file that contains the DB2 accounting records you want to process.

1. On the session menu, select option 4 **Reporting**.
2. Select reporting option 5 **DB2**.

   The DB2 accounting exception report and extract options are divided into two panels: the exception criteria are on the second panel.
3. Select whether you want to request a report, an extract, or both.
4. Optionally, specify a report interval.

   If you omit the report interval, the report and extract cover the entire input log file.
5. If you are requesting an extract, specify the name of the extract data set.

6. Optionally, specify filtering criteria for SSIDs, plans, and connection types.
7. Select the DB2 system or the SMF file that you want to report against.

   If you select a system, then the reports and extracts use all of the SMF files associated with that system for this session.
8. Specify exception criteria on the second panel.
9. Press Enter to generate the request JCL.

   The JCL is displayed in a **Notepad** panel.

   Tip: The **OUTPUT** parameter of the **REPORT** command in the SYSIN data set specifies the ddname of the output data set that will contain the report.
10. Submit the job: enter **SUB** on the command line.
11. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Related reference

DB2 accounting exception reports and extracts
DB2 accounting exception reports and extracts are generated from DB2 accounting (SMF type 101) records. DB2 accounting records from the specified input files are checked against optional *filtering criteria* for the DB2 subsystem ID (SSID), plan, and connection type. Records that match the filtering criteria are checked against *exception criteria* that specify thresholds for various performance-related values, such as CPU time.

Creating OPERLOG reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch printouts of OPERLOG, the z/OS operations log (log stream).

You can create OPERLOG reports either from a session (under dialog option 1 **Sessions**) or from your personal ad hoc list of log files (option 4 **Process**).

1. Use either of the following methods to select OPERLOG reporting:

Option	Description
From a session	On a session menu, select option 4 **Reporting**, and then select reporting option 6 **OPERLOG**.
From your personal ad hoc list of log files	a. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**. b. Insert the following log file name, if it is not already listed: `OPERLOG:SYSPLEX.OPERLOG` c. Enter line action SUB next to the log file name. The **Submit Batch Request** window opens. d. Type / next to **Report**. **Note:** OPERLOG reports ignore the report formatting options (for example, **Form** or **STD**).

2. Specify a report interval.
3. Optionally, specify a filter:

Option	Description
From a session	Specify either or both of the following criteria:

Option	Description
	Message ID A message prefix (for example, ICH for any RACF message) or a message number (for example, ICH70001I). **Find text** A string anywhere in the message. If you specify both criteria, they are combined by a logical AND: the report only includes records that meet both criteria.
From your personal ad hoc list of log files	Select the name of a predefined filter that specifies conditions for OPERLOG records (log type MVS, log code CA52). **Tip:** To define a new filter now, without returning to the primary menu: enter **FILTERS** on the command line to display the list of filters, and then enter **NEW**.

4. Press Enter to generate the report JCL.

   The JCL is displayed in a **Notepad** panel.
5. Submit the job: enter **SUB** on the command line.
6. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task.

Related reference

REPORT command for OPERLOG reports
Prints selected records from OPERLOG, the z/OS operations log (log stream).

Chapter 16. Session history: writing notes, resuming browsing from tags, and reviewing jobs

Each session has a history that consists of jobs for workflow tasks that have been run, notes that you or other users have written about the session, and tags that bookmark a position in the log browser.

1. On the session menu, select option 6 **History**.

   The **History** panel for the session is displayed.

   The **Type** column identifies the type of history item:

   JOB
   A job that has been run in the session. These jobs correspond to tasks in the session workflow. The **History** panel shows a separate job for each time a task has run.

   NOTE
   A comment that a user has written to document the problem analysis.

   Tip: Notes in the session history are not related to note tasks in the session workflow. Use notes in the session history for ad hoc commentary about the analysis. Use note tasks to record or recommend specific analysis tasks.

   TAG
   A bookmark to the time stamp of a particular log record, with a comment. Typically, the comment describes why that log record is of interest.

   Users can create tags while browsing logs. A session history also contains a tag for the most recent browsing position of each user, enabling you to resume browsing at the position another user was browsing.

   For jobs, the **Description** column shows the description of the corresponding workflow task. Otherwise, the **Description** column shows the first line of the note or tag.
2. Choose one of the following actions:

   - To write a new note, enter **NEW** on the command line. A new, empty note is displayed in the ISPF editor.

     Tip: Where appropriate, instead of writing notes here on the session **History** panel, consider creating note tasks on the session **Tasks** panel.
   - To delete any item, enter D next to that item.
   - To edit a note, the description of a tag, or job JCL, enter S next to that item.
   - To display job output, enter ? next to a job.
   - To resume browsing logs at the position bookmarked by a tag, enter R next to a tag.

Related tasks

Bookmarking log records with permanent tags
While browsing logs in a session, you can tag a record, and then you or other users can later resume browsing at that position by selecting the tag in the session history. Resuming from a tag resumes not just the browsing position, but also aspects of the browsing state, including the filter that was active at the time the tag was created, the log files that were selected, and the time slice (if any).

Part 4. Browsing logs interactively

You can use the Transaction Analysis Workbench ISPF dialog to browse records from different logs merged in a single view. You can select a record to browse its fields, and then select a field to browse a detailed description of its contents.

You can start browsing logs either from a session or from your personal ad hoc list of files (ISPF dialog option 4 **Process**).

If you are a new user, or you just want to browse logs without the additional keystrokes involved in creating a session, start browsing from your personal ad hoc list.

Related concepts

Interactive log browsing
The Transaction Analysis Workbench ISPF dialog user interface includes a log browser that offers a consistent user interface for all supported record types, with features for navigating, finding, filtering, bookmarking, formatting, and extracting log records. You can select any combination of supported logs and browse them merged in a single view.

Related tasks

Tutorial: Creating a session and browsing a log
This tutorial shows you how to create a session for a problem, manually specify a log file for the problem, and then browse the log file.

Analyzing problems using sessions
Using sessions offers a structured, team-oriented approach to analyzing problems. Sessions encapsulate information about a problem and its analysis that you can share with other users.

Chapter 17. Browsing logs in your personal ad hoc list

Using your personal ad hoc list of logs offers a private alternative to using sessions. There might be occasions when you want to browse logs without registering a session. Simply add logs to your personal ad hoc list, and then select them for browsing.

This task can be summarized in three steps:

1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.
2. Enter the location of a log. For example, the data set name of a dumped SMF file.
3. Enter S next to the log to browse its entire contents, or `W` to browse a time slice.

The following procedure expands on this summary, with details for different logs.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.

   Your personal ad hoc list of logs is displayed.
2. In the **Log File** column, enter the data set names or the log stream names of one or more logs that you want to browse.
   - To specify a fully qualified data set name, enclose the name in single quotes. For example:

     ```
     'MVS1.SMF.MAN1.DUMP'
     ```
   - To specify a log stream, prefix the log stream name with the log stream type, followed by a colon. For example:

     ```
     OPERLOG:SYSPLEX.OPERLOG
     ```
3. Depending on the type of log, you might also need to enter values in the following columns:

   Rel
   (IMS log files only.) Overrides, for this log only, the default IMS release specified in the **IMS Release** field at the top of the panel.

   You need to specify the IMS release because the format of IMS log records varies between releases. If you specify an incorrect release, Transaction Analysis Workbench might incorrectly interpret the log records.

   Filter
   (Optional, for any log.) Specifies a filter that you want to use when processing this log:
   - The name of a filter that is stored in the control repository
   - A log type
   - A log code
   - A log type and code separated by a colon (:), such as `IMS:01` or `DTR:3`

   Zone
   (Optional, for any log.) Overrides, for this log only, the default time zone specified in the **Zone** field at the top of the panel.

   UNIT and VOLSER
   (Uncataloged data sets only.) The device type (**UNIT**) and volume serial number of the data set. If you specify **VOLSER**, you must also specify **UNIT**.

   To scroll hidden columns into view, press the Left function key (F10) or the Right function key (F11).
4. Decide whether you want to browse entire logs or a *time slice*: a specific time period across one or more logs.
   - **To browse entire logs**, type S next to one or more logs, and then press Enter.
   - **To browse a time slice:**

a. Type `W` next to one or more logs, and then press Enter.

   The **Investigate** panel is displayed, listing the selected logs.

b. Specify the start and duration of the time slice that you want to browse, and then press Enter. To set the time slice to match the start and duration of a particular log, enter T next to the log.

   The **Coverage** column displays information about how much of the time slice, if any, each log covers.

c. Select the logs that you want to browse:

   – To select all logs, enter S on the first line under the slash (**/**) column heading.

   – To select one or more logs, type S next to each log, and then press Enter.

Time slicing uses a sampling method that favors speed over accuracy to locate log records in large files. The sampling method sometimes includes records that are just outside the specified time slice. When you specify a time slice, the log browser scrolls to the first log record in the time slice. If you press the Backward function key (F7), you might see records that are outside the specified time slice, but which have been included by the sampling method.

The **Duration** column displays the length of time that each log covers. If the **Duration** column is not visible, press the Right function key (F11) to scroll the column into view.

Tips:

- For large log files and log streams, browsing a time slice improves performance.
- To generate JCL that creates a CSV file, extract, or formatted record report, type SUB next to a log.
- To see a list of available line actions, enter / (slash) next to any of the logs.
- If you select multiple logs for processing, Transaction Analysis Workbench applies the values of **Rel**, **Filter**, and **Zone** of the top selected log to all selected logs.
- To organize your list of logs, leave blank lines between groups of logs and enter comments in the **Log File** column. Comments must begin with an asterisk (*). You can use comments as headings for groups of related logs. For example:

  ```
  * Daily SMF files for MVSA
  ```

- If you have organized your list of logs in a particular order, do not use the **SORT** command to sort the list; and do not select the point-and-shoot **Log File** column heading, which has the same effect. To undo a sort, and any other changes to the list, such as adding or deleting items, press the Cancel function key (F12), which leaves the panel without saving changes.

The log browser displays the records in the selected logs.

Related tasks

Processing ad hoc log files in batch

You can use the Transaction Analysis Workbench ISPF dialog to generate JCL that processes the log files in your personal ad hoc list (ISPF dialog option 4 **Process**). Using your personal ad hoc list offers a private alternative to using sessions.

Related reference

Log stream types

Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

Chapter 18. Browsing logs in a session

After creating a session and adding log files to the session, you can use the Transaction Analysis Workbench ISPF dialog to browse the logs.

You can browse logs in a session from either of the following session menu options:

- Option 5 **Investigate**, which is described in the step-by-step procedure presented here.
- Option 3 **Files**: on the **Locate and Manage Log Files** panel, enter S next to a log to browse the entire file, or `W` to browse a time slice from that file. For details on browsing a time slice, see step "4" on page 137 of the procedure presented here.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**.
2. Enter S next to the session.
3. On the session menu, select option 5 **Investigate**.

   If you have previously browsed logs in this session, a pop-up window might open, offering you the option to resume browsing where you left off. Resuming skips step "4" on page 137 of this procedure. The resume option that you select determines whether or not this window is displayed in future. To change the resume option later, go to dialog option 0.1 **Workbench Personal Settings** and set the value of the **Resume Investigate** field.

   Otherwise, the **Investigate** panel is displayed, listing the log files available for browsing.
4. Decide whether you want to browse entire logs or a *time slice*: a specific time period across one or more logs.

   - To browse entire logs, set **Time Slice** off. Either use **Time Slicing** in the action bar, or enter `SLICE OFF` on the command line. Entering `SLICE` with no parameter toggles time slicing on and off.
   - To browse a time slice:

     a. Set **Time Slice** on. Either use **Time Slicing** in the action bar, or enter `SLICE ON` on the command line.

     b. Specify the start and duration of the time slice that you want to browse, and then press Enter. To set the time slice to match the start and duration of a particular log, enter T next to the log.

        The **Coverage** column displays information about how much of the time slice, if any, each log covers.

   For large files and log streams, browsing a time slice improves performance.

   Time slicing uses a sampling method that favors speed over accuracy to locate log records in large files. The sampling method sometimes includes records that are just outside the specified time slice. When you specify a time slice, the log browser scrolls to the first log record in the time slice. If you press the Backward function key (F7), you might see records that are outside the specified time slice, but which have been included by the sampling method.

   The **Duration** column displays the length of time that each log covers. If the **Duration** column is not visible, press the Right function key (F11) to scroll the column into view.
5. Select the logs that you want to browse:

   - To select all logs, enter S on the first line under the slash (**/**) column heading.
   - To select one or more logs, type S next to each log, and then press Enter.

The log browser displays the records in the selected logs.

Related tasks

Adding log files to a session

If you know the data set names or log stream names of the log files associated with a problem, then you can add them to a session manually. Otherwise, you can use the automated file selection utility to locate the log files for a particular time interval and for a particular system, and add those log files to a session.

Registering sessions

Registering (creating) a session is the recommended first step for using Transaction Analysis Workbench to analyze a problem.

Chapter 19. Log browser navigation controls

The main log browser panel (that displays a formatted view of log browser records) contains a variety of controls for navigating logs.

Tip: For a more detailed version of the following information, move the cursor to the command line of the main log browser panel, and then press the Help function key (F1).

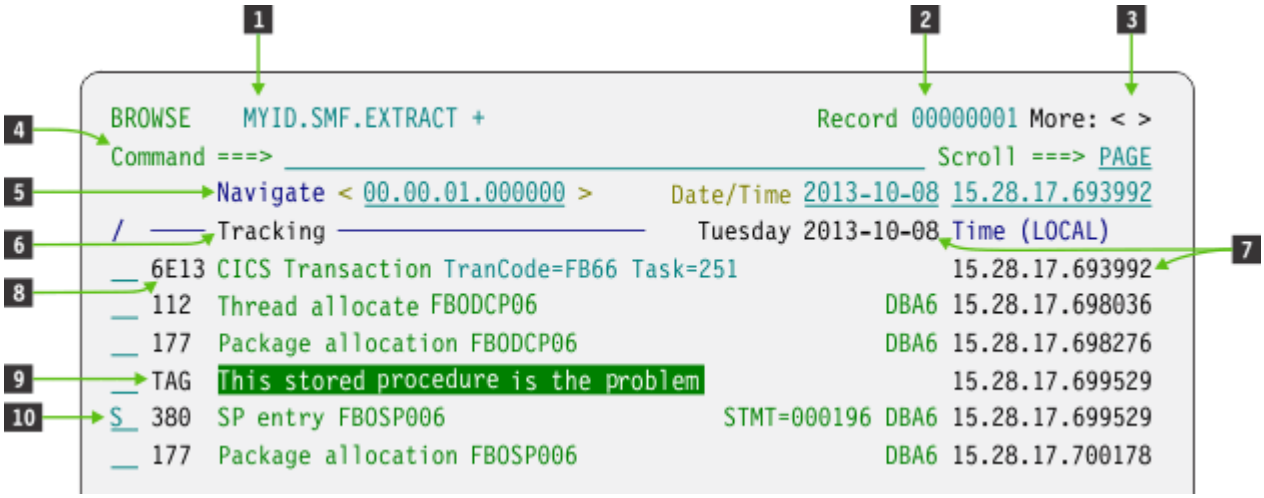


Figure 56. Main log browser panel navigation controls

1

The data set name or log stream name of the log file that you are browsing.

If you are browsing a merged view of more than one log file, this is the name of the first log file in the list that you selected. A plus sign (+) after the name indicates that you are browsing more than one log file.

Log stream names are prefixed by the log stream type followed by a colon (:). For example, SMF:IFASMF.ZOS1.MAN1.

2

The **Record** number identifies the record currently at the top of the display, starting at 1.

The record number of a particular record can change depending on the set of log files that you are browsing and the time slice that you have selected. If you have selected a time slice, then record number 1 is the record at the start of the physical disk track (or equivalent) where the slice starts. If you are browsing an entire single log file, then record number 1 is the first record in the file. If you are browsing a merged view of multiple log files, then the record numbers depend on how the records are merged in time sequence.

To scroll to a particular record number, use the **LOCATE** command: for example, `L 10504`. Then bookmark the position by creating a tag or a label.

3

To cycle through different combinations of column headings and data, press the Right function key (F11) or the Left function key (F10).

4

For details of the primary commands that you can enter on this panel, select **Help** > **Commands Help** from the action bar.

5

To scroll forwards or backwards by an amount of time, specify an amount of time between **<** and **>** in the 24-hour clock format *hh*.*mm*.*ss*.*thmiju*, and then select (move your cursor to, and then press Enter) either **<** or **>**.

To scroll to a point in time, edit the date and time displayed next to the **Date/Time** field, and then press Enter.

To scroll to the point in time already displayed next to **Date/Time**, select the **Date/Time** field. This is useful when you have edited the date and time values to scroll to a point in time, and then, later, you want to return to that point in time.

6

If filtering is on, the line above the first log record displays the word **Filtering**.

If tracking is on, the line displays the word **Tracking**.

7

The date displayed to the left of the **Time** column heading is the date of the record currently at the top of the display.

8

The **Code** column displays the log code of each log record.

To display the log type of each log record:

1. Enter **DISPLAY** on the command line. The **Display settings** window opens.
2. Select the option to display the log sequence number (LSN) with the log record type, and then press the Exit function key (F3).
3. Press the the Right function key (F11) until the **LSN** column scrolls into view.

9

If you are using a session, you can permanently bookmark a position, that you or other users can return to, by creating a tag.

10

For a list of the line actions that you can perform on a log record (such as S, to browse the details of a log record), enter / (slash) next to a log record.

Related concepts

Event time stamps versus log record time stamps

Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Related reference

Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Chapter 20. Browsing multiple records, a single record, or a single field

You can browse logs at the following levels of detail: multiple abridged records, an entire single record, or a single field in a record (also called field zoom).

When you start the log browser, it displays multiple records from the log files that you have selected. The multi-record display shows a subset of fields from each record, depending on their record type.

You can select a single record to display all of its fields, and then zoom on a single field for a description of that field and its values.

Switch to browsing a different level of detail:

- To select a single record from the multi-record display, enter S next to the record.
- To zoom on a field in the single-record display, move the cursor to the field and then press Enter.
- To exit the current level of detail, press the Exit function key (F3).

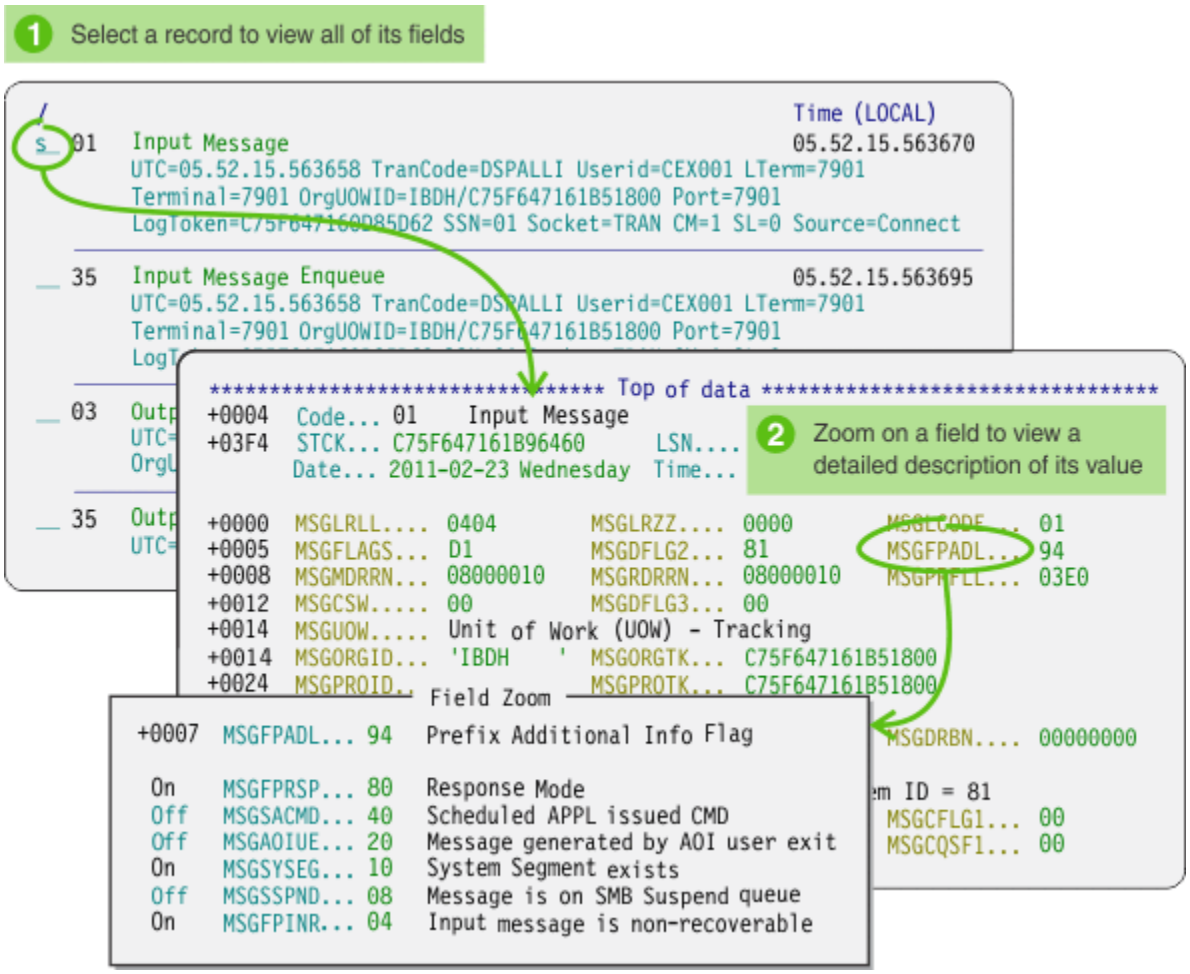


Figure 57. Browsing logs: select a record to view all of its fields, and then zoom on a field

On the multi-record display:

- To cycle through combinations of column headings and data, press the Right function key (F11).
- To switch between a view of the records formatted by Transaction Analysis Workbench and the raw record data in ISPF Browse, press the Browse function key (F4).

- To show or hide separator lines between records, or control the format of the **LSN** (logical sequence number) column, enter **DISPLAY** or select **Options** > **Display**.

On the single-record display, you can view the record in various formats.

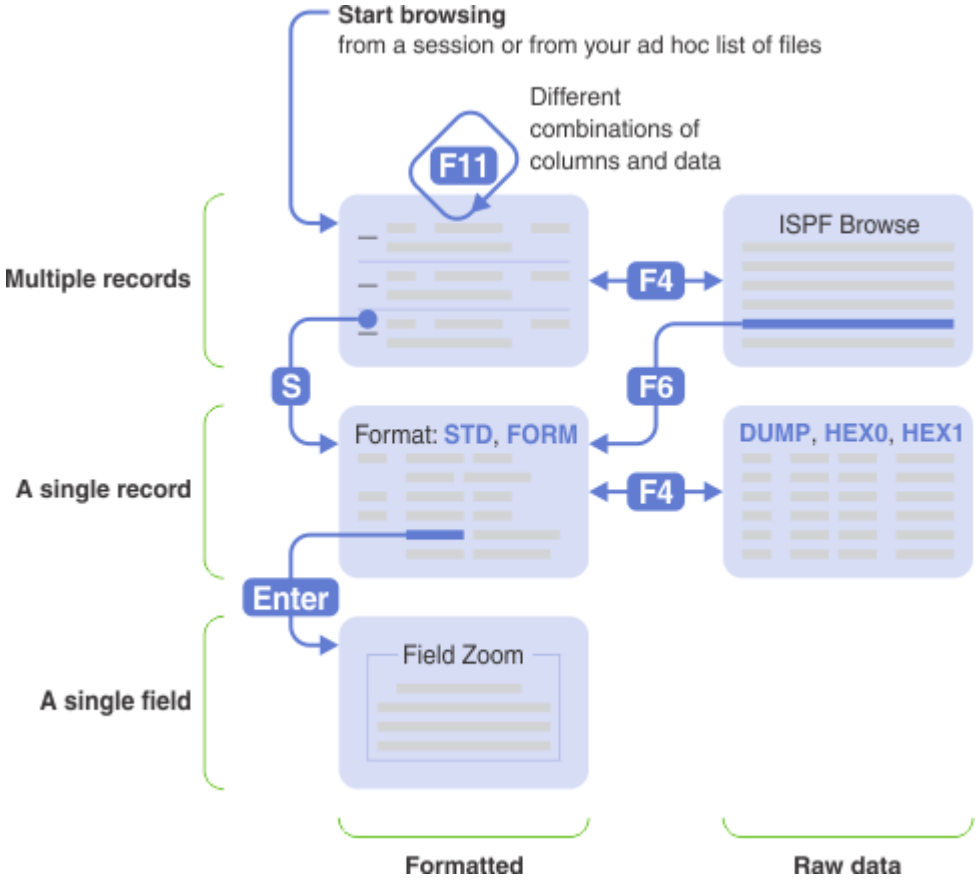


Figure 58. Browsing logs: switching between levels of detail and views

Related tasks

Formatting log records

You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

Chapter 21. Tracking a transaction or an IMS unit of recovery

Tracking displays log records that are related to a particular transaction. For IMS transactions, you can track log records that are related to a particular unit of recovery (UOR) within a transaction.

Tracking ignores any active filter, with the following exceptions:

- Tracking excludes log codes that the filter *unconditionally* excludes.

  If the filter excludes a log code, but with conditions, then tracking always displays the log records of that log code that belong to the transaction, regardless of those conditions.
- If the filter includes a log code for a trace record that belongs to a trace level higher than the current trace level, and you start tracking on a trace record at that higher level, then the tracking result set includes trace records at that higher level. While you are tracking, the trace level is effectively set to that higher level.

  For example:

  1. Suppose that you are browsing logs that include trace records, that you have set the trace level to 2, and that you have specified a filter that includes the log code for a level-3 trace record.

     The log browser displays all level-1 and level-2 trace records, and also level-3 trace records that are specifically included by the filter.
  2. You start tracking by entering line action TX next to a level-3 trace record.

     That action implicitly sets the trace level to 3 for the tracking result set.

     The tracking result set includes all level-1 and level-2 trace records that are related to the transaction. The tracking result set also includes all level-3 trace records that are related to the transaction, not just the specific level-3 trace records that are included by the filter.
  3. You press the Exit function key (F3) to exit tracking.

     The trace level returns to 2.

  Tip: To set the trace level, enter **TRACE** on the log browser command line.

On the multi-record log browser display, enter one of the following line actions next to a log record:

Option	Description
To track a transaction	Enter TX next to any log record that is related to the transaction.
To track an IMS UOR	Enter TU next to any log record that is related to the UOR.

The log browser hides records that are not related to the selected log record.

The word **Tracking** is displayed above the first log record on the panel to indicate that you are now browsing a *tracking result set* rather than the full set of log records.

To display log record time stamps relative to the first log record in the tracking result set, enter R next to the first record. To return to displaying wall-clock times, enter W next to any log record.

To create an extract containing the tracking result set, scroll to the top of data, and then enter **EXTRACT** on the command line.

To exit tracking and return to browsing the full set of log records, press the Exit function key (F3).

Changing the filter exits tracking, with the following exceptions: you can enter X or XT to next to a log record to exclude that log code or type, respectively.

Related concepts

Transaction tracking

Transaction tracking identifies the significant events in the lifecycle of a single transaction across subsystems and logs.

Event time stamps versus log record time stamps
Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Trace levels versus filters
Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Chapter 22. Log file duration not calculable (N/C)

If Transaction Analysis Workbench cannot calculate the duration of a log file, the **Investigate** panel displays the value N/C (not calculable).

To calculate the duration of a log file, Transaction Analysis Workbench reads the first record of the first track of the file and the first record of the last track of a file.

N/C indicates either of the following situations:

- The records are not in chronological order.
- The file has been reused and wrapped, so that later records in the file have an earlier time stamp than the first record.

  In this situation, the file contains two durations out of order. The time slice coverage depends on those two durations: for example, one of the durations might completely cover the time slice. The following figure illustrates a set of two IMS online data sets (OLDS), where one has wrapped.

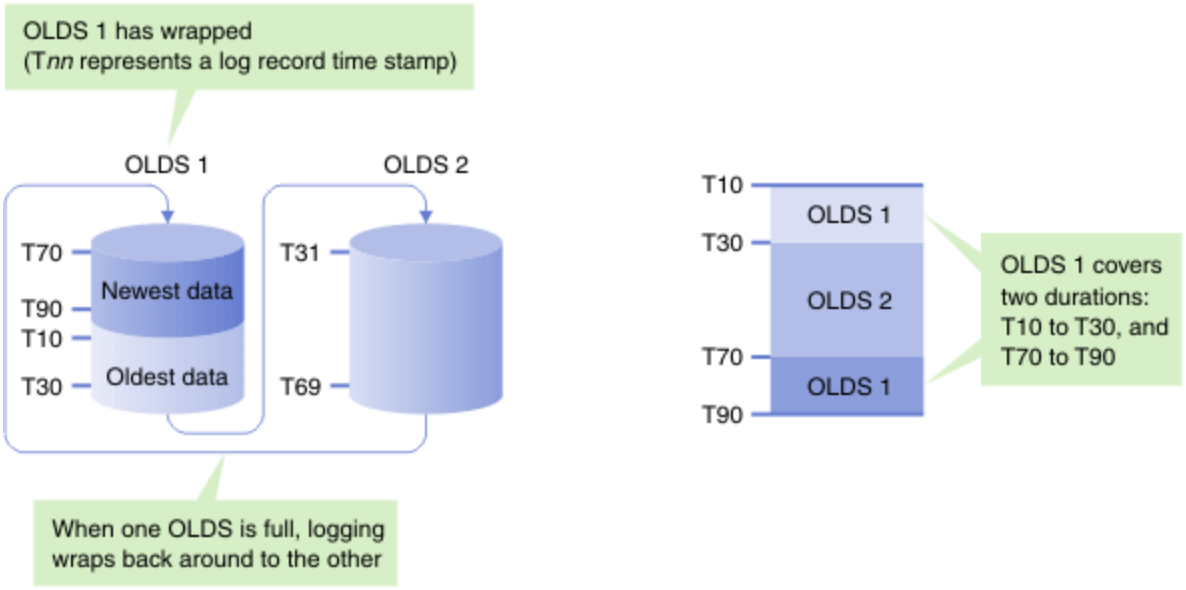


Figure 59. Log file duration N/C (not calculable) can indicate wrapping due to reuse of log files

Chapter 23. Time slice coverage

A time slice defines the start and duration of a "window" of time that you want to analyze, potentially spanning many log files. The **Investigate** panel indicates whether files provide partial or complete coverage of the specified time slice, or are completely outside (before or after) the time slice.

The following figure illustrates the concept of time slice coverage and the corresponding values displayed in the **Coverage** column on the **Investigate** panel: complete, partial, before, or after. The **Coverage** column displays these values only when time slicing is on.

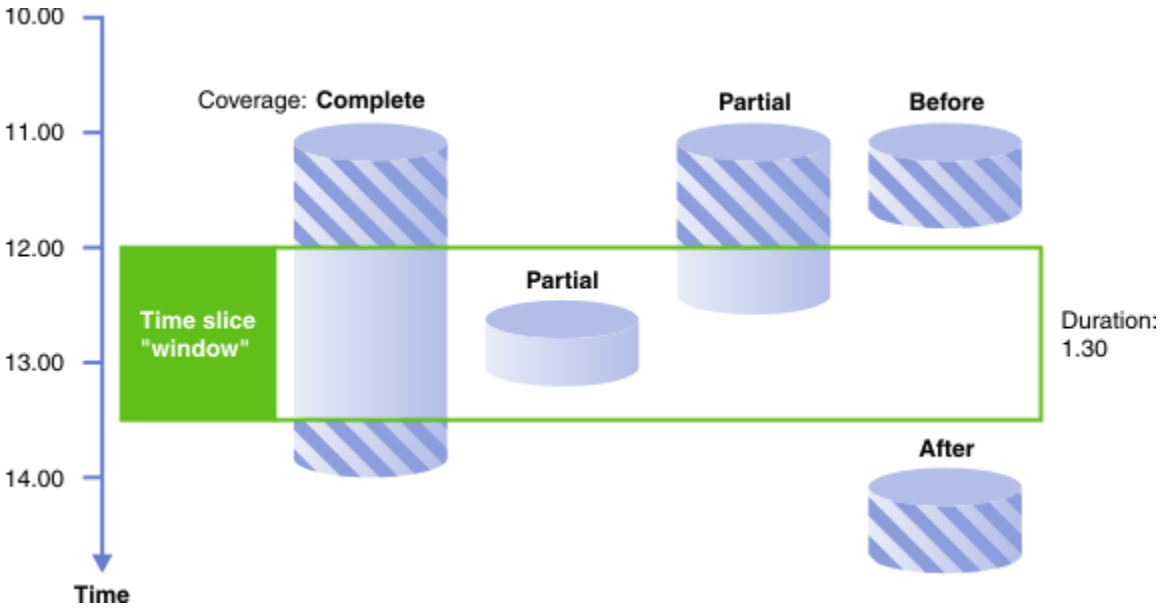


Figure 60. Time slice coverage provided by log files

If Transaction Analysis Workbench cannot calculate the coverage of a file, or cannot read a file, the **Coverage** column displays a mnemonic value with a prefix. For example:

- *NCAT indicates a log file is not cataloged, and so cannot be read.
- *LOGR indicates a problem connecting to a log stream.

To display more information about one of these values, enter S next to the affected log file.

Chapter 24. Time column display modes

The **Time** column in the log browser displays the event time stamp of each log record. You can display event time stamps in one of several modes, such as relative to the event time stamp of a selected record, or as the elapsed time between records.

To change the display mode of the **Time** column, enter one of the line actions described in the following table.

*Table 7. **Time** column display modes*

Display mode	Description	Line action
Wall-clock (default)	Displays the time stamp of each log record as a time of day	W
Relative	Displays the time of each visible log record relative to the selected record	R
Elapsed	Displays the elapsed time between each visible record	E
Nanosecond precision (relative or elapsed)	Switches the relative or elapsed display between nanosecond precision and the default microsecond precision	N

For example, to display times relative to a particular record, enter R next to that record.

Alternatively, enter the following primary command:

```
TIME mode
```

where *mode* is the same character as the line action.

Related concepts

Event time stamps versus log record time stamps

Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Chapter 25. Formatting log records

You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

1. In the log browser, enter S next to a record that you want to display.

   By default, the log browser displays the record in standard Transaction Analysis Workbench format.
2. Enter one of the following values in the **Format** field in the top-right corner of the panel:

   STD
   Standard Transaction Analysis Workbench format. This format uses knowledge modules to interpret log records: naming and organizing fields (based on DSECT mapping structures), and displaying field values according to their data type.

```
...
 Form    ===>          +     Use Form in Filter               Format ===> STD
 ******************************** Top of data ********************************
 +0004  Code... CA01  Transaction
 +0408  STCK... C62D2D082BD70621      LSN.... 0000000000000023
        Date... 2010-06-24 Thursday   Time... 15.35.01.411184.383

 +0000  LL......... 0418        ZZ......... 0000        Type....... CA
 +0005  Subtype.... 01          Vers....... ©IPI420©

 +00A4  ID......... Transaction Identification section
 +00A4  TranCode... ©MQATREQ1©  Program.... ©MQATPGM ©
 +00B4  Userid..... ©FUNTRM15©  ITerm...... ©FUNTRM15©
 +00C4  LTerm...... ©FUNTRM15©  LTermOut... ©FUNTRM15©
 +00D4  Terminal... ©SC0TCP15©  LTermOvr... ©        ©
...
```

   Figure 61. Panel: STD (standard) log record display format

   Record segments are delineated by a blank line.

   Segment or field group names have a description.

   Field names are point-and-shoot. To zoom in to see more details of a field, move the cursor to the field name, and then press Enter.

   Field contents are displayed in different formats depending on the type of field:

   - Character data is enclosed in quotes
   - Numeric data is prefixed by + for positive values or - for negative values
   - Hexadecimal data and flags are hexadecimal digits
   - Dump fields are in dump format showing the offset within the field, the hexadecimal data, and EBCDIC character translation

   FORM
   Form-based Transaction Analysis Workbench format. Same as STD, but limited to displaying the fields defined in a form: either the form that you select in the **Form** field or, if you select the **Use Form in Filter** option, the form for the currently displayed record type named in the active filter.

   DUMP
   Dump format, similar to a SYSUDUMP dump data set.

```
...                                                          Format ===> DUMP
******************************** Top of data ********************************
+0000  04180000 CA01C9D7 C9F4F2F0 00030000   *......IPI420....*
+0010  00000050 02B00001 00000300 01080001   *...&............*
+0020  00000000 00000000 00000000 00000000   *................*
+0030  00000000 00000000 00000000 00000000   *................*
+0040  00000000 00000000 00000000 00000000   *................*
+0050  C9C1C4C7 40404040 C62D2D08 2BD24F41   *IADG    F....K|.*
+0060  C9C1C4C7 40404040 C62D2D08 2BD24F41   *IADG    F....K|.*
+0070  F5F1DCB0 40000006 D02C0000 40000840   *51.. ....... .. *
+0080  0000F5F5 00000000 00000001 000001A5   *..55...........v*
+0090  000001B4 00000001 00000001 00000000   *................*
+00A0  00000000 D4D8C1E3 D9C5D8F1 D4D8C1E3   *....MQATREQ1MQAT*
+00B0  D7C7D440 C6E4D5E3 D9D4F1F5 C6E4D5E3   *PGM FUNTRM15FUNT*
+00C0  D9D4F1F5 C6E4D5E3 D9D4F1F5 C6E4D5E3   *RM15FUNTRM15FUNT*
+00D0  D9D4F1F5 E2C3F0E3 C3D7F1F5 40404040   *RM15SC0TCP15    *
+00E0  40404040 40404040 40404040 40404040   *                *
...
```

Figure 62. Panel: DUMP log record display format

HEX1

Vertical hexadecimal dump format with decimal offsets.

```
...                                                          Format ===> HEX1
******************************** Top of data ********************************
    1  ----+----1----+----2----+----3----+----4----+----5----+----6
       ......IPI420.......&.....................................
       0100C0CDCFFF000000050B000000000000000000000000000000000000000000
       4800A19794200300000020010030180100000000000000000000000000000000

   61  ----+----7----+----8----+----9----+----0----+----1----+----2
       ..................IADG    F....K|.IADG    F....K|.51.. ...
       00000000000000000000CCCC4444C2202D44CCCC4444C2202D44FFDB4000
       00000000000000000000914700006DD8B2F1914700006DD8B2F151C00006

  121  ----+----3----+----4----+----5----+----6----+----7----+----8
       .... .. ..55...........v.................MQATREQ1MQATPGM
       D200400400FF00000000000A000B0000000000000000DDCEDCDFDDCEDCD4
       0C0000800055000000010015001400010001000000004813958148137740

  181  ----+----9----+----0----+----1----+----2----+----3----+----4
       FUNTRM15FUNTRM15FUNTRM15FUNTRM15SC0TCP15
       CEDEDDFFCEDEDDFFCEDEDDFFCEDEDDFFECFECDFF4444444444444444444
       64539415645394156453941564539415230337150000000000000000000

...
```

Figure 63. Panel: HEX1 log record display format

HEX0

Vertical hexadecimal dump format with hexadecimal offsets.

```
...
                                                          Format ===> HEX0
******************************** Top of data ********************************
+0000  0---4---8---C---0---4---8---C---0---4---8---C---0---4---8---C---
       ......IPI420.......&.........................................
       0100C0CDCFFF000000050B000000000000000000000000000000000000000000
       4800A19794200300000020010030180100000000000000000000000000000000

+0040  0---4---8---C---0---4---8---C---0---4---8---C---0---4---8---C---
       ................IADG    F....K|.IADG    F....K|.51.. ....... ..
       0000000000000000CCCC4444C2202D44CCCC4444C2202D44FFDB4000D2004004
       00000000000000000914700006DD8B2F1914700006DD8B2F151C000060C000080

+0080  0---4---8---C---0---4---8---C---0---4---8---C---0---4---8---C---
       ..55...........v.....................MQATREQ1MQATPGM FUNTRM15FUNT
       00FF00000000000A000B0000000000000000DDCEDCDFDDCEDCD4CEDEDDFFCEDE
       00550000000100150014000100010000000048139581481377406453941564 53

+00C0  0---4---8---C---0---4---8---C---0---4---8---C---0---4---8---C---
       RM15FUNTRM15FUNTRM15SC0TCP15                                  ....
       DDFFCEDEDDFFCEDEDDFFECFECDFF4444444444444444444444444444444442115
       941564539415645394152303371500000000000000000000000000000000007F
...
```

Figure 64. Panel: HEX0 log record display format

Tip: To switch between the current Transaction Analysis Workbench format (STD or FORM) and the current dump format (DUMP, HEX1, or HEX0), press the Switch function key (F4).

Related tasks

Browsing multiple records, a single record, or a single field
You can browse logs at the following levels of detail: multiple abridged records, an entire single record, or a single field in a record (also called field zoom).

Using forms to exclude fields when browsing log records
You can use forms to exclude fields from display when browsing log records in the ISPF dialog.

Creating formatted record reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports that reproduce the formatting that the ISPF dialog displays when you browse an individual log record. These batch reports are known as *formatted record reports*.

Using forms to exclude fields when creating formatted record reports
When using the report and extract utility to create a formatted record report, you can optionally use forms to exclude fields. Formatted record reports reproduce the formatting that the ISPF dialog displays when you browse an individual record in the log browser.

Related reference

FORMAT command
Specifies options for presenting data in formatted record reports, matching the options that you can specify when browsing records in the ISPF dialog. Ignored for extracts, CSV, and JSON output.

Chapter 26. Highlighting log types in different colors

You can customize the color and highlighting of records according to their log type.

1. On the log browser multi-record display, enter **HILITE** or select **Options** > **Color highlighting**.

   The **Color Highlighting** panel is displayed.
2. Specify color and highlighting values.
3. Press the Exit function key (F3) to return to the log browser.

Example

The following figure shows the descriptions of DB2 trace records (log type DTR) record highlighted in pink and DB2 log records (log type DB2) in yellow.

```
  File  Mode  Filter  Time  Labels  Options  Help

FUWPRBRF  FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> _____________________________________________ Scroll ===> PAGE
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/       Tracking                            Tuesday 2013-10-08 Time (LOCAL)
__ CA01 IMS Transaction TranCode=FBOIAT41 Region=0002          17.10.09.284086
__ 01   Input Message TranCode=FBOIAT41                        17.10.09.284086
__ 35   Input Message Enqueue TranCode=FBOIAT41                17.10.09.284110
__ 08   Application Start TranCode=FBOIAT41 Region=0002        17.10.09.284366
__ 31   DLI GU TranCode=FBOIAT41 Region=0002                  17.10.09.284390
__ 5600 Sign-on to ESAF Region=0002 SSID=DBA6                 17.10.09.290476
__ 112  Thread allocate FBOIAP41                         DBA6 17.10.09.291061
__ 073  Create thread end                                DBA6 17.10.09.291130
__ 177  Package allocation FBOIAP41                      DBA6 17.10.09.291357
__ 380  SP entry FBOSP007                  STMT=001031   DBA6 17.10.09.291616
__ 177  Package allocation FBOSP007                      DBA6 17.10.09.291800
__ 061  SQL UPDATE                         STMT=000001   DBA6 17.10.09.291941
__ 0020 Begin UR                                             17.10.09.292976
__ 0600 Savepoint                                            17.10.09.292976
__ 0600 Update in-place in a data page                       17.10.09.292976
__ 058  SQL UPDATE               SQLCODE=0 STMT=000001 DBA6 17.10.09.293314
__ 065  SQL OPEN C1                        STMT=000001 DBA6 17.10.09.293405
```

Figure 65. Panel: Highlighting log records in different colors

Related reference

Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Chapter 27. Finding character strings

To search for a character string in the log browser, use the **FIND** primary command. You can use **FIND** when browsing a list of records or an individual record.

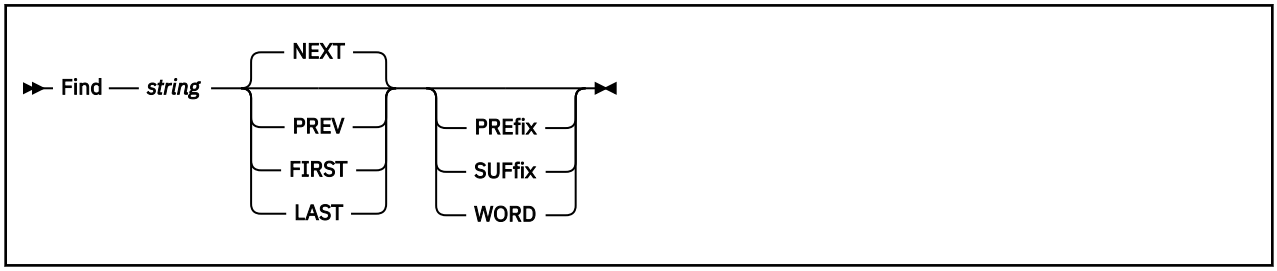
The behavior and syntax of the **FIND** command is similar to the ISPF editor **FIND** command. You can abbreviate **FIND** to **F**.

If the command finds the string, it moves the cursor to the start of the string and, if necessary, scrolls the display to bring the string into view.

To find the next occurrence of the string, press the Repeat Find function key (F5).

The search is not case sensitive; uppercase and lowercase characters are treated the same.

Format



Parameters

string
The character string that you want to find.

If the string contains embedded spaces, enclose the string in quotation marks. For example:

```
FIND 'My tag description'
```

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string*.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string*.

PREFIX
Locates *string* at the beginning of a word.

SUFFIX
Locates *string* at the end of a word.

WORD
string is delimited on both sides by blanks or other non-alphanumeric characters.

Usage

- When searching a list of log records, the **FIND** command only searches data in the current view, such as the set of columns currently displayed. Press the Left function key (F10) or the Right function key (F11) to scroll between different views, bringing **Time**, **LSN**, and other global fields into or out of view as required for your search.

- Data hidden by a filter or form is not included in the search.
- When analyzing large files, you can improve efficiency by limiting the number of records searched by the **FIND** command, known as FINDLIM. To set the find limit, enter FINDLIM on the command line or select **Options** > **Find Limit...** in the action bar.

  For example, entering **FINDLIM 99999** enables the **FIND** command to return control of the display back to the user if no match was found after 99,999 records have been searched. You can press the Repeat Find function key (F5) to recommence the search at the start of the next block of 99,999 records.

  When you search for a character string in the log browser, you can use the **TIMEOUT** command along with **FINDLIM** to improve the performance issues. **FINDLIM** and **TIMEOUT** operate together and are primarily intended to work with TX and TU tracking. You can set a time limit on waits caused by long data set scans by using the **TIMEOUT** command. You can specify a value for **TIMEOUT** between 1 and 99 seconds or 0 for no time out.

  Note: You can specify **FINDLIM 0** and **TIMEOUT 0** to process an unlimited number of records. With large log files, this command might lock the terminal for an extended period of time. You must use this setting as a temporary measure and reinstate a limit once processing is complete.
- If the **FIND** command reaches the find limit before finding the string, the panel displays the message *n* records searched. To continue searching, press the Repeat Find function key (F5).

Example

When browsing a list of log records, and you have selected a view that shows global field values for each record, the following command searches for a record that is related to the program name MQATPGM:

```
FIND PROGRAM=MQATPGM
```

This command will only find this string if you have scrolled to a view that shows Program=*name* values.

Chapter 28. Filtering log records while browsing logs

You can apply a filter to display only the log records that you want to analyze, and exclude others. For example, you can apply a filter to display only records of particular log types and codes, or records for a particular transaction code or user ID.

The following procedure describes how to define or edit the current temporary filter while browsing logs.

Alternatively, as a shortcut to editing the filter, you can enter the line actions X or XT while browsing logs:

- To hide all records of a particular log *code*, enter line action X next to one of those records. That record, and all other records with that log code, are excluded from display. For example, if you enter X next to an IMS 01 log record, then all IMS 01 log records are excluded from display.
- To hide all records of a particular log *type*, enter line action XT next to one of those records. For example, if you enter XT next to any IMS log record, then all IMS log records are excluded from display.

The X and XT line actions insert "exclude" lines in the current temporary filter. To show the excluded records again, use one of the following commands:

- To show records hidden by the X line action, enter `RESET X` on the command line.
- To show records hidden by the XT line action, enter `RESET XT` on the command line.
- Enter `FILTER` on the command line, and then delete the corresponding lines in the filter.

To apply a filter while browsing logs:

1. Press the Filter function key (F18) or enter `FILTER` on the command line.

   The **Filter** panel is displayed.
2. Specify filter criteria.

   For example, to show only IMS Message log records, enter **IMS** under the **Log** heading, and **01** under the **Code** heading.

   To select from the list of filters stored in the control repository, move the cursor to the **Filter** field, and then press the Prompt function key (F4). After selecting a filter, you can edit the filter details; editing these details affects only the current temporary filter being used by the log browser, not the filter stored in the control repository.

   Tip: To save the current temporary filter to the control repository, enter **SAVEAS** on the command line of the **Filter** panel.

   For details on defining filters, see “Defining filters” on page 396.
3. Press the Exit function key (F3) to apply the filter, and return to browsing logs.

   If no records from the current position onwards match the filtering criteria, then a message containing the following text is displayed:

   ```
   …contains no records that match the specified Filtering Criteria.
   ```

   This message applies only to records *from the current position onwards*. Depending on the filter and the data in the logs that you are browsing, pressing the Backward function key (F7) could scroll into view log records that match the filter.
4. To switch off the filter, press the Flip Filter function key (F6) or enter `FILTER OFF`.

   To switch the filter on again without changing it, press the Flip Filter function key (F6) again or enter `FILTER ON`.

   Unconditional exclude lines in the filter, such as those created by the X and XT line actions, always apply, regardless of whether the filter is on or off.

Related concepts

Trace levels versus filters

Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Related tasks

Defining filters
To select only the log records that are of interest to you, define and then use a filter. A filter specifies which log record codes to select or exclude and, optionally, more detailed conditions based on field values.

Defining forms
To hide log record fields that are not of interest to you, define and then use a form.

Chapter 29. Bookmarking log records

To bookmark log records while browsing logs, you can create either labels or tags. Labels are temporary, private bookmarks that disappear when you exit the log browser. Tags are permanent bookmarks that you can share, but require you to use a session because they are stored in a session history. Labels save only the log record position; tags also save aspects of the current browsing state such as the active filter and currently selected log files.

Bookmarking log records with permanent tags

While browsing logs in a session, you can tag a record, and then you or other users can later resume browsing at that position by selecting the tag in the session history. Resuming from a tag resumes not just the browsing position, but also aspects of the browsing state, including the filter that was active at the time the tag was created, the log files that were selected, and the time slice (if any).

To tag records, you must be browsing logs in a session, under dialog option 1 **Sessions**. Tags are saved in the history for each session, which is saved in the session repository. You cannot use tags while browsing logs in your personal ad hoc list of log files, under dialog option 4 **Process**.

A tag consists of the following information:

- A note (a multi-line comment about the tag)
- Details of the log record attached to the tag
- The data set or log stream names of all currently selected log files
- Active filter
- Time slice (if any)

In addition to tags that users have explicitly created while browsing logs, each session history contains a tag for the most recent browsing position of each user, enabling you to resume browsing where another user has left off.

1. **To tag a record** while browsing logs, enter G next to the record.

   A blank note for the tag is displayed in the ISPF editor.
2. Enter at least one line describing the tag.

   The log browser and the session history use the start of the first line of the note as the tag description.
3. Press the Exit function key (F3) to save the tag and return to the log browser.

   The tag is displayed on a line above the tagged record.

   Tips:

   - To edit the note for a tag, enter S next to the tag.
   - To delete a tag, you must exit the browser, go to the session history, and then enter D next to the tag. You cannot delete a tag while browsing logs.
   - The log browser displays all tags for a session, regardless of the currently selected log files or their time interval. Tags that are attached to log records outside the time interval of the currently selected log files are displayed above or below the log records of the currently selected log files. Tags that are attached to log records that fall inside the time interval of the currently selected log files, but do not belong to the currently selected log files, are displayed merged in time sequence.
4. **To resume browsing** at the tagged browsing position, you or other users can go to the session history, and then enter R next to the tag.

Related tasks

Session history: writing notes, resuming browsing from tags, and reviewing jobs

Each session has a history that consists of jobs for workflow tasks that have been run, notes that you or other users have written about the session, and tags that bookmark a position in the log browser.

Bookmarking log records with temporary labels

You can label a record while browsing logs, and then refer to the label to return to the record. Labels are temporary private bookmarks that disappear when you exit the log browser. To create permanent bookmarks that you can share, use tags instead of labels.

1. To label the record that is currently at the top of the log browsing panel, enter a period (.) followed by a 1- to 7-character label on the command line.

   For example, enter the following on the command line:

   ```
   .MYTRAN
   ```

   To label any other record on the panel, enter line action L next to a record. This automatically generates a label consisting of the letter Z followed by a number (for example, Z000001).

   The label is displayed in the **Description** column, overlaying the log record description.

   To show or hide labels, enter LABELS ON or LABELS OFF, or select **Labels** > **Set Label display ON|OFF** in the action bar. To toggle between showing and hiding labels, enter LABELS with no parameter.

   You can assign multiple labels to a single record, but the label display shows only one label per record.
2. To return to a labelled record, either:

   - Enter the **LOCATE** command (or its abbreviations, LOC or L), without any parameters. A window opens, listing the labels. Select the label that you want.
   - Enter LOCATE *label* (without the leading period that you used to assign the label). For example:

     ```
     L MYTRAN
     ```

     If you assigned a numeric label to a record, then you must include the leading period; otherwise, the **LOCATE** interprets the number as a record number, not a label. For example:

     ```
     L .999
     ```

   The log browsing panel scrolls to show the selected labelled record at the top.

   Tips:

   - As a quicker alternative to labels, enter LOCATE *record_number* to scroll to a particular record, where record number 1 is the first record in the log file, or time-sliced set of merged log files, that you are browsing. For example (note the absence of a leading period before the record number):

     ```
     L 999
     ```

   - To delete a label, enter LOCATE, and then enter D next to the label.
   - To delete all labels, enter LABELS RESET, or select **Labels** > **Reset labels** in the action bar.

Chapter 30. Saving records displayed in the log browser to an extract file

You can create an extract file containing records displayed in the log browser from the current scroll position to the bottom of data marker, or a selected block of records.

The extract contains only the records currently displayed in the log browser. For example, if the log browser is displaying a tracking result set, then the extract contains only those records in the tracking result set; if a filter is set, then the extract contains only those records that match the filter.

1. On the multi-record log browser display, use one of the following methods to extract the records that you want:

Option	Description
To extract records from the record at the top of the current scroll position to the bottom of data marker	Enter **EXTRACT** on the command line.
To extract all records, from the top of data to the bottom of data marker	a. Scroll to the top of data: type **M** on the command line and then press the Backward function key (F7). b. Enter **EXTRACT** on the command line.
To extract a block of records	a. Type CC next to the first record in the block. b. Type CC next to the last record in the block. c. Enter **EXTRACT** on the command line.

   The **Extract Request** window opens.
2. Edit the extract data set details and then press Enter.

   The log browser performs the extract request immediately, as a foreground process, rather than displaying JCL for you to submit.

The following figure demonstrates extracting a block of records:

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    MY.CICS.SMF +                              Record 00000001 More: < >
Command ===> extract                                          Scroll ===> PAGE
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 15.28.17.693992
/                                           Tuesday 2013-10-08 Time (Relative)
cc 6E13 CICS Transaction TranCode=FB66 Task=251                 15.28.17.693992
__ AP   3250 D2D2  ENTRY DB2_API_CALL                    00251       +0.003413
__ AP   3263 D2D2  EVENT ABOUT_TO_ISSUE_DB2_CREATE_THREA 00251       +0.003576
__ 112  Thread allocate FBODCP06                          DBA6       +0.004043
__ AP   3264 D2D2  EVENT RETURN_FROM_DB2_CREATE_THREAD_F 00251       +0.004129
__ AP   326C D2D2  EVENT ABOUT_TO_ISSUE_DB2_API_REQUEST  00251       +0.004141
__ 177  Package allocation FBODCP06                       DBA6       +0.004283
__ AP   326D D2D2  EVENT RETURN_FROM_DB2_API_REQUEST     00251       +0.004489
__ AP   3251 D2D2  EXIT  DB2_API_CALL/OK                 00251       +0.004504
__ AP   3181 D2EX1 EXIT  APPLICATION-REQUEST             00251       +0.004517
__ AP   3250 D2D2  ENTRY DB2_API_CALL                    00251       +0.005369
cc AP   326C D2D2  EVENT ABOUT_TO_ISSUE_DB2_API_REQUEST  00251       +0.005381
__ 380  SP entry FBOSP006                    STMT=000196 DBA6       +0.005536
__ 177  Package allocation FBOSP006                       DBA6       +0.006185
__ 0020 Begin UR                                                     +1.510247
__ 0600 Savepoint                                                    +1.510263
__ 0600 Update in-place in a data page                               +1.510263
__ 380  SP exit  FBOSP006           SQLCODE=0 STMT=000196 DBA6       +1.510898
```

Figure 66. Panel: Creating an extract file of a block of records displayed in the log browser

You can use the resulting extract as an input log file for Transaction Analysis Workbench.

Related tasks

Creating extracts of log files in a session
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Creating extracts of log files in your ad hoc list
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Part 5. Creating session templates

Subject-matter experts create session templates to offer other users blueprints for analyzing particular types of transaction rather than starting with a blank session. If users create a session from a template, then the new session inherits details, systems, files, and tasks from the template.

If you want to create a session template that contains tasks to perform automated file selection, then you must first create system definitions that contain the details necessary for automated file selection. For example, if you want to create a session template that contains a task to automatically select DB2 log files, then you must first create a DB2 system definition that refers to the DB2 bootstrap data set, among other details.

You can either start with a new, blank session template or you can save an existing session as a template.

Related concepts

Sessions, workflows, and templates
Transaction Analysis Workbench offers an optional framework for analyzing problems in *sessions*. A session encapsulates information about a problem and the analysis of that problem. This information can include the time period in which a problem occurred, the systems involved, and the corresponding log files.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Chapter 31. Creating session templates starting with a blank template

Creating a session template starting with a blank template is the quickest way to create a basic template that specifies systems and contains tasks to perform automated file selection and extraction for those systems. Such a template enables users to create a session for a particular type of transaction and then, without assistance from subject-matter experts, select the log files, and create extracts of those log files for the time when the session problem occurred. If you have already created the necessary system definitions, then creating a basic template typically takes less than two minutes.

To create a new, blank session template:

1. On the Transaction Analysis Workbench Primary Option Menu, select option 2 **Controls**.
2. On the **Controls Menu**, select option 3 **Session Templates**

   The **Session Templates** panel is displayed.
3. Enter **NEW** on the command line.

   The **New Session Template** window opens.
4. Enter a name for the new template.

   The **Template Details** panel is displayed for the new template.
5. Enter the template details, including the systems involved.

   The details for a template are the same as the details for a session, except that a template does not specify a particular time period for a problem.

   When you create a session from a template, the session inherits all of the template details except the description.

   Tip: If the systems involved run in a different time zone than the system running Transaction Analysis Workbench, then replace the default LOCAL value of the **Zone** field with the time zone of those systems.
6. Press the Exit function key (F3) to save the template.

1. Create workflow tasks.

   If you specified systems on the **Template Details** panel, then, when you exit the panel, the **Auto Task Creation** window opens, enabling you to create automated file selection and extraction tasks for those systems.

   You can also use the following options on the template menu to create tasks:

   - To create tasks that generate batch reports, select option 4 **Reporting**.
   - To create tasks that select log files, create extracts, or perform your own custom JCL, select option 2 **Workflow**.
2. If the systems involved write to log files with fixed names, making automated file selection tasks unnecessary, or the systems write to log files that cannot be selected automatically, then use template menu option 3 **Files** to add the files to the template. Sessions created from the template will inherit those files.
3. Test the template.

   In a template, batch job tasks contain only *unresolved JCL*, with substitution variables to represent values such as the session date and time. To test a template, you need to create a session from the template. In a session, Transaction Analysis Workbench resolves those variables, replacing them with actual values.

   Create a session from the template and then run the tasks. For example, to run all of the tasks in sequence, select 2 **Workflow** from the session menu, and then enter the command **SCHED**

Tip: If a batch job task fails because of issues with the unresolved JCL inherited from the template, then you can use the following method to iteratively edit and test the JCL without creating a new session each time: on the session **Tasks** panel, enter line action SU next to the task and then edit the JCL. The SU line action displays the *unresolved* JCL, before Transaction Analysis Workbench resolves substitution variables. After editing the JCL, resubmit the task.

Chapter 32. Creating session templates from existing sessions

Creating a session template from an existing session enables you to build and iteratively test a workflow in a session before saving it to a session template. Working with a session is similar to working with a template. However, you can only submit batch job tasks from a session, not from a template.

Create a session that you want to use as a template. Typically, this involves creating a session that contains workflow tasks. Use the following options on the session menu to create tasks:

- To create tasks that select log files, create extracts, or perform your own custom JCL, select option 2 **Workflow**.
- To create tasks that generate batch reports, select option 4 **Reporting**.

In a template, batch job tasks contain only *unresolved JCL*, with substitution variables representing values such as the session date and time. In a session, Transaction Analysis Workbench resolves those variables, replacing them with actual values.

Tip: To edit unresolved JCL in a session: on the session **Tasks** panel, enter line action SU next to the task.

To save a session as a template:

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**.
2. Enter line action T next to the session.
3. Enter a name for the new template.

Chapter 33. Workflow task JCL substitution variables

The JCL for workflow tasks can contain substitution variables that are specific to Transaction Analysis Workbench.

Transaction Analysis Workbench resolves variables at either *JCL generation time* or *run time*, depending on the character that prefixes the variable name:

%
: Transaction Analysis Workbench resolves variables that are prefixed by a percent sign (%) at JCL generation time: when you select a JCL task from the list of tasks in a workflow (to submit the task) and, equivalently, when a task scheduler job processes a **TASK** command.

+
: Transaction Analysis Workbench resolves variables that are prefixed by a plus sign (+) at run time: when a running batch program supports them. For example, when the FUWBATCH program runs with the **SUBMIT** parameter, to save task output.

All variables can be prefixed by % and resolved at JCL generation time. Only some variables can be prefixed by + and resolved at run time.

Table 8. Workflow task JCL substitution variables

Source	Variable	Description	Can be resolved at run time?
User profile	JOBCARD	User job card. This variable can optionally be preceded by // starting in column 1; for example, //%JOBCARD. The remainder of the skeletal line containing this variable is ignored.	Yes
	PREFIX	For ISPF dialog users: the data set name prefix in your TSO profile. If you do not use a prefix (PROFILE NOPREFIX) then Transaction Analysis Workbench resolves %PREFIX to an empty string and omits the trailing period after %PREFIX, if specified. For plug-in users: the data set name prefix in the FUWVARS configuration member. If you do not use a prefix (no PREFIX specified in FUWVARS) then Transaction Analysis Workbench resolves %PREFIX to an empty string and omits the trailing period after %PREFIX, if specified.	
	FUWLINK	Data set name of the Transaction Analysis Workbench load library. The default low-level qualifier is SFUWLINK. This variable must be specified in the STEPLIB DD statement of JCL skeletons that run Transaction Analysis Workbench programs (such as the report and extract utility, FUWBATCH).	Yes
	FUWPROBR	Data set name of the Transaction Analysis Workbench session repository.	Yes
	FUWSYSDF	Data set name of the system definition repository that contains CICS, DB2, and related systems. Used for SMF file selection.	Yes
	IPILINK	Data set name of the IMS Performance Analyzer load library. The default low-level qualifier is SIPILINK. This variable must be specified in the STEPLIB DD statement of JCL skeletons that run IMS Performance Analyzer.	Yes
	CPALINK	Data set name of the CICS Performance Analyzer load library. The default low-level qualifier is SCPALINK. This variable must be specified in the STEPLIB DD statement of JCL skeletons that run CICS Performance Analyzer.	Yes

Table 8. Workflow task JCL substitution variables (continued)

Source	Variable	Description	Can be resolved at run time?
System environment	JOBNAME	Job name. When resolved at JCL generation time, this is the job name of the TSO user.	Yes
	JOBID	Job name. When resolved at JCL generation time, this is the job ID of the TSO user.	Yes
	SYSNAME	System name (SMF ID).	Yes
	USERID	RACF® (or equivalent security product) user ID.	Yes
Date and time	DATE	Current date in the format D*yymmdd*	Yes
	DATEISO	Current date and time in the format *yyyy-mm-dd-hh.mm.ss.thmiju*	Yes
	TIME	Current time in the format T*hhmmss*	Yes
Session repository: session details	PID	Session key. When this variable is used to generate a data set name, then the leading zero is replaced with the letter P to conform to data set naming conventions; for example, P0000678.	
	PSDATE	Start date in the format *yyyy-mm-dd*	
	PSTIME	Start time in the format *hh.mm.ss.th*	
	PEDATE	End date.	
	PETIME	End time.	
	FILE	Selection criteria for session files that are to be used as input to the task.	
	FUWINDD	DDname. Substituted with the appropriate ddname for the system and file type according to the FILE variable that is on the same line of unresolved JCL.	
	LSNAME	Log stream name, when the input log file is a log stream.	
	SMFIN	SMF input source for the CICS Performance Analyzer `IN(%SMFIN)` operand; resolves to either a ddname, such as SMFIN001, or an SMF log stream name.	
Session repository: tasks	TASKID	Task uniqueness identifier consisting of a character string of 16 hexadecimal digits. For example, `TASK=%TASKID` resolves to `TASK=CB4E02B128F88F06`. This identifier enables a job to read output from the JES spool, and then attach that output to the workflow task that submitted the job, so that you can later select the task and view the output, as you would using SDSF.	

Table 8. Workflow task JCL substitution variables (continued)			
Source	**Variable**	**Description**	**Can be resolved at run time?**
Session repository: files	IMSVRM	IMS version; required for IMS-related input only. IMSVRM=%IMSVRM replaces EXEC PARM='V*vvr*'; this allows %FILE processing to establish the VRM, then substitute it later in the JCL command input.	
	SID	When resolved at JCL generation time: the owning system ID of the log file. When resolved at run time: the job name of the job output being saved.	Yes
	STYP	When resolved at JCL generation time: the type of system that created the log file. When resolved at run time: the job ID of the job output being saved.	Yes
	FTYP	When resolved at JCL generation time: the type of log file. When resolved at run time: the character string SAVE	Yes
	XTYP	When resolved at JCL generation time: the type of extract: EXTRACT, CSV, or INDEX. Truncated to X, C, or I if the resolved data set name would otherwise exceed 44 characters. When resolved at run time: the character string OUTPUT	Yes

User profile variable values

The values of user profile variables depend on what you use to resolve the variables:

ISPF dialog
Gets the values from your personal ISPF profile settings.

To set the values of these variables (except for PREFIX), use Transaction Analysis Workbench ISPF dialog option 0.1 **Workbench Personal Settings**.

Task scheduler
Gets the values from the data set defined by the FUWVARS ddname in the task scheduler batch job.

Plug-in
Gets the values from the FUWVARS configuration member of the Transaction Analysis Workbench control library for Common Services Library server.

The task scheduler and the plug-in can refer to the same member.

Related reference

FUWVARS: Workflow task JCL substitution variables
The FUWVARS member of the Transaction Analysis Workbench control library defines substitution variables that the plug-in uses to resolve workflow task JCL.

Task scheduler

The task scheduler manages the execution of some or all of the tasks in a session workflow, allowing you to perform a workflow without user intervention. The scheduler runs as a batch job, submitting the tasks in order, stopping if a task fails.

Resolution of JCL substitution variables

How Transaction Analysis Workbench resolves a JCL substitution variable depends on the variable and, in some cases, the context in which the variable is used.

Replace variable with its value
This method applies to most variables. Transaction Analysis Workbench resolves the variable simply by replacing the variable with its value.

Variable names can be terminated by any character. For example, %USERIDx resolves to JOHNx

If the variable name is terminated by two trailing periods, one period is removed. For example, %USERID..MY.JCL (note the two consecutive periods after %USERID) and %USERID.MY.JCL both resolve to JOHN.MY.JCL

Any blanks after the variable name are kept. For example:

```
 %USERID is a boy
```

resolves to:

```
JOHN is a boy
```

% and + are treated as normal characters when they do not prefix a known variable name. For example, %PERCENT and +PLUS remain unchanged.

Omit DD statement if variable that specifies the data set name is blank
This method applies only to data set name variables that are used in a DD statement.

This method is the same as the previous method, except that when the variable is null, Transaction Analysis Workbench omits the DD statement.

For example, given the following line of unresolved JCL:

```
//STEPLIB DD DISP=SHR,DSN=%FUWLINK
```

If the Transaction Analysis Workbench load library data set name is defined in your profile, then the resolved JCL contains the DD statement with the data set name:

```
//STEPLIB DD DISP=SHR,DSN=FUW.SFUWLINK
```

Otherwise, the DD statement is omitted; the Transaction Analysis Workbench load library is assumed to be link-listed.

Replace line containing variable with one or more lines of JCL
This method applies only to JOBCARD and FILE. Transaction Analysis Workbench replaces the line containing the variable with one or more lines of JCL.

Replace variable with a generated value, depending on context
This method applies only to FUWINDD. Transaction Analysis Workbench replaces FUWINDD with a ddname according to the FUWBATCH program convention for input log files.

FILE: JCL substitution variable for input log files

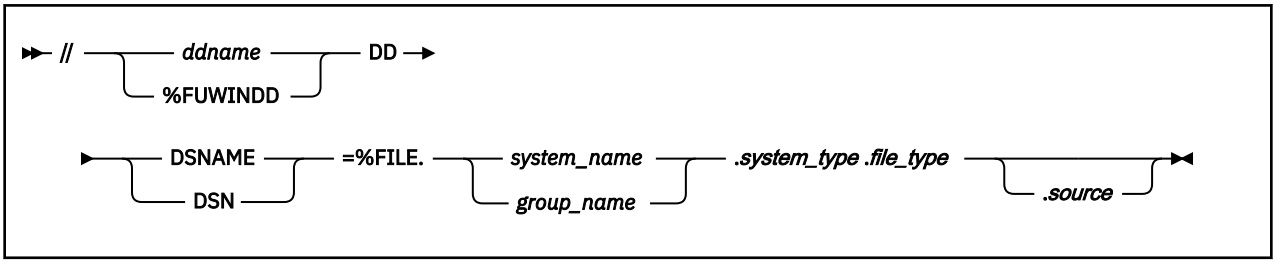
The FILE variable selects input files from the log files that have been added to a session. The FILE variable selects files based on the following criteria: system or group name, system type, file type, and source.

Specifying the FILE variable in a DD statement instructs the workflow task JCL generation process to replace that DD statement in the unresolved JCL with the following lines in the resolved JCL:

- DD statements for the selected data sets
- **LOGSTREAM** commands, in the SYSIN in-stream data set, for selected log streams

Syntax

The parameters immediately following the FILE variable name specify criteria that select log files from the session.



ddname
A fixed ddname, such as SYSUT1, or a reference to the substitution variable FUWINDD.

system_name
The name of the system that wrote the log file.

group_name
The name of a group containing systems that wrote the log file. All systems in this group are eligible for selection.

system_type
The type of system that wrote the log file.

file_type
The type of log file.

source
Optional. If specified, limits the files inserted depending on how the files were added to the session:

EXTRACT
Extract data set

INDEX
Transaction index

MANUAL
File manually entered in the list of session files

ORIGINAL
Original log file that was located by automated file selection

If omitted, all sources are considered in the order of precedence of the items in the preceding list.

If no session files match the selection criteria, then the JCL remains unresolved, causing a JCL error if submitted.

A single FILE variable can select multiple files.

Do not extend the DD statement containing the FILE variable across multiple lines of JCL. The remainder of the skeletal JCL line containing the FILE variable is ignored, except for the ddname.

FUWINDD: JCL substitution variable for input log file ddnames

If you specify %FUWINDD as the ddname for the DD statement containing the FILE variable, then the resolved JCL contains ddnames appropriate to the system type and file type specified by the FILE variable:

Table 9. DDnames generated by the FUWINDD JCL substitution variable

System type	System type description	File type	File type description	DDname
Any system types that write to SMF		SMF	SMF	SMFIN*nnn*
DB2	DB2 subsystem	NTH	Tivoli OMEGAMON XE for DB2 Performance Expert near-term history	NTHIN*nnn*
IMS	IMS subsystem	LOG	IMS log	LOGIN*nnn* or L*xxxxnnn*, where *xxxx* is an IMS subsystem name
IMS	IMS subsystem	File types other than LOG; for example, MON	For example, IMS monitor	LOGIN*nnn*
CONNECT	IMS Connect system	CEX	IMS Connect Extensions journal	CEXIN*nnn*
Any system types that write to GTF		GTF	Generalized trace facility	GTFIN*nnn*
Any system type		INPUT	"Other": combinations of system type and file type not specified in this table	INPUT*nnn*

The generated ddnames have a 3-digit numeric suffix *nnn*. For example, SMFIN001, SMFIN002.

Example 1: using the FUWINDD variable to generate ddnames

Unresolved JCL:

```
//%FUWINDD DD  DSN=%FILE.CICSP1.CICS.SMF.EXTRACT
```

The FILE variable selects SMF extracts for the CICS system CICSP1.

Resolved JCL:

```
//SMFIN001 DD  DISP=SHR,DSN=MY.FUW.CICSP1.CICS.SMF.EXTRACT1
//SMFIN002 DD  DISP=SHR,DSN=MY.FUW.CICSP1.CICS.SMF.EXTRACT2
```

In this example, the session contains two files that match the selection criteria.

The unresolved JCL specified the FUWINDD variable rather than a fixed ddname, so each file in the resolved JCL has its own ddname.

Example 2: using a fixed ddname

Unresolved JCL:

```
//SMFIN    DD  DSN=%FILE.CICSP1.CICS.SMF.EXTRACT
```

Resolved JCL:

```
//SMFIN    DD  DISP=SHR,DSN=MY.FUW.CICSP1.CICS.SMF.EXTRACT1
//         DD  DISP=SHR,DSN=MY.FUW.CICSP1.CICS.SMF.EXTRACT2
```

The unresolved JCL specified a fixed ddname, so the resolved JCL concatenates the two selected files in that single ddname.

Example 3: SMF log stream

Unresolved JCL:

```
//SMFIN    DD  DSN=%FILE.CICSP1.CICS.SMF.ORIGINAL
```

If FILE selects an SMF log stream, then, instead of a DD statement, the resolved JCL contains a **LOGSTREAM** command in the SYSIN in-stream data set.

Resolved JCL:

```
//SYSIN    DD  *
LOGSTREAM SMF:IFASMF.ZOS1.SMF.MAN1
```

Related reference

System and file types
System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

Part 6. Processing ad hoc log files in batch

You can use the Transaction Analysis Workbench ISPF dialog to generate JCL that processes the log files in your personal ad hoc list (ISPF dialog option 4 **Process**). Using your personal ad hoc list offers a private alternative to using sessions.

Related tasks

Browsing logs in your personal ad hoc list

Using your personal ad hoc list of logs offers a private alternative to using sessions. There might be occasions when you want to browse logs without registering a session. Simply add logs to your personal ad hoc list, and then select them for browsing.

Chapter 34. Creating formatted record reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports that reproduce the formatting that the ISPF dialog displays when you browse an individual log record. These batch reports are known as *formatted record reports*.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.

   Your personal ad hoc list of log files is displayed.
2. If the list does not already include the log files that you want to process, insert those log files now.

   For details on inserting log files on this panel, see Chapter 17, "Browsing logs in your personal ad hoc list," on page 135.

   You can create a report either from one or more data sets, or from a single log stream. You cannot create a report from a mix of data sets and log streams, or from multiple log streams.
3. Type SUB next to either a single log stream or one or more data set names, and then press Enter.

   The **Submit Batch Request** window opens.
4. Optionally, select a filter to limit the records included in the report.
5. Specify a report interval.

   When reporting on data sets, the interval is optional: if you do not specify an interval, all records of the data sets are included (unless filtered). When reporting on a log stream, you must specify at least the start of the interval.
6. Type / next to **Report**.
7. Select one or more report formats.

   If you are reporting on an OPERLOG, the report formatting options are ignored: instead, the generated JCL uses the **OPERLOG** parameter of the **REPORT** command to request an OPERLOG report.
8. Press Enter to generate the JCL for the request.

   The JCL is displayed in the ISPF editor.
9. On the ISPF editor command line, enter **SUB** to submit the job.
10. Use your site-specific procedures to view the job output.

Related tasks

Formatting log records

You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

Related reference

REPORT command for formatted record reports

Requests a formatted record report. Formatted record reports reproduce how the ISPF dialog log browser displays log records.

Chapter 35. Creating extracts of log files in your ad hoc list

Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

You can use other methods to create extracts. For example:

- You can create extracts of log files in a session.
- While browsing logs, you can use the **EXTRACT** command to create an extract of the displayed log records.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.
2. Type SUB next to one or more log files.

   The **Submit Batch Request** window opens.
3. Optionally, specify filtering criteria and an interval to extract only matching records.
4. Select the **Extract** option.
5. Specify the data set name of the extract that you want to create.

   The panel suggests a data set name based on a template that you specify in ISPF dialog option 0.3 **Extract Data Set Allocations**.
6. Press Enter to generate the request JCL.

   The JCL is displayed in an edit panel.
7. Submit the job: enter **SUB** on the command line.
8. Press the Exit function key (F3) to exit the edit panel.
9. Use your site-specific procedures to view the job output.

Related tasks

Creating extracts of log files in a session
Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Saving records displayed in the log browser to an extract file
You can create an extract file containing records displayed in the log browser from the current scroll position to the bottom of data marker, or a selected block of records.

Related reference

EXTRACT command
Writes the selected log records to an extract data set.

Chapter 36. Creating CSV files from log files in your ad hoc list

You can use Transaction Analysis Workbench to save data from any supported log record type to a comma-separated values (CSV) file. You can use the CSV file with other applications, such as PC-based spreadsheet applications. A CSV file created by Transaction Analysis Workbench contains selected fields from a single log record type and code.

Before using the Transaction Analysis Workbench ISPF dialog to create a CSV file from log files in your ad hoc list, you must define a form for the corresponding log record type and code.

The procedure described here uses the Transaction Analysis Workbench ISPF dialog to generate JCL that invokes the **CSV** command of the Transaction Analysis Workbench report and extract utility.

The ISPF dialog generates JCL that uses only a subset of the features of the **CSV** command. To use other features of the **CSV** command, edit the generated JCL or write your own JCL.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.

   Your personal ad hoc list of log files is displayed.
2. If the list does not already include the log files that you want to process, insert those log files now.

   For details on inserting log files, including log streams, on this panel, see Chapter 17, "Browsing logs in your personal ad hoc list," on page 135.

   You can create a CSV file either from one or more data sets, or from a single log stream. You cannot create a CSV file from a mix of data sets and log streams, or from multiple log streams.
3. Type SUB next to one or more log files, and then press Enter.

   The **Submit Batch Request** window opens.
4. Optionally, specify filtering criteria and an interval to extract only matching records.
5. Select **CSV**, and then specify the details of the CSV file, including the name of the form that specifies the type of log record and the fields that you want to save to the CSV file.

   The panel suggests a data set name based on a template that you specify in ISPF dialog option 0.3 **Extract Data Set Allocations**.
6. Press Enter to generate the JCL for the request.

   The JCL is displayed in an edit panel.
7. Submit the job: enter **SUB** on the command line.
8. Press the the Exit function key (F3) to exit the edit panel.
9. Use your site-specific procedures to view the job output.

Related tasks

Defining forms
To hide log record fields that are not of interest to you, define and then use a form.

Using forms to specify the columns in CSV files
When using the ISPF dialog to create a comma-separated values (CSV) file, you must specify a form name. The form identifies the type of log record and which fields to write to the CSV file.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Part 7. Forwarding logs to analytics platforms

You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

To convert logs to CSV or JSON, use the **CSV** or **JSON** command of the Transaction Analysis Workbench report and extract utility.

The **CSV** and **JSON** commands can write CSV or JSON data to MVS data sets, z/OS UNIX files, or TCP sockets. Writing to a TCP socket is also known as *streaming*. To stream data, you combine the **CSV** or **JSON** command with a **STREAM** command.

To run these commands, you can either write JCL yourself or use option 5 **Analytics** of the Transaction Analysis Workbench ISPF dialog to create JCL for you.

The dialog option offers only a small subset of the log record types supported by Transaction Analysis Workbench:

- Several of the most common and useful SMF record types.
- IMS transaction index records, which consolidate multiple IMS log records into one record per transaction.

Furthermore, the JCL created by the dialog extracts only a subset of fields from those records.

You can tailor the JCL created by the dialog, or write JCL yourself, to extract *any* of the fields from *any* of the log record types supported by Transaction Analysis Workbench.

ISPF dialog option 5 **Analytics** offers suboptions for different forwarding methods and destinations:

1 Stream
: Creates JCL that streams log data in JSON Lines format over a TCP network, optionally with security (SSL/TLS). Use this option for Splunk, Elastic, and any other destination that can be configured to listen on a TCP port for JSON Lines from Transaction Analysis Workbench.

2 DB2
: Creates JCL that extracts log data in CSV format, and then loads the data into a DB2 table.

3 Data Set
: Creates JCL that converts logs to either CSV or JSON format, and then saves the output without forwarding over a network or loading into a DB2 table. Use this option for testing CSV or JSON output from Transaction Analysis Workbench before forwarding.

Remember: When using Transaction Analysis Workbench, it is essential to note that the input logs you provide may contain sensitive information. Therefore, taking the necessary steps to protect any data you forward from these logs is crucial, just as you would secure the input log data.

You must ensure that all access to the data you forward is restricted to authorized personnel. You can also implement appropriate security measures such as encryption and access controls to secure against unauthorized access or disclosure of sensitive information.

Related concepts

Log forwarding

You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

Related tasks

Extracting logs to CSV or JSON

Transaction Analysis Workbench can extract logs from their original z/OS-based formats to comma-separated values (CSV) or JavaScript Object Notation (JSON) format for ingestion by analytics platforms and other applications.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 37. Forwarding logs to Elastic

You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a TCP network to the Elastic Stack.

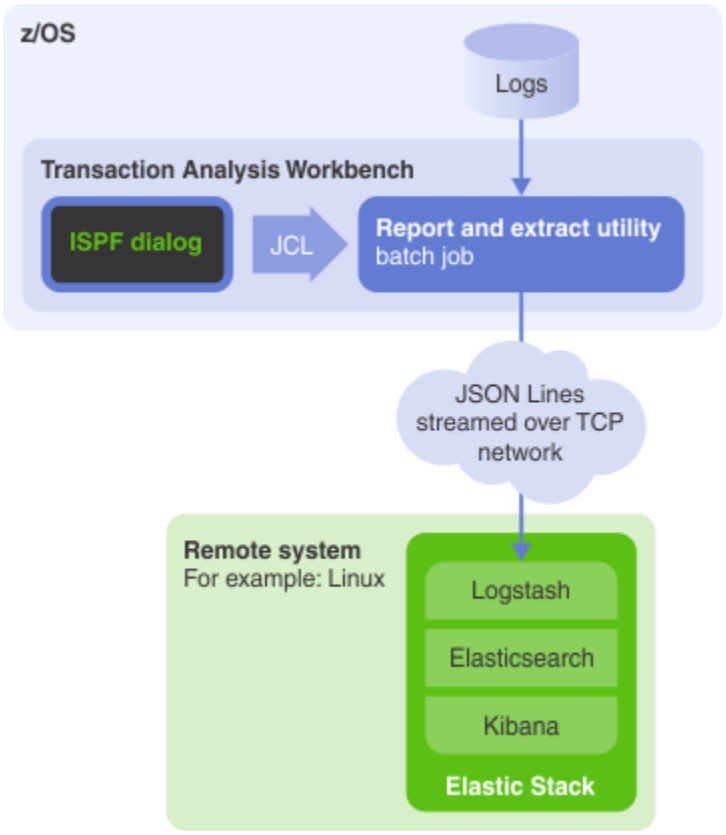


Figure 67. Forwarding logs to Elasticsearch via Logstash: streaming JSON Lines over TCP

The system running Elastic might not be "remote": you can run Elastic on Linux® on the same z System server as Transaction Analysis Workbench.

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Elasticsearch configuration

By default, from Elastic 5.0, Elasticsearch maps string fields to the `text` type. Elasticsearch parses the contents of `text` fields into tokens for full-text search. You might not want that default behavior. Many string fields in log data are names or identifiers. It makes more sense to search these fields as whole values, so the `keyword` type is a better choice. You can configure Elasticsearch by creating an index template that maps string fields to the `keyword` type.

Here is an example of why you might choose to map string fields to the `keyword` type. Suppose you forward to Elasticsearch a string field named `tran` with the value `STC@CICS`. By default, Elasticsearch analyzes (tokenizes) that string field. When you refer to the `tran` field, Elasticsearch returns the value of the first token, `stc`. To return the complete, original value, you need to refer to the name of a separate, keyword ("raw") version of that field.

If you want to avoid that default behavior, then, as a starting point, consider creating an index template that maps all string fields forwarded by Transaction Analysis Workbench to the `keyword` type. Then consider adjusting the mapping of specific string fields that you might want analyzed, such as URLs, to the `text` type.

For details on how Elasticsearch analyzes string fields, the differences in behavior when searching `text` fields versus `keyword` fields, how to create index templates, and the syntax for specific versions of Elasticsearch, see the Elasticsearch documentation.

Example: From Elastic 5.0

The following index template for the index pattern `fuw-*` maps all string fields to the `keyword` type.

```
{
  "template": "fuw-*", 1
  "mappings": {
    "_default_": {
      "dynamic_templates": [ {
        "string_fields": {
          "match": "*",
          "match_mapping_type": "string",
          "mapping": {
            "type": "keyword"
          }
        }
      } ]
    }
  }
}
```

1

For this index template to apply to data that you forward from Transaction Analysis Workbench, you must forward data to an Elasticsearch index that matches the `fuw-*` index pattern. The output section of your Logstash configuration file must specify an `index` property value that begins with "fuw-".

Example: Elasticsearch 2.*x*

For the same reasons that you might wish to map string fields to `keyword` in later versions of Elastic, you might wish to set string fields in Elasticsearch 2.*x* to `not_analyzed`.

The following index template for the index pattern `fuw-*` maps all string fields to `not_analyzed`. This example is for Elasticsearch 2.4.

```
{
  "template" : "fuw-*",
  "mappings" : {
    "_default_" : {
      "dynamic_templates" : [ {
        "string_fields" : {
          "mapping" : {
            "index" : "not_analyzed",
            "omit_norms" : true,
            "type" : "string"
          },
          "match_mapping_type" : "string",
          "match" : "*"
        }
      } ]
    }
  }
}
```

Logstash configuration for output to Elasticsearch

The Logstash configuration file ("config") for listening on a TCP port for JSON Lines from Transaction Analysis Workbench is concise and works for all log record types from Transaction Analysis Workbench.

To create the Logstash config, either copy the following example, or run the Transaction Analysis Workbench report and extract utility, and specify a **JSON** command with a **LOGSTASHCONFIG** parameter.

Example Logstash config

```
input {
  tcp {
    port => 6789 1
    codec => json_lines
  }
}
filter {
  date {
    match => ["time", "ISO8601"] 2
  }
}
output {
  elasticsearch {
    hosts => ["localhost"]
    document_type => "%{type}"
    index => "fuw-%{type}-%{+YYYY.MM.dd}"

    # The following manage_template setting assumes that
    # you have created an index template for the fuw-* index pattern.
    # For details, see the Workbench product documentation. 3
    manage_template => false
  }
}
```

Notes:

1

The config specifies the TCP port number on which Logstash listens for JSON Lines input.

2

The date filter sets the value of the Logstash `@timestamp` field to the value of the `time` field in the JSON Lines input. The `time` field is the event time stamp of the original log record.

The `ISO8601` value specified by the `match` option is compatible with the `TIMEFORMAT(ISO8601)` parameter of the Transaction Analysis Workbench **JSON** command. If you specify a `TIMEFORMAT` parameter value that is not compatible with this date filter, then you must edit the date filter to match.

3

See “Elasticsearch configuration” on page 189.

Example JCL to create the Logstash config

The following JCL creates the Logstash config shown in the previous example:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//LSCONFIG DD SYSOUT=*
//SYSIN    DD *
SCHEMAONLY 1
STREAM NAME(ELASTIC) HOST(myserver) PORT(6789) 2
JSON CODE(SMF:30.) + 3
  STREAM(ELASTIC) +
  EBCDIC CODEPAGE(037) + 4
  LSCONFIG(LSCONFIG) 5
/*
```

Notes:

1

The **SCHEMAONLY** command allows the **JSON** command to produce some outputs, such as a Logstash config, without any input logs.

2

The **SCHEMAONLY** command prevents the **JSON** command from producing any JSON data output, so this **STREAM** command will not stream any data. However, the **PORT** parameter value is used in the Logstash config.

3

CODE is a required parameter of the **JSON** command. However, in this instance, the value of the **CODE** parameter is not important, because the **JSON** command creates the same Logstash config for all record types.

4

If a **JSON** command specifies a **STREAM** parameter, then, by default, all output from the **JSON** command is in ASCII. However, for this example, we want to create the Logstash config in EBCDIC.

5

The job writes the Logstash config to the ddname LSCONFIG.

Logstash configs for CSV

JSON Lines is the recommended data format for forwarding logs from Transaction Analysis Workbench to Logstash. If you forward JSON Lines, you can use a single, concise Logstash config for all logs from Transaction Analysis Workbench.

However, if you prefer to forward CSV, you can. To create a Logstash config for CSV, use the **CSV** command with a **LOGSTASHCONFIG** parameter. In the previous example JCL, simply replace the **JSON** command keyword with **CSV**.

The **CSV** command creates Logstash configs that are not only specific to each record type, but are specific to the particular set of fields that you select to forward.

Chapter 38. Forwarding logs to Splunk

You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format to a TCP data input on a remote Splunk indexer.

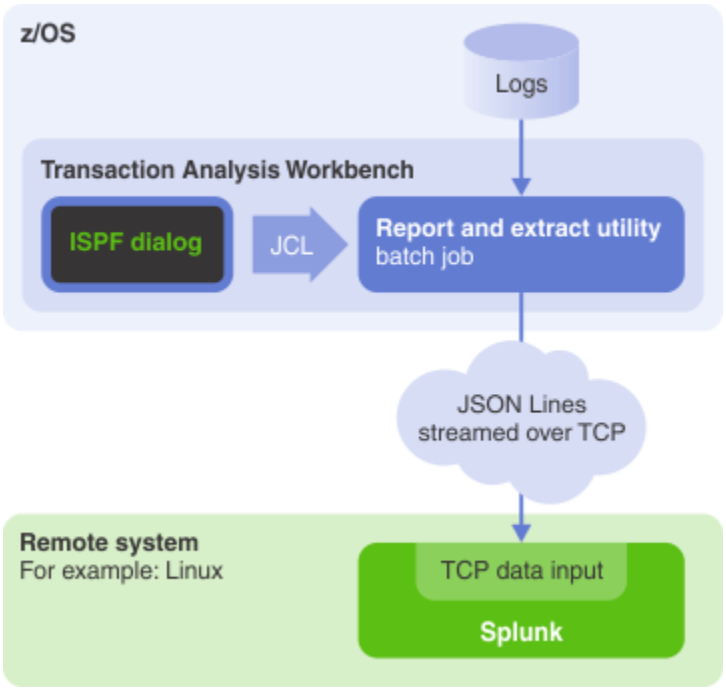


Figure 68. Forwarding logs to Splunk: streaming JSON Lines over TCP

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Basic Splunk configuration for streaming JSON Lines over TCP

To stream JSON Lines to Splunk over TCP, you need to configure a Splunk TCP data input that breaks each line of the stream into a separate event, recognizes event time stamps, and specifies the event data format as JSON.

The following Splunk configuration stanzas define a minimal basic configuration for streaming JSON Lines over TCP: one stanza in `inputs.conf`, and one in `props.conf`.

Depending on your own site practices, you might perform additional configuration, such as assigning different source types, routing events to different indexes, or using secure TCP.

Location of Splunk configuration stanzas

This Transaction Analysis Workbench documentation refers to Splunk configuration (`.conf`) file names, but not directory paths. It is your decision where to store the Splunk configuration stanzas for Transaction Analysis Workbench.

For example, you might choose to create a Splunk application directory named *`your-organization`*`-fuw` specifically for Transaction Analysis Workbench, and save the configuration files there:

`$SPLUNK_HOME/etc/apps/`*`your-organization`*`-fuw/local/*.conf`

inputs.conf

The following stanza in `inputs.conf` defines an unsecure TCP input that listens on port 6068, assigns the source type "fuw" to all incoming events, and stores the events in the default index (typically, main):

```
[tcp://:6068]
sourcetype = fuw
```

The port number and source type shown here are examples only. The actual values are your choice.

props.conf

The following stanza in `props.conf` defines the properties of the "fuw" source type:

```
[fuw]
SHOULD_LINEMERGE = false
KV_MODE = json
TIME_PREFIX = {\"time\":\"
TIME_FORMAT = %Y-%m-%dT%H:%M:%S.%6N%:z
```

The combination of `SHOULD_LINEMERGE = false` and `KV_MODE = json` defines the incoming data as JSON Lines: one event per line, data in JSON format. These two settings apply to different stages in the Splunk data pipeline: **SHOULD_LINEMERGE** applies to parsing, before indexing; **KV_MODE** applies later, to search-time field extraction.

The example regular expression for `TIME_PREFIX` is case sensitive; it matches the lowercase field name `time`, which is the default field name for event time stamps in JSON from Transaction Analysis Workbench.

The example value for `TIME_FORMAT` matches time stamps from Transaction Analysis Workbench that have been created by a **JSON** command that specifies the parameter `TIMEFORMAT(ISO8601)`.

Related concepts

Time stamps in CSV and JSON
CSV and JSON output by Transaction Analysis Workbench includes the time stamp of the corresponding input log record, also known as the *event* time stamp. The event time stamp is always included in the output, regardless of any **CSV** or **JSON** command parameters, such as **FIELDS**, that specify which other fields to include. Depending on the input record type and the selected fields, other fields in the output might also be time stamps. All time stamps in the output, including the event time stamp, have the same format.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

STREAM command
Forwards log data in JSON Lines or CSV format to a TCP socket.

Splunk source types

Rather than assigning the same source type to all events from Transaction Analysis Workbench, you might prefer more granularity; more source types. Here are some methods for assigning different source types to events.

Per port

You can define a TCP input (port) for each source type.

For example, in `inputs.conf`:

```
[tcp://:6068]
# Miscellaneous z/OS events
sourcetype = fuw

[tcp://:6069]
# CICS events
sourcetype = cics

[tcp://:6070]
# DB2 events
sourcetype = db2
```

In `props.conf`, use the regex-type "or" operator (|) to set the same properties for all of these source types:

```
[fuw|cics|db2]
# Common properties for these source types
```

With this method, you must forward logs to the corresponding port for each source type. For instance, in this example, you must ensure that the Transaction Analysis Workbench **JSON** commands for the logs that are to be source type "cics" refer to a **STREAM** command for port 6069.

Per stream

If you add the following setting to your Splunk `inputs.conf` stanza:

```
requireHeader = true
```

then you can use the **HEADER** parameter of the Transaction Analysis Workbench **STREAM** command to send a header line that overrides the source type for events sent in the subsequent JSON Lines.

For example:

```
STREAM NAME(SPLUNK) +
       HEADER(***SPLUNK*** sourcetype=cics)
```

Per event

You can use transforms in Splunk to override the source type per event.

Each line of JSON Lines from Transaction Analysis Workbench contains a `type` field that identifies the log type and code of the original log record. You can use this field to set the Splunk source type.

For example, in `props.conf`, append the following line to the stanza for the corresponding source type or input:

```
TRANSFORMS-changesourcetype = set_sourcetype_fuw
```

and add the following stanza to `transforms.conf`:

```
[set_sourcetype_fuw]
# Set sourcetype to type field in JSON from FUW
REGEX = \"type\":\"([^\"]+)\"
FORMAT = sourcetype::$1
DEST_KEY = MetaData:Sourcetype
```

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

STREAM command

Forwards log data in JSON Lines or CSV format to a TCP socket.

Splunk indexes

Rather than storing all events from Transaction Analysis Workbench in the same Splunk index, you might prefer to use multiple indexes. Here are some methods for controlling which indexes events are stored in.

The following examples assume that you have already created the indexes that you want to use.

Per port

You can define a TCP input (port) for each index.

For example, in `inputs.conf`:

```
[tcp://:6068]
# Miscellaneous z/OS events
index = fuw

[tcp://:6069]
# CICS events
index = cics

[tcp://:6070]
# DB2 events
index = db2
```

With this method, it is up to you forward the logs to the corresponding port for that index. When writing the JCL to forward logs, you must ensure that the **JSON** commands for the logs that you want stored in "cics" refer to the **STREAM** command for the correct port.

Per stream

If you add the following setting to your Splunk `inputs.conf` stanza:

```
requireHeader = true
```

then you can use the **HEADER** parameter of the Transaction Analysis Workbench **STREAM** command to send a header line that overrides the index for events sent in the subsequent JSON Lines. For example:

```
STREAM NAME(SPLUNK) +
       HEADER(***SPLUNK*** index=cics)
```

Per event

You can use transforms in Splunk to override the index per event.

Each line of JSON Lines from Transaction Analysis Workbench contains a `type` field that matches the log type of the original log record. You can use this field to specify the Splunk index. For example, in `props.conf`, append the following line to the stanza for the corresponding source type or input:

```
TRANSFORMS-changeindex = set_index_fuw
```

and add the following stanza to `transforms.conf`:

```
[set_index_fuw]
# Route events to type-specific index, or fall back to default index
REGEX = \"type\":\"(dtr|cmf|ims)-
FORMAT = $1
DEST_KEY = _MetaData:Index
DEFAULT_VALUE = misc
```

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

STREAM command
Forwards log data in JSON Lines or CSV format to a TCP socket.

Splunk secure TCP

You can configure Transaction Analysis Workbench and Splunk to use Transport Layer Security (TLS) over TCP.

`inputs.conf`

To configure an *unsecure* Splunk TCP input, you add a `[tcp]` stanza to the `inputs.conf` file.

To configure a *secure* TCP input, you add a `[tcp-ssl]` stanza, and specify security details in the `[SSL]` stanza. The settings in the `[SSL]` stanza apply to all secure TCP inputs.

For example:

```
[SSL]
rootCA = $SPLUNK_HOME/etc/auth/cacert.pem
serverCert = $SPLUNK_HOME/etc/auth/server.pem

[tcp-ssl:6071]
sourcetype = fuw
```

Cipher suites: numbers versus names

If you want to explicitly specify which cipher suites to use, you need to be aware that Splunk and Transaction Analysis Workbench use different identifiers to refer to the same cipher suites:

- Splunk uses the cipher suite names from OpenSSL.

  The **`cipherSuite`** setting in the `[SSL]` stanza of the `inputs.conf` file accepts a colon-delimited list of cipher suite names.
- Transaction Analysis Workbench uses the 4-digit cipher suite numbers from IBM Global Security Kit (GSKit).

  The **CIPHERS** parameter of the **STREAM** command accepts an undelimited list of 4-digit cipher suite numbers.

For example, the Transaction Analysis Workbench parameter `CIPHERS(000A0039)` and the Splunk setting:

```
cipherSuite = DES-CBC3-SHA:DHE-RSA-AES256-SHA
```

both refer to the cipher suites TLS_RSA_WITH_3DES_EDE_CBC_SHA and TLS_DHE_RSA_WITH_AES_256_CBC_SHA that are listed in the Internet Assigned Numbers Authority (IANA) TLS cipher suite registry and defined in Internet Engineering Task Force (IETF) Request for Comments (RFCs).

To map an OpenSSL cipher suite name to a GSKit cipher suite number, use the RFC name. For example, suppose you know the GSKit 4-digit cipher suite number that you want to use, and you want to configure Splunk to use that cipher suite:

1. Go to the cipher suite definitions in the IBM z/OS Cryptographic Services System SSL programming documentation.
2. Look up the RFC-standard "short name" for the cipher suite number.

Tip: The IANA TLS cipher suite registry contains a similar table that maps cipher suite numbers to RFC names (labeled in the registry as "descriptions").

3. Go to the OpenSSL **ciphers** command documentation.
4. Look up the corresponding OpenSSL name for that RFC name.

Chapter 39. Forwarding logs to Hadoop

You can convert log data to CSV format, and then use file transfer methods to forward the data to a remote Hadoop system.

To transfer files to the remote system, you can use one of the following methods, both based on the Secure Shell (SSH) network protocol:

SFTP
To use this method, you must have OpenSSH installed on z/OS and SSH running on the remote system. OpenSSH is available as part of IBM Ported Tools for z/OS.

Co:Z
To use this method, you must have Dovetailed Technologies Co:Z Co-Processing Toolkit for z/OS installed, and the associated Co:Z Target System Toolkit (sometimes referred to as the "agent") installed on the remote system. Co:Z requires OpenSSH.

While SFTP simply transfers files to the remote system, the Co:Z step runs commands on the remote system issues a Hadoop command on the remote system to put the CSV file into HDFS and, optionally, a Hive command to create a catalog table.

The following figure presents an overview of the process, including the HCatalog table schema and log forwarding using Co:Z or SFTP:

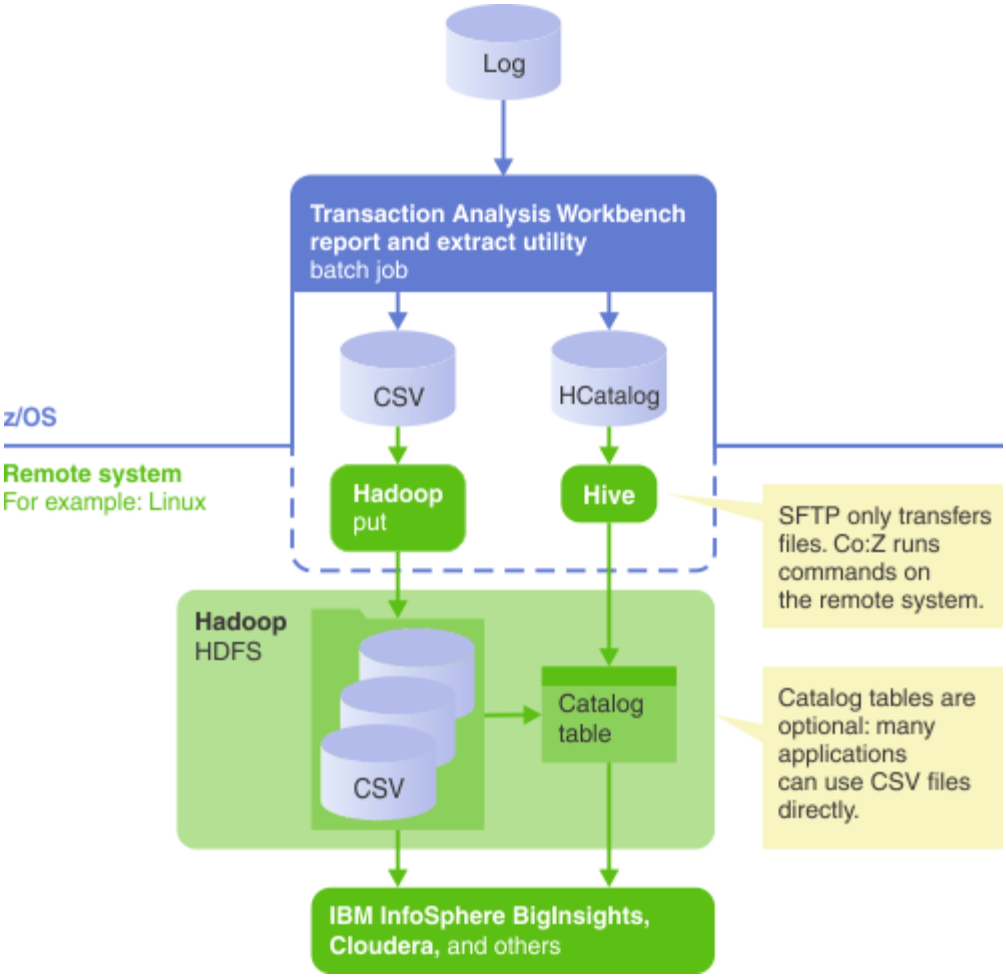


Figure 69. Forwarding logs to Hadoop: transferring CSV and HCatalog files

For simplicity, the previous figure shows a batch job that creates only a single CSV file and corresponding HCatalog file. In practice, Transaction Analysis Workbench creates a CSV file and an HCatalog file for each record type that you select for forwarding.

Creating JCL that forwards log data to Hadoop

To create JCL that forwards log data to Hadoop, tailor the example JCL provided here.

You must have a Hadoop distribution installed on some platform. You need to know how to create directories and put files in HDFS on that platform, and how to use Hadoop-based tools to open CSV files or catalog tables. If you want to use catalog tables instead of raw CSV files, it is helpful if you understand the syntax of the **CREATE TABLE** Hive data definition language (DDL) command in HCatalog table schemas; in particular, how the **LOCATION** parameter defines an external table that refers to data files in an HDFS directory.

To perform the optional log forwarding step, you must have SFTP or Co:Z installed on the local z/OS system that is running Transaction Analysis Workbench and on the remote system that is running Hadoop.

You need to know the location on z/OS of the logs that you want to forward.

The example JCL runs the **CSV** command of the Transaction Analysis Workbench report and extract utility. The **CSV** command extracts log records to a comma-separated values (CSV) file and creates a corresponding HCatalog table schema. An optional step then uses SFTP or Co:Z to forward the extracted data to the remote system where Hadoop is running.

The following JCL forwards CICS monitoring facility performance class records and DB2 accounting records to Hadoop:

```
//UIDFUW   JOB NOTIFY=&SYSUID  1
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK  2
//SYSPRINT DD  SYSOUT=*
//SMFIN001 DD  DISP=SHR,  3
//             DSN=SYS1.DAILY.SMF
//CICSC    DD  PATH='/u/hadoop/cics.csv',  4
//             PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//             PATHDISP=(KEEP,DELETE),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//CICSM    DD  PATH='/u/hadoop/cics.hcatalog',
//             PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//             PATHDISP=(KEEP,DELETE),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//DB2C     DD  PATH='/u/hadoop/db2.csv',
//             PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//             PATHDISP=(KEEP,DELETE),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//DB2M     DD  PATH='/u/hadoop/db2.hcatalog',
//             PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//             PATHDISP=(KEEP,DELETE),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSIN    DD  * 5
* CICS CMF performance class
CSV CODE(CMF) +
    OUTPUT(CICSC) +
    HCAT(CICSM) +
    LOCATION(/data/cics) +
    TABLE(cics) +
    THM CODEPAGE(1047) NL
*   LABELS  /* Uncomment to include header rows
FIELDS(
       TRAN
       USRCPUT
       /* More fields...
      )
  .
  .
  )

* DB2 accounting
CSV CODE(DTR:003) +
    OUTPUT(DB2C) +
    HCAT(DB2M) +
    LOCATION(/data/db2) +
    TABLE(db2) +
```

```
    THM CODEPAGE(1047) NL
*   LABELS  /* Uncomment to include header rows
FIELDS(
  SM101SID:LPAR
  SM101SSI:SSID
  QWHCCTR:Tran
  QWHCPLAN:Plan
  QWHCOPID:Userid
  QWHCCN:APPLID
  ET1
  :
  )

/*
// IF CSV.RC LE 4 THEN
//FORWARD  EXEC PGM=COZLNCH,REGION=0M, 6
// PARM='MSGFILE(COZLOG)/user@hadoop'
//STEPLIB  DD DISP=SHR,DSN=COZ.LOADLIB
//COZCONF  DD DISP=SHR,DSN=COZ.SAMPJCL(COZCFGD)
//         DD *
/*
//COZLOG   DD SYSOUT=*
//COZOUT   DD SYSOUT=*,
//            DCB=(RECFM=VB,LRECL=255,BLKSIZE=10000)
//STDOUT   DD SYSOUT=*,
//            DCB=(RECFM=VB,LRECL=255,BLKSIZE=10000)
//STDERR   DD SYSOUT=*,
//            DCB=(RECFM=VB,LRECL=255,BLKSIZE=10000)
//STDIN    DD *
hadoop fs -put <(fromfile /u/hadoop/cics.csv) /data/cics/$(date +"%s")\
.csv
# To create a catalog table, run Hive:
#hive -f <(fromfile /u/hadoop/cics.hcatalog)
hadoop fs -put <(fromfile /u/hadoop/db2.csv) /data/db2/$(date +"%s").c\
sv
# To create a catalog table, run Hive:
#hive -f <(fromfile /u/hadoop/db2.hcatalog)
/*
// ENDIF
//*
// IF FORWARD.RC = 0 THEN
//DELETE   EXEC PGM=IEFBR14 7
//CICSC    DD  PATH='/u/hadoop/cics.csv',
//             PATHDISP=(DELETE,DELETE)
//CICSM    DD  PATH='/u/hadoop/cics.hcatalog',
//             PATHDISP=(DELETE,DELETE)
//DB2C     DD  PATH='/u/hadoop/db2.csv',
//             PATHDISP=(DELETE,DELETE)
//DB2M     DD  PATH='/u/hadoop/db2.hcatalog',
//             PATHDISP=(DELETE,DELETE)
// ENDIF
```

Notes on the JCL:

1

The job statement, as specified by the **Job Statement Information** field under dialog option 0.1 **Workbench Personal Settings**.

2

The library containing the Transaction Analysis Workbench executable load modules, as specified by the **Workbench Load Library** field under dialog option 0.1 **Workbench Personal Settings**.

3

The DD statement for the input SMF file.

4

The DD statements for each CSV and HCatalog output file. In this example, with two record types selected, there are four output files: two CSV files and two HCatalog files. In this example JCL, the ddname suffix "C" identifies CSV files, "M" (metadata) identifies HCatalog files.

You can write output to either z/OS UNIX files or MVS sequential data sets. If you want to use SFTP log forwarding, write output to z/OS UNIX files.

5

The SYSIN data set for the Transaction Analysis Workbench report and extract utility program, FUWBATCH. For each record type selected for extraction, the SYSIN data set contains a **CSV** command that includes the following parameters:

CODE
The log type and code of the record type. **CODE(CMF)** is an abbreviated alias for the log type and code combination **CODE(CMF:6E13)**.

OUTPUT
The ddname of the output CSV file.

HCAT
The ddname of the output HCatalog file.

LOCATION
The value of the **LOCATION** parameter in the HCatalog file; the HDFS directory where Co:Z puts the CSV file on the remote system.

TABLE
The table name for the **CREATE TABLE** command in the HCatalog file.

FIELDS
The list of fields to be extracted to the CSV file. For conciseness, the example JCL listing shows only the first few fields.

You can rename fields to make them more meaningful. For example, `FIELDS(QWHCCTR:Tran)` extracts the DB2 field QWHCCTR; the HCatalog file and CSV header row, if requested, will refer to the field as "Tran".

These **CSV** commands perform the following actions:

- Extract the selected record types from the dumped SMF data set 'SYS1.DAILY.SMF' to the following CSV files on z/OS UNIX:

  /u/hadoop/cics.csv
  Contains data extracted from CICS monitoring facility performance class records

  /u/hadoop/db2.csv
  Contains data extracted from DB2 accounting records

- Create the following HCatalog files, corresponding to the CSV files:

  /u/hadoop/cics.hcatalog
  Contains a **CREATE TABLE** command that creates an external table named "cics" that refers to data in CSV files in the HDFS directory /data/cics

  /u/hadoop/db2.hcatalog
  Contains a **CREATE TABLE** command that creates an external table named "db2" that refers to data in CSV files in the HDFS directory /data/db2

6

The Co:Z log forwarding step. This step uses Co:Z to perform the following commands on the remote system named "hadoop":

1. Put the CSV files in the following HDFS paths on the remote system:

   /data/cics/1429081644.csv
   /data/db2/1429081646.csv

   where the CSV file names represent a number of seconds since the UNIX epoch.

   The **LOCATION** parameter in each HCatalog file refers to the HDFS directory containing the corresponding CSV files.

2. If you uncomment the `hive` commands: use the HCatalog files to create two catalog tables, "cics" and "db2".

You only need to run these **hive** commands once, to create the catalog tables that refers to the HDFS directory containing CSV files. You do not need to run that **hive** command each time you put a new CSV file into the HDFS directory.

To use SFTP instead of Co:Z:

1. Replace the FORWARD step in the example JCL with the following step:

```
//FORWARD  EXEC PGM=IKJEFT01,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSTSIN  DD *
BPXBATCH SH sftp -b /u/hadoop/batch.sftp user@hadoop
/*
```

2. Create a corresponding SFTP script file (/u/hadoop/batch.sftp). For example:

```
ascii
-mkdir -p /u/input/logs/cics/
cd input/logs/cics/
put /u/hadoop/cics.hcatalog
put /u/hadoop/cics.csv
-mkdir -p /u/input/logs/cics/
put /u/hadoop/db2.hcatalog
put /u/hadoop/db2.csv
```

This example uses fixed CSV file names; with this method, you need to manage CSV file names.

Note that this SFTP-based method only puts files on the remote system; this method does not load files into Hadoop.

7

The step to delete the CSV and HCatalog output files from z/OS, because they are no longer required: the CSV files have been forwarded to the remote system, and the HCatalog files have been used to create the catalog tables. Deleting the output files makes the JCL reusable; for example, for use in an automated job schedule to process new logs.

You can tailor the JCL to meet your specific requirements. For example, you can filter the records to be extracted. To only extract CICS monitoring facility performance class records for transaction codes starting with "MOB" that had response times greater than two seconds, insert the following **CODE** command and **COND** statements after the corresponding **CSV** command:

```
CSV CODE(CMF)
⋮
CODE(CMF)
  COND TRAN EQ 'MOB*'
  COND RESPONSE GT 2.0
```

The log type and code specified by the **CODE** command must match the **CODE** parameter of the preceding **CSV** command: in this example, CMF.

HCatalog table schema for Hadoop

When extracting logs to comma-separated values (CSV) format, you can create an HCatalog table schema for the CSV data, for use with Hadoop.

To create an HCatalog table schema, specify the **HCATALOG** parameter of the **CSV** command of the Transaction Analysis Workbench report and extract utility.

The HCatalog table schema consists of a Hive DDL **CREATE TABLE** statement that creates a catalog table; specifically, an external table that uses data in CSV files stored in HDFS.

Example

The following **CSV** command and subsequent **FIELDS** command:

```
CSV CODE(CMF) + 1
    HCATALOG(HCAT) +
    TABLE(cics) + 2
    LOCATION(/data/cics) + 3
    FIELDCASE(UPPER) +
    ⋮
FIELDS( 4
  SMFSID:LPAR
  APPLID
  TRAN
  ⋮
  )
```

creates the following HCatalog file in the destination specified by the ddname HCAT:

```
CREATE EXTERNAL TABLE cics( 2
`TIME` TIMESTAMP COMMENT "Timestamp", 4
`LPAR` STRING COMMENT "System identification",
`APPLID` STRING COMMENT "Generic APPLID",
`TRAN` STRING COMMENT "DFHTASK C001 Transaction …
)
COMMENT "CICS Transaction"
ROW FORMAT
DELIMITED FIELDS TERMINATED BY ','  ESCAPED BY '\\' STORED AS TEXTFILE
⋮
LOCATION '/data/cics' 3
TBLPROPERTIES("type" = "CMF", "code" = "6E13"); 1
```

Notes:

1

In this example, the **CODE** parameter of the **CSV** command selects CICS monitoring performance class records. `CODE(CMF)` is an abbreviated alias for the log type and code combination `CODE(CMF:6E13)`.

In the HCatalog file, the **TBLPROPERTIES** parameter defines the log type and code as table properties.

2

The **TABLE** parameter of the **CSV** command specifies the name of the catalog table to be created: in this example, "cics".

3

The **LOCATION** parameter of the **CSV** command specifies the path of the HDFS directory that contains, or will contain, one or more CSV files that match the table schema.

4

The **FIELDS** parameter of the **CSV** command specifies the fields to be extracted. The order of field names in the **FIELDS** parameter is not significant. The order of field names in the HCatalog file matches the order of the fields in the corresponding CSV files.

Chapter 40. Loading logs into DB2

You can convert log data to CSV format, and then load the data into DB2.

The **CSV** command of the Transaction Analysis Workbench report and extract utility has parameters specifically for loading logs into DB2:

SCHEMA
Writes a DB2 DDL **CREATE TABLE** statement that matches the columns in the CSV output.

DB2LOAD
Writes a corresponding **LOAD** control statement for the DB2 load utility.

Creating JCL that loads logs into DB2

To create JCL that loads logs to DB2, use option 5 **Analytics** of the Transaction Analysis Workbench ISPF dialog. The dialog creates JCL that runs the **CSV** command of the Transaction Analysis Workbench report and extract utility. The **CSV** command extracts log records to a comma-separated values (CSV) file and creates a corresponding DB2 DDL **CREATE TABLE** statement and DB2 load utility **LOAD** control statement. The generated JCL includes a step that runs the DB2 load utility.

The DB2 database and table space where you want to store the log data must already exist.

You must have the authority to create tables in that table space, and then load data into the tables.

The following procedure creates a DB2 DDL **CREATE TABLE** statement for each record type that you select for forwarding. You will need to use your own site-specific procedures to run that statement; the generated JCL does not perform that step. If you need help with that step, contact your DB2 administrator.

You need to know the following details:

- The data set name of the SMF file or IMS log that contains the data you want to forward.
- The DB2 subsystem ID (SSID).
- The data set name of the DB2 load library that contains the DB2 load utility program. The default low-level qualifier of this data set name is SDSNLOAD.

The following procedure uses the Transaction Analysis Workbench ISPF dialog to create JCL for two jobs:

1. The first job creates a DB2 DDL **CREATE TABLE** statement and a DB2 load utility **LOAD** control statement for each record type that you have selected to load into DB2.

   You edit the database and table name in the resulting statements to match your site requirements, and then you use the **CREATE TABLE** statements to create the DB2 tables.
2. The second job converts log records to CSV format, and then runs the DB2 load utility, with the **LOAD** control statement created by the first job, to load the CSV data into the DB2 tables that you created using the DDL from the first job.

You only need to run the first job once, before you load a record type into DB2 for the first time.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 5 **Analytics**.
2. Select a data source: SMF or IMS.
   a) Enter the data set name of the input SMF data set or IMS log data set.
   b) For SMF, select one or more record types.

   The generated JCL will contain a **CSV** command for each selected record type.
3. Select a destination: on the command line, enter the menu option number for DB2.

   The panel for forwarding to DB2 is displayed.

4. If you have previously loaded into DB2 all of the record types that you have selected for forwarding, then the corresponding DB2 tables already exist: skip to step “11” on page 206. Otherwise, continue to the next step.
5. Select option 1 **Create DDL statements**.
6. Enter output data set names for the DB2 **DDL CREATE TABLE** statements.

   To allocate the data sets, the generated JCL uses the settings specified in dialog option 0.3 **Extract Data Set Allocations**.

   If you selected more than one record type, use the substitution variable %RTYP to create unique output data set names. In the generated JCL, this variable name is replaced by a mnemonic that represents the record type. For example, for CICS monitoring facility performance class records, the data set name `'MY.%RTYP.CSV'` becomes `'MY.CICS.CSV'`.
7. Enter SUB on the command line to create the JCL.

   The JCL is displayed in an edit panel.
8. Submit the JCL.

   The generated JCL writes **CREATE TABLE** statements to ddnames with the suffix "S", **LOAD** statements to ddnames with the suffix "L".
9. Edit the database and table names in the **CREATE TABLE** and **LOAD** statements to match your site requirements.
10. Use the **CREATE TABLE** statements to create DB2 tables.

    How you create DB2 tables depends on local practices at your site. If you need help, contact your DB2 administrator.
11. Select option 2 **Create CSV and LOAD statements, and then load into DB2**.
12. Enter the output data set names for the CSV data and the DB2 load utility **LOAD** control statements.

    As for the DDL statements created earlier, if you selected more than one record type, use the substitution variable %RTYP to create unique output data set names.
13. Enter the destination DB2 subsystem ID.
14. Enter the data set name of the DB2 load module library that contains the DB2 load utility.

    The default low-level qualifier of this library is SDSNLOAD. For example, `'DB2.PROD.SDSNLOAD'`.
15. Enter SUB on the command line to create the JCL.

    The JCL is displayed in an edit panel.

Example JCL

Example JCL for option 1, creating DDL statements for two SMF record types, CICS CMF performance class and DB2 accounting:

```
//UIDFUW   JOB NOTIFY=&SYSUID 1
//CSV      EXEC PGM=FUWBATCH
//STEPLIB  DD  DISP=SHR,
//             DSN=FUW.SFUWLINK 2
//SYSPRINT DD  SYSOUT=*
//CICSS    DD  DSN=MYID.CICS.DDL,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//DB2S     DD  DSN=MYID.DB2.DDL,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSIN    DD  *
SCHEMAONLY 3
* CICS CMF performance class
CSV CODE(CMF) +
    OUTPUT(CICSC) +
    SCHEMA(CICSS) +
    TOKENS(CICS) DELIMITER(,) +
    FIELDCASE(UPPER) +
    TABLE(MYSCHEMA.CICS) + 4
```

```
    NOLABELS +
    CODEPAGE(1047)
FIELDS(
  ⋮
  )

* DB2 accounting
CSV CODE(DTR:003) +
    OUTPUT(DB2C) +
    SCHEMA(DB2S) +
    TOKENS(DB2) DELIMITER(,) +
    FIELDCASE(UPPER) +
    TABLE(MYSCHEMA.DB2) +
    NOLABELS +
    CODEPAGE(1047)
FIELDS(
  ⋮
  )
/*
```

Notes on the JCL:

1

The job statement is specified by the **Job Statement Information** field under dialog option 0.1 **Workbench Personal Settings**.

2

The Transaction Analysis Workbench executable load module library is specified by the **Workbench Load Library** field under dialog option 0.1 **Workbench Personal Settings**.

3

The **SCHEMAONLY** global command causes other commands to ignore any input logs. In this example, the only output from the **CSV** commands is DDL statements, as specified by the **SCHEMA** parameters. The **OUTPUT** parameters, which specify where to write converted input logs, produce no output.

4

The **TABLE** parameter specifies the table name used in the generated DDL statement.

Example JCL for option 2, creating CSV data and **LOAD** statements, and then loading into DB2:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//CSV      EXEC PGM=FUWBATCH
//STEPLIB  DD  DISP=SHR,
//             DSN=FUW.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN001 DD  DISP=SHR,
//             DSN=SYS1.DAILY.SMF  5
//CICSC    DD  DSN=MY.CICS.CSV,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//CICSL    DD  DSN=MY.CICS.LOAD,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//DB2C     DD  DSN=MY.DB2.CSV,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//DB2L     DD  DSN=MY.DB2.LOAD,
//             DISP=(NEW,CATLG),
//             RECFM=VB,LRECL=32756,BLKSIZE=32760,
//             UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSIN    DD  *
* CICS CMF performance class
CSV CODE(CMF) +
    OUTPUT(CICSC) +  6
    DB2LOAD(CICSL) +  7
    TOKENS(CICS) DELIMITER(,) +
    FIELDCASE(UPPER) +
    TABLE(FUW.CICS) +
    NOLABELS +
    CODEPAGE(1047)
FIELDS(
  ⋮
  )
```

```
* DB2 accounting
CSV CODE(DTR:003) +
    OUTPUT(DB2C) +
    DB2LOAD(DB2L) +
    TOKENS(DB2) DELIMITER(,) +
    FIELDCASE(UPPER) +
    TABLE(FUW.DB2) +
    NOLABELS +
    CODEPAGE(1047)
FIELDS(
  :
  )
/*
//*
// IF RC LE 4 THEN
//CICS     EXEC PGM=DSNUTILB,PARM='DB2A' 8
//STEPLIB  DD  DISP=SHR,DSN=DB2.SDSNLOAD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=MY.CICS.LOAD
//CICSC    DD  DISP=SHR,DSN=MY.CICS.CSV
// ENDIF
//*
// IF RC LE 4 THEN
//DB2      EXEC PGM=DSNUTILB,PARM='DB2A' 8
//STEPLIB  DD  DISP=SHR,DSN=DB2.SDSNLOAD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=MY.DB2.LOAD
//DB2C     DD  DISP=SHR,DSN=MY.DB2.CSV
// ENDIF
```

Notes on the JCL:

5
The SMFIN*nnn* DD statement specifies the input data set that contains the SMF records to load into DB2.

6
The **OUTPUT** parameter specifies where to write the log records after converting them to CSV format.

7
The **DB2LOAD** parameter writes a **LOAD** control statement for the DB2 load utility statement.

8
The JCL contains a DB2 load step for each record type. The **PARM** parameter of the **EXEC** statement specifies the destination DB2 subsystem ID.

For brevity, these examples elide the list of field names from the **FIELDS** commands.

Follow your site-specific procedures to schedule the JCL to run periodically, to load log data into DB2.

DB2 table schema

When extracting log data to CSV or JSON, you can optionally request a DB2 DDL **CREATE TABLE** statement (sometimes referred to as a table schema) that describes the field names and data types in the CSV or JSON output. You can use the schema to create a DB2 table into which you can import the data.

To request a DB2 table schema, specify the **SCHEMA** parameter of the **CSV** or **JSON** command.

The DB2 schema consists of a DB2 **CREATE TABLE** command that defines a column for each field in the corresponding CSV or JSON output.

Depending on the input log records, the DB2 schema can contain the following field types:

TIMESTAMP
Date and time field.

CHAR
Character field with a known length. This type includes strings of hexadecimal digits.

VARCHAR
Character field with an unknown or variable length.

INTEGER
Count field.

SMALLINT
Boolean flag field. The value 0 or 1 indicates a single field value. Higher values indicate the accumulated number of instances where the flag is on.

DECIMAL(15,6)
Duration in seconds, with fractions of seconds after the decimal point and a maximum precision of one microsecond.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 41. Other platforms, and testing without forwarding

To convert logs to CSV or JSON for use by platforms not listed under Transaction Analysis Workbench ISPF dialog option 5 **Analytics**, or to test output for listed platforms without forwarding, go to option 5 **Analytics**, and then select option 3 **Data Set**.

Option 3 **Data Set** creates JCL that can generate all of the outputs of other options in the menu, but without forwarding. Instead, this option writes output to MVS data sets, z/OS UNIX files, or sysout.

Uses for this option include:

- Exploring the different types of schema (metadata) that Transaction Analysis Workbench can create.
- Creating CSV or JSON for ingestion by an analytics platform, but without forwarding to the platform.

  You might find it useful to check that output from Transaction Analysis Workbench matches the requirements of your analytics platform *before* forwarding.
- Creating CSV or JSON for ingestion by analytics platforms not listed under option 5 **Analytics**.

  For example, creating CSV files for use by spreadsheet applications such as Microsoft Excel.

Tip: To write output to sysout, specify an asterisk (*) as the output destination rather than an MVS data set name or z/OS UNIX file path.

Part 8. Mobile Workload Pricing

Mobile Workload Pricing (MWP) for z/OS is a pricing model that reduces the cost of running mobile workloads on z/OS. To take advantage of MWP, you use the Mobile Workload Reporting Tool (MWRT) to submit monthly reports to IBM. MWRT requires two types of input files: CSV and SMF. You can use Transaction Analysis Workbench to create these files.

To create the files for MWRT, you write JCL that runs the **MWP** and **EXTRACT** commands of the Transaction Analysis Workbench report and extract utility. The **MWP** command creates a CSV file in the format required by MWRT; the **EXTRACT** command creates an SMF file. You must have collected the necessary log records on z/OS. To create the SMF file, you need the following records:

- SMF type 70 subtype 1
- SMF type 89 subtypes 1 and 2

Each CSV file contains mobile workload CPU times for one of the following subsystems (or "product families") on z/OS: CICS, DB2, IMS, WebSphere Application Server, or IBM MQ. The log records required to create these CSV files depend on the subsystem and whether only some or all of the transactions processed by an address space are mobile.

To *tag* (identify) transactions or address spaces as being eligible for MWP, you apply filters to the **MWP** command.

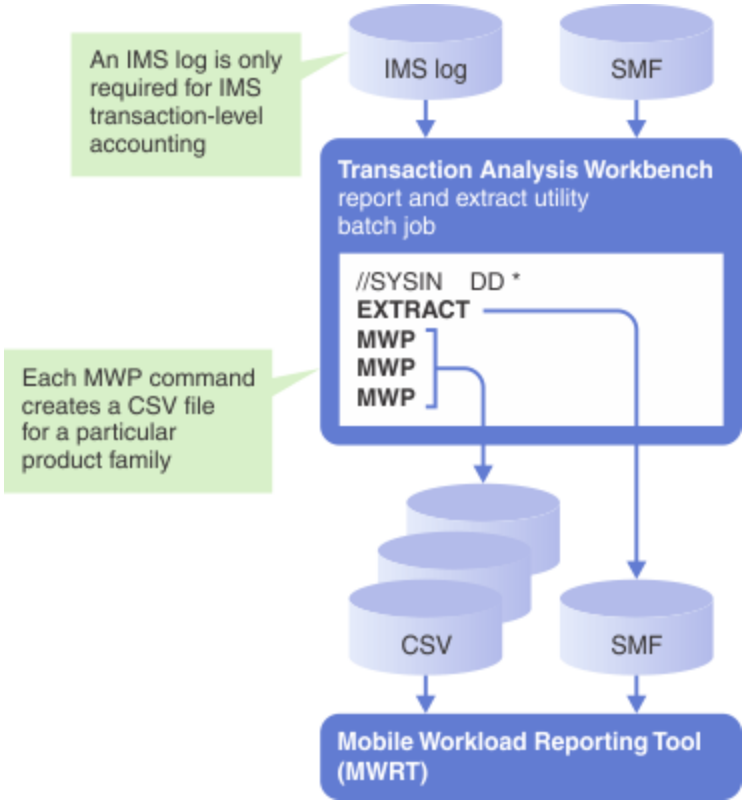


Figure 70. Overview of using Transaction Analysis Workbench to create CSV and SMF files for MWRT

Related reference

EXTRACT command
Writes the selected log records to an extract data set.

MWP command
Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file

reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

Chapter 42. Input logs and CPU time formulas for Mobile Workload Pricing

To attribute CPU time to mobile workloads, you can use one of two accounting methods, depending on whether only some or all of the transactions processed by an address space are mobile. The accounting method determines the type of input log records used by the **MWP** command of the Transaction Analysis Workbench report and extract utility. The input log records determine the formula that the **MWP** command uses to calculate CPU time.

The two accounting methods are known as *transaction-level* accounting and *address space-level* accounting.

Transaction-level accounting

Use this method for address spaces that process a mix of workloads, where only some of the transactions are mobile.

If you use this method, the **MWP** command requires log records from the individual subsystem. For IMS, you can choose between two types of IMS log record. The CPU times in the CSV file are calculated from fields in these log records according to the formula shown in the following table. For DB2, the formula used depends on whether you specify the NESTED parameter.

Table 10. MWP transaction-level accounting: log records required and formula for calculating CPU time

Parameter of the MWP command that identifies the product family	Product family	Required log records	Corresponding log type and code in Transaction Analysis Workbench	Field or formula for CPU time
CICS	CICS Transaction Server for z/OS	SMF 110 CMF performance class	CMF:6E13	CPUTONCP (group DFHTASK, field ID S436)
DB2	DB2 for z/OS	SMF 101 DB2 accounting (IFCID 003)	DTR:003	Default: (QWACEJST - QWACBJST) + QWACSPCP + QWACUDCP + QWACTRTE With NESTED parameter specified: QWACSPCP + QWACUDCP + QWACTRTE
MQ	IBM MQ for z/OS	SMF 116-0 IBM MQ accounting class 1	SMF:7401	(QWACEJST - QWACBJST) + (QWACESRB - QWACBSRB)
WAS	WebSphere Application Server for z/OS	SMF 120-9 WebSphere Application Server for z/OS request	SMF:7809	SM1209DA - SM1209DB - SM1209DF

Table 10. MWP transaction-level accounting: log records required and formula for calculating CPU time (continued)

Parameter of the **MWP** command that identifies the product family	Product family	Required log records	Corresponding log type and code in Transaction Analysis Workbench	Field or formula for CPU time
IMS:07	Information Management System for z/OS (IMS)	IMS log type X'07' application termination	IMS:07	DLREXTIM
IMS:56FA	IMS	IMS log type X'56FA' transaction-level accounting	IMS:56FA	TPEXTIME

For DB2 and IBM MQ, the CSV file contains class 1 CPU time, which includes application and "in DB2/MQ" CPU time.

For IMS (**IMS:07** or **IMS:56FA** parameter), the input log file must be an IMS log file (SLDS), not an IMS transaction index.

To tag transactions as being eligible for MWP, either use the **FILTER** parameter to refer to a predefined filter, or follow the **MWP** command with a **CODE** command for the corresponding log type and code and **COND** statements that specify filter conditions. For example, to tag CICS transactions that have codes beginning with "MOB":

```
MWP CICS PID(5655-xxx)
CODE(CMF:6E13)
  COND TRANCODE EQ 'MOB*'
```

Avoid double-accounting of DB2 CPU time: Transactions that run on z/OS can involve multiple subsystems. For example, CICS and IMS transactions can call DB2. In some cases, the accounting records from one subsystem can include CPU time used by another subsystem. In particular, the transaction-level accounting log records for CICS, IBM MQ for z/OS, WebSphere Application Server for z/OS, and IMS include class 1 DB2 CPU time, excluding CPU time used by a nested or WLM task. If you create a transaction-level accounting CSV file for DB2 a corresponding CSV file for a subsystem that calls DB2, then you must ensure that you do not double-account for DB2 CPU time. To collect DB2 CPU time used by a nested or WLM task only, specify the **DB2** parameter with the **NESTED** parameter.

Address space-level accounting

Use this method for address spaces that only process mobile workloads, and so the entire address space CPU time is attributable to the mobile workload. To select this method, specify the **SMF30** parameter on the **MWP** command, after the parameter that identifies the type of subsystem. For IMS, specify IMS SMF30; that is, the **IMS** parameter without a trailing log code.

If you use this method, the **MWP** command requires SMF type 30 (X'1E') common address space work records, subtypes 2 and 3; in Transaction Analysis Workbench, log type and code SMF:1E.

The SMF reporting interval must be less than or equal to an hour.

This method uses the following field-based formula to calculate CPU time:

SMF30CPT + SMF30CPS

To tag the SMF 30 records for a particular address space as being eligible for MWP, specify a filter condition that tests the value of the field that contains the job name, SMF30JBN. To specify the condition, either use the **FILTER** parameter to refer to a predefined filter, or follow the **MWP** command with a **CODE**

command for SMF 30 records and a **COND** statement that specifies the job name. For example, to tag records for address spaces that have names beginning with "DB2M":

```
MWP DB2 SMF30 PID(5615-DB2)
CODE(SMF:30-)
  COND SMF30JBN EQ 'DB2M*'
```

Chapter 43. Mobile workload tags

To take advantage of MWP, you need to tag (identify) the log records that report the CPU time for mobile workloads. A mobile workload tag is a set of one or more field values in a log record that identifies the record as belonging to a mobile workload. In Transaction Analysis Workbench, you express mobile workload tags as *filter conditions*.

To help you define mobile workload tags, browse the corresponding log files in the Transaction Analysis Workbench ISPF dialog, and use the **FILTER** command to experiment with different filter conditions. Then specify those filter conditions in **COND** statements following the **MWP** commands in your JCL.

The following **MWP** commands and subsequent **COND** statements express example mobile workload tags.

Example: Job name

```
MWP DB2 SMF30 PID(5615-DB2)
  CODE(SMF:30-)
    COND SMF30JBN EQ 'DB2M*'
```

All CPU time reported in SMF type 30 records for DB2 subsystems with job names that match the pattern "DB2M*" is eligible for MWP.

Example: Multiple transaction-level conditions

```
MWP CICS PID(5655-xxx)
  CODE(CMF)
    COND TRAN   EQ 'BANK'
    COND PHTRAN EQ 'MOB*'
    COND IPADDR EQ '172.71.7.*'
```

CPU time reported in CICS monitoring facility performance class (CMF; SMF type 110) records that meet all of the following conditions is eligible for MWP:

- The CICS transaction ID is BANK
- The CICS transaction ID of the immediately previous task in another CICS system, with which this task is associated, matches the pattern "MOB*"
- The IP address of the originating client matches the pattern "172.71.7.*"

Related reference

Filter conditions

A filter condition is an expression that selects log records based on a field value. Conditions refine filters. Without conditions, filters select log records based only on log types and codes.

Chapter 44. Creating CSV and SMF files for MWRT

To create CSV and SMF files for input to the Mobile Workload Reporting Tool (MWRT), write JCL that runs the **MWP** and **EXTRACT** commands of the Transaction Analysis Workbench report and extract utility.

You must have MWRT installed.

You need to know how to tag the log records for mobile workloads; specifically, how to express mobile workload tags as filters according to the syntax of the **CODE** command and related **COND** statement of the Transaction Analysis Workbench report and extract utility.

You need to know the program ID of each product family that you are using to process mobile workloads.

You need to know the location of the data sets containing the log records that you want to use for MWP.

For details on writing the JCL, see the syntax of the **MWP** and **EXTRACT** commands.

1. Write JCL that uses **MWP** commands to create one or more CSV files for MWRT.
2. Follow each **MWP** command with a **CODE** command and one or more **COND** statements that specify the mobile workload tag.
3. In the same SYSIN data set, include an **EXTRACT** command to create an SMF file for MWRT.
4. Submit the JCL.

The batch job creates CSV and SMF files for MWRT.

Example

The details of the **MWP** commands depend on your particular requirements. Typically, however, you should include the following **EXTRACT** command and subsequent **CODE** commands to extract the SMF records required by MWRT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SMFIN    DD DISP=SHR,DSN=SYSA.PROD.SMF(0)
//EXTRACT  DD DSN=MY.FUW.MWRT.SMF,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//* Insert DD statements here for MWP command log input and CSV output
//SYSIN    DD *

* Insert MWP commands here

EXTRACT EXTERNAL
  CODE(SMF:70-1)
  CODE(SMF:89-1)
  CODE(SMF:89-2)
/*
```

Follow the instructions in the MWRT documentation to transfer the resulting CSV and SMF files from z/OS, and then use them in MWRT.

Example: Mobile Workload Pricing using address space-level accounting, for all subsystems

This example JCL uses SMF 30 records to generate CSV files for the Mobile Workload Reporting Tool (MWRT), for all supported subsystems. The mobile workload tags, represented by **COND** statements, select the job names that are known to run only mobile workloads.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=MVS1.SMF.WEEKLY
```

```
//MWPCICS  DD  DSN=MWP.CICS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPDB2   DD  DSN=MWP.DB2.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPMQ    DD  DSN=MWP.MQ.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPWAS   DD  DSN=MWP.WAS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPIMS   DD  DSN=MWP.IMS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//EXTRACT  DD  DSN=MWP.SMF.EXTRACT,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *
MWP CICS SMF30 PID(5655-Y04) OUTPUT(MWPCICS)
CODE(SMF:30.)
COND SMF30JBN EQ 'CICSM*'

MWP DB2 SMF30 PID(5615-DB2) OUTPUT(MWPDB2)
CODE(SMF:30.)
COND SMF30JBN EQ 'DB2M*'

MWP MQ SMF30 PID(5655-W97) OUTPUT(MWPMQ)
CODE(SMF:30.)
COND SMF30JBN EQ 'MQM*'

MWP WAS SMF30 PID(5655-W65) OUTPUT(MWPWAS)
CODE(SMF:30.)
COND SMF30JBN EQ 'WASM*'

MWP IMS SMF30 PID(5635-A04) OUTPUT(MWPIMS)
CODE(SMF:30.)
COND SMF30JBN EQ 'IMSM*'

EXTRACT OUTPUT(EXTRACT) EXTERNAL
CODE(SMF:70.1)
CODE(SMF:89.1)
CODE(SMF:89.2)
/*
```

Related reference

MWP command

Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

Example: Mobile Workload Pricing using transaction-level accounting from SMF records

This example JCL uses SMF records from various subsystems to generate CSV files for the Mobile Workload Reporting Tool (MWRT). The tags, represented by **COND** statements, select transactions that match specific criteria.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=MVS1.SMF.WEEKLY
//MWPCICS  DD  DSN=MWP.CICS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPDB2   DD  DSN=MWP.DB2.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPMQ    DD  DSN=MWP.MQ.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//MWPWAS   DD  DSN=MWP.WAS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//EXTRACT  DD  DSN=MWP.SMF.EXTRACT,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *
MWP CICS PID(5655-Y04) OUTPUT(MWPCICS)
CODE(CMF:6E13)
COND TRANCODE EQ 'MOB*'
COND PORT     EQ 57556
```

```
MWP DB2 CLASS1 PID(5615-DB2) OUTPUT(MWPDB2)
CODE(DTR:003)
COND TRANCODE EQ 'MOB*'
COND USERID   EQ 'WEB*'

MWP MQ CLASS1 PID(5655-W97) OUTPUT(MWPMQ)
CODE(SMF:116.0)
COND QWHCOPID EQ 'MOB*'

MWP WAS PID(5655-W65) OUTPUT(MWPWAS)
CODE(SMF:120.9)
COND SM1209EK EQ 'ip addr=172.17.69.56 port=57556'
COND SM1209EO EQ '/MOBILE*'

EXTRACT OUTPUT(EXTRACT) EXTERNAL
CODE(SMF:70.1)
CODE(SMF:89.1)
CODE(SMF:89.2)
/*
```

Related reference

MWP command

Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

Example: Mobile Workload Pricing for IMS using transaction-level accounting

This example JCL uses IMS log records to generate a CSV file for the Mobile Workload Reporting Tool (MWRT).

For IMS, you can use either IMS type 56FA transaction-level accounting log records or IMS type 07 application termination log records.

In this example, the tags (**COND** statements) select workloads from dependent regions whose job name matches the pattern "WEBMOB*" or "WWWMOB*".

Using IMS type 56FA log records

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=IMSP.SLDS1
//MWPIMS   DD  DSN=MWP.IMS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *
IMSVRM 154
MWP IMS:56FA PID(5635-A04) LPAR(2827-0F4D7:PR01) OUTPUT(MWPIMS)
CODE(IMS:56FA)
COND TPJOBN EQ 'WEBMOB*'
COND TPJOBN EQ 'WWWMOB*'
/*
```

Using IMS type 07 log records

As for IMS type 56FA log records, but with the following details:

```
MWP IMS:07 PID(5635-A04) LPAR(2827-0F4D7:PR01) OUTPUT(MWPIMS)
CODE(IMS:07)
COND DLRNJOB EQ 'WEBMOB*'
COND DLRNJOB EQ 'WWWMOB*'
```

Notes:

- The log type and code specified by the **CODE** command matches the **IMS:07** parameter of the **MWP** command.
- The field *values* specified by the **COND** statements are the same as those for **IMS:56FA**, but the field *names* are different, reflecting the different field names in each record type.

Related reference

MWP command

Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

Part 9. Extracting logs to CSV or JSON

Transaction Analysis Workbench can extract logs from their original z/OS-based formats to comma-separated values (CSV) or JavaScript Object Notation (JSON) format for ingestion by analytics platforms and other applications.

You need to understand the requirements of your destination analytics platform or application. For example:

- Which format is more appropriate for ingestion: CSV or JSON?
- Can you configure this destination to listen for data on a TCP socket, or do you need to transfer data in a file to the local file system?
- Does this destination require certain fixed field names?
- Are field names case-sensitive at this destination?
- What time stamp formats does this destination recognize, and to what precision?

To extract logs to CSV or JSON, use the **CSV** or **JSON** command of the Transaction Analysis Workbench report and extract utility. The **CSV** and **JSON** commands can write data to a ddname, a z/OS UNIX file path, or a TCP socket.

You can write your own JCL to run the utility, or you can use the following options of the Transaction Analysis Workbench ISPF dialog to create JCL for you:

- CSV only: option 4 **Process**: enter SUB next to a log, and then select the option to convert the log to CSV.
- CSV or JSON: Option 5 **Analytics** creates JCL that forwards logs to a variety of analytics platforms. The JCL extracts the logs to either CSV or JSON, depending on the destination platform.

The JCL created by the ISPF dialog uses only a subset of the features of the **CSV** and **JSON** commands. You can tailor the JCL to use other features.

Related concepts

Log forwarding
You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

Related tasks

Forwarding logs to analytics platforms
You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 45. Differences between CSV and JSON

CSV and JSON are common data exchange formats. Inherent differences between these formats affect, and in some cases dictate, differences in the CSV and JSON output by Transaction Analysis Workbench.

Inherent differences

The following table lists some differences between CSV and JSON that are unrelated to Transaction Analysis Workbench.

Table 11. CSV versus JSON

Point of difference	CSV	JSON
Fields and field order	All records (lines) in a CSV file should have the same fields, in the same order. According to the Internet Engineering Task Force (IETF) RFC 4180, "Each line [in a CSV file] should contain the same number of fields throughout the file". However, in practice, if each record in a CSV file or stream contains, in the same column, a field that identifies the record type, then records might be *heterogeneous*: different column layouts in each record. Applications that interpret the records, such as analytics platforms, can use that field to detect the record type, and then apply a corresponding column layout definition. (*How* an application detects the record type and applies a column layout definition depends on the application.)	Each JSON object can have different fields. Field order is not significant. More precisely, a JSON object is an *unordered* set of name/value pairs; the order of name/value pairs is not significant. Name/value pairs are also known as key-value (KV) pairs.

Header	If the data records in a CSV file are *homogeneous* (all data records have the same column layout), then the first record might be a header that contains field names.
	
	The following example shows a header followed by two data records:
	
	`tran,count`
	`A,5`
	`B,10`
	
	In this context, field names are also known as labels.
		No header. Each JSON object contains field names.
		
		The following example shows two JSON objects separated by a newline character. This format is known as JSON Lines:
		
		`{"tran": "A", "count": 5}`
		`{"tran": "B", "optfld": "X", "count": 10}`

Table 11. CSV versus JSON (continued)

Point of difference	CSV	JSON
Structured fields	CSV is strictly a two-dimensional, tabular format, with no nested structures. **Tip:** To handle repeating sections in CSV output, Transaction Analysis Workbench offers a method known as *vertical separation*. For details, see Chapter 53, "Log records that contain repeating sections," on page 251.	JSON can contain nested structures. In JSON, a value can be one of several types, including an array or an object. Some z/OS-based log record types contain nested structures, such as repeating sections. If you want to preserve such structures in a common data exchange format, JSON is a better fit than CSV. For example, in the following JSON object, the value of the `steps` field is an array; each element of the array is an object. This is a simplified example to illustrate nesting, not actual log data. `{ "job": "A", "rc": 8, "steps": [ {"name": "A1", "rc": 0}, {"name": "A2", "rc": 0}, {"name": "A3", "rc": 8} ] }`
Data types	CSV does not identify data types. CSV needs external metadata to identify data types, so some analytics platforms might need more configuration to ingest CSV than JSON. **Tip:** Transaction Analysis Workbench creates metadata for DB2, Elastic (Logstash), and Hadoop. For details, see the **SCHEMA**, **LOGSTASHCONFIGURATION**, and **HCATALOG** parameters of the **CSV** command.	JSON uses JavaScript data types. For example, strings are enclosed in double quotation marks, while numbers are unquoted. If you forward logs to an analytics platform that needs to distinguish between strings and numbers, JSON already does this. Regardless of whether you use CSV or JSON, you might need to configure the analytics platform to recognize the format of date values, such as event time stamps.
Verbosity	CSV is significantly more concise than JSON.	JSON is significantly more verbose than CSV. Each value is accompanied by a field name (also known as a "key"). Disadvantage: if you forward logs to an analytics platform with a licensing model that is based on the volume of data ingested, and that calculation includes keys, then you might pay more to ingest keys than values. Advantage: reduced risk of data corruption due to incorrect field mapping, such as changes to record formats over time. Each value in each record is explicitly paired with the correct key.

Differences specific to Transaction Analysis Workbench

The first field of each CSV record or JSON object output by Transaction Analysis Workbench is the event time stamp.

In JSON output, the second field of each JSON object is the event type. Here, the terms "first" and "second" refer to the order in which Transaction Analysis Workbench *writes* these fields to an output file or stream. If you use string-based methods, such as regular expressions, to parse JSON, then you can rely on this order. However, you should not rely on a JSON parser to return fields in this order, because JSON defines objects as an *unordered* set of name/value pairs.

In CSV output, the second field (column) is the event type only if the **CSV** command that created the data specified the **TYPECOLUMN** parameter. By default, CSV records output by Transaction Analysis Workbench do not contain this event type field because, typically, all data records in a CSV file have the same event type.

By default, the event type refers to the log type and code that Transaction Analysis Workbench uses to identify the input log record. However, you can use the **TYPE** parameter of the **CSV** or **JSON** command to set your own event type.

Subsequent fields are selected from the input log record.

Here is the start of an example CSV data record, showing the event time stamp and the event type:

```
2015-12-31T13:59:00.123456Z,cmf-6e13,…
```

Here is the start of an equivalent JSON object (shown with line breaks, spaces, and indenting for readability):

```
{
  "time": "2015-12-31T13:59:00.123456Z",
  "type": "cmf-6e13",
  …
```

In JSON output, how the remaining fields are represented depends on the following parameters of the **JSON** command:

FLAT(YES|NO)
: `FLAT(YES)` writes "flat" JSON: each JSON object is a sequence of key-value pairs without nested structures. This is the default behavior. You can abbreviate `FLAT(YES)` as `FLAT`.

  `FLAT(NO)` writes JSON objects that reflect the structure of the original log records. Fields are nested in sections.

  For details, see “Flat JSON versus JSON with nested structures” on page 233.

MISSING(EXCLUDE|INCLUDE)
: Specifies whether to exclude or include fields that are selected for output but are missing from the input log record.

  `MISSING(EXCLUDE)` excludes such missing fields from the output. This is the default behavior.

  `MISSING(INCLUDE)` includes such missing fields in the output, with the JavaScript value `null`.

Finally, for both CSV and JSON output, you can optionally append correlation tokens.

Related concepts

Time stamps in CSV and JSON
CSV and JSON output by Transaction Analysis Workbench includes the time stamp of the corresponding input log record, also known as the *event* time stamp. The event time stamp is always included in the output, regardless of any **CSV** or **JSON** command parameters, such as **FIELDS**, that specify which other fields to include. Depending on the input record type and the selected fields, other fields in the output might also be time stamps. All time stamps in the output, including the event time stamp, have the same format.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 46. Flat JSON Lines for ingestion by analytics platforms

You can configure the **JSON** command with parameters to create JSON with different structures for different purposes. By default, the **JSON** command creates "flat" JSON Lines that is optimized for ingestion by analytics platforms such as Elastic or Splunk.

Example

The **JSON** command in the following JCL streams all CICS monitoring facility performance class records from the SMF data set `'SMF.MVS1.DAY'` to a remote analytics platform such as Elastic or Splunk. The analytics platform is running on a system with host name "analytics" and has been configured to listen for JSON Lines on unsecure TCP port number 6789.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=SMF.MVS1.DAY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD
OUTZONE(Z)
STREAM NAME(S1) HOST(analytics) PORT(6789) +
       TIMEFORMAT(ISO8601)
JSON STREAM(S1) CODE(CMF)
/*
```

The simplest **JSON** command requires only a single parameter, **CODE**, to identify which log type and code to extract from the input logs. In this example, `CODE(CMF)` is a synonym for `CODE(CMF:6E13)`: log type CMF, log code 6E13.

The **JSON** command in this example specifies a **STREAM** parameter, which causes the **JSON** command to write JSON data to the TCP socket specified by a **STREAM** command. By default, without a **STREAM** parameter, the **JSON** command writes JSON data to the ddname JSON. That default behavior is equivalent to specifying the parameter `OUTPUT(JSON)`.

Typically, each **JSON** command is followed by a **FIELDS** qualifying command that specifies which fields from the input log records to include in the output. For brevity, this example omits the **FIELDS** command; all fields are included in the output.

For details of other commands and the surrounding JCL in this example, see Chapter 47, “Streaming logs as JSON Lines over TCP to an analytics platform,” on page 235.

Default JSON command parameters

The **JSON** command in the previous example implicitly specifies the following default parameters:

```
LINES +
FLAT +
MISSING(EXCLUDE) +
FIELDCASE(LOWER) +
ASCII +
EOL(LF)
```

The following list describes how each default parameter value configures the JSON output for ingestion by analytics platforms:

LINES
: Outputs JSON Lines. Each input log record is converted to a JSON object on a separate line. JSON Lines is typically easier for analytics platforms to ingest than the alternative, specified by the **ARRAY** parameter, which is a JSON array where each element represents an input log record.

FLAT
Outputs JSON with no nested structures. All fields are first-level properties. For example, to refer to a field in a Splunk search, you simply use the field name, such as `response`, rather than a period-separated concatenation of antecedent property names that ends in the field name, such as `data.dfhtask.response`.

MISSING(EXCLUDE)
Reduces data volume by excluding fields that are selected for output but are missing from the input log record. The alternative, `MISSING(INCLUDE)`, includes such missing fields in the output, and assigns them the JavaScript value `null`.

FIELDCASE(LOWER)
Outputs field names in lowercase. Field names in analytics platforms are often case-sensitive. Using single-case names avoids "field not found" errors caused by mistyping mixed-case names.

ASCII
Encodes output in ASCII rather than EBCDIC.

The default encoding of output from the **JSON** command is normally EBCDIC.

However, if a **JSON** command specifies a **STREAM** parameter, as in this example, then the default encoding is ASCII.

To explicitly specify EBCDIC encoding, use the **EBCDIC** parameter.

EOL(LF)
Outputs a line feed (LF) character as the end-of-line (EOL) delimiter for the JSON Lines.

The default behavior of the **JSON** command is normally `EOL(NONE)`: no delimiter. This default behavior is appropriate for writing to record-oriented MVS data sets.

However, if a **JSON** command specifies a **STREAM** parameter, as in this example, then the default EOL delimiter is a line feed character.

`EOL(LF)` is compatible with the Logstash `json_lines` codec and the default value of the Splunk **LINE_BREAKER** attribute.

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

JSON versus JSON Lines

In some cases, you might need to extract logs as JSON Lines, where each line is a JSON object. In other cases, you might need JSON that can be parsed by functions that only accept a single JSON object or array.

JSON Lines is more useful than a JSON array in the following cases:

- Forwarding many thousands of log records to an analytics platform. Large arrays can be problematic (memory-intensive) to parse. For many analytics platforms, JSON Lines is inherently easier to ingest.
- If you want several **JSON** commands to write to the same destination, such as a TCP socket, then you *must* output JSON Lines.

JSON Lines

By default, the **JSON** command of the Transaction Analysis Workbench report and extract utility outputs JSON Lines. Each line of output is a JSON object that represents an input log record:

```
{ record 1 }
{ record 2 }
{ record 3 }
...
```

JSON: A single array

If you specify the **ARRAY** parameter, the **JSON** command writes a single JSON array. Each element of the array is a JSON object that represents an input log record:

```
[
  { record 1 },
  { record 2 },
  { record 3 },
  ...
]
```

An array is useful in contexts that require a single JSON object, such as the JavaScript `JSON.parse()` method in web browsers.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

"Flat" JSON versus JSON with nested structures

The structure of some z/OS-based log records can be complex. Records can be divided into sections that repeat, and sections can contain subsections. In some cases, you might need to convert log records to JSON that retains the original nested structures. In other cases, you might prefer "flat" JSON, where each input record is represented as a flat sequence of key-value (KV) pairs, without nested structures.

"Flat" JSON

By default, the **JSON** command writes "flat" JSON: JSON without nested structures. All fields are first-level properties, so you can refer to them simply by name. You don't need to qualify field names with any antecedent names. For example (shown with line breaks and indenting for readability):

```
{
  "time": "2015-12-31T13:59:00.123456Z",
  "type": "smf-xx",
  "smfxxa1": 1.3,
  "smfxxa2": "MYID",
  "smfxxb1": 2.4
}
```

Flat JSON is useful when you need KV pairs but you don't want nested structures. Some analytics platforms ingest KV-pair format data faster than delimited data formats such as CSV, because KV pairs enable field extraction to be reliably deferred until search time; the correct field names are embedded in the raw ingested data.

JSON with nested structures

If you specify the FLAT(NO) parameter, the **JSON** command writes JSON with nested structures. Log data is nested in a data property. Inside the data property, fields are nested in sections. For example:

```
{
  "time": "2015-12-31T13:59:00.123456Z",
  "type": "smf-xx",
  "data": {
    "dsecta": {
      "smfxxa1": 1.3,
      "smfxxa2": "MYID"
    }
    "dsectb": {
      "smfxxb1": 2.4
    }
  }
}
```

where:

- dsecta and dsectb are data sections (DSECTs) in the z/OS assembler mappings of the original log records
- smfxxa1, smfxxa1, and smfxxa3 are field names

In some contexts, nested structures in JSON are useful. For example:

- Encapsulating event-specific data in a common data property can simplify parsing. All objects have the same high-level properties, including time, type, and data. You can use the value of the type property to determine how to parse the contents of the data property.
- Nesting fields in section properties enables you to reproduce the original log record structure; for example, if you want to present log data in a user interface with fields arranged under section headings.

However, in other contexts, nesting can have undesirable consequences, such as having to qualify field names with their antecedent property names, requiring you to remember which section each field belongs to. For example, referring to a response time field might mean using the verbose field reference data.dfhtask.response instead of just the field name response.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 47. Streaming logs as JSON Lines over TCP to an analytics platform

You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

You must configure the analytics platform to listen on a TCP port for JSON Lines. You need to know the analytics platform host name (or IP address) and port number.

For secure TCP, you need to know the corresponding security details that Transaction Analysis Workbench will require: for example, the location on z/OS of the security certificates, such as a SAF key ring.

You need to know the location on z/OS of the logs that you want to forward.

Streaming gets data off z/OS without the storage and processing costs of writing data to temporary staging files on z/OS, and then transferring those files off z/OS.

You can configure some analytics platforms, such as Elastic and Splunk, to listen on a TCP port and ingest data as it arrives.

You can configure some log forwarding tools, such as Logstash (from the Elastic Stack), to listen on a TCP port and forward data to a variety of destinations. The destinations can include analytics platforms that do not natively ingest data via TCP. For example, you can configure Logstash to receive JSON Lines input over TCP, and then forward the data in a different, platform-specific output format over HTTP, to Elasticsearch or other destinations. Such log forwarding tools expand the range of destinations for data from Transaction Analysis Workbench.

Many analytics platforms ingest data in JSON Lines format. In some cases, a "key-value (KV) pair" data format such as JSON Lines is the *preferred* format.

To stream logs in JSON Lines format to a TCP socket, use the **JSON** and **STREAM** commands of the Transaction Analysis Workbench report and extract utility.

You can either write JCL yourself to run these commands, or you can use Transaction Analysis Workbench ISPF dialog option 1 **Stream** to create the JCL for you.

The dialog option offers only a small subset of the data sources and record types supported by Transaction Analysis Workbench. You can tailor the JCL created by the dialog, or write JCL yourself, to forward data from *any* of the data sources and record types supported by Transaction Analysis Workbench.

The following procedure uses the dialog option to create JCL. If you prefer to write JCL yourself, skip to the example JCL following the procedure.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 5 **Analytics**.
2. Select a data source: SMF or IMS.
   a) Enter the data set name of the input SMF data set or IMS log.
   b) For SMF, select one or more record types.

   The JCL will contain a **JSON** command for each selected record type.
3. Select option 1 **Stream**.
4. Enter the host name and the port number on which the analytics platform (for example, Logstash or Splunk) is listening.
5. Optionally, enter a timeout value.
6. If the analytics platform TCP port uses security (TLS), enter the security details.

   For more information about the security details, press the Help function key (F1) or see the corresponding security parameters of the **STREAM** command.

7. Enter SUB on the command line to create the JCL.

   The JCL is displayed in an ISPF edit panel.

Example

The following JCL extracts selected fields from CICS monitoring facility performance class records in a dumped SMF data set, converts them to JSON Lines in ASCII, and then forwards them over TCP in a single stream to a listening analytics platform, such as Elastic (specifically, Logstash) or Splunk. The analytics platform is installed on a computer with host name "analytics", listening on unsecure TCP port number 6789.

```
//UIDFUW   JOB NOTIFY=&SYSUID  1
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK  2
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
OUTZONE(Z)  3
STREAM NAME(STREAM1) HOST(analytics) PORT(6789) +
       TIMEFORMAT(ISO8601)  4
JSON CODE(CMF) STREAM(STREAM1)
FIELDS(  5
       TRAN
       USRCPUT
       /* More fields...
      )
/*
```

Notes on the JCL:

1

If you use the ISPF dialog to create JCL, then the job statement is specified by the **Job Statement Information** field under dialog option 0.1 **Workbench Personal Settings**.

2

If you use the ISPF dialog to create JCL, then the Transaction Analysis Workbench executable load module library (STEPLIB) is specified by the **Workbench Load Library** field under dialog option 0.1 **Workbench Personal Settings**.

3

The `OUTZONE(Z)` command causes time stamps to be output in UTC. The `TIMEFORMAT(ISO8601)` parameter on the subsequent **STREAM** command causes time stamps to be output in ISO 8601 format. Hence, the time stamps in the output have a trailing Z zone designator, indicating UTC.

4

Some parameters of the **JSON** command, such as **TIMEFORMAT**, can be specified on the **STREAM** command, ensuring consistency between **JSON** commands that write to the same stream.

5

The **FIELDS** command is optional. If you omit the **FIELDS** command, the output includes all fields from the selected records.

Here is a line from the output stream:

```
{"time":"2015-11-30T08:00:00.000001Z","type":"cmf-6e13","tran":"TRNA","usrcput":0.003456,…}
```

For a secure TCP port, add the following parameters to the **STREAM** command in the previous JCL listing:

```
SECURITY(TLS*) FIPS KEYRING(my/fuw.stream)
```

where `my/fuw.stream` identifies the RACF key ring that contains the CA certificate used by Splunk or Logstash.

You can forward multiple record types in the same stream. For example, to add SMF type 30 job termination records to the stream, append the following lines to the SYSIN data set:

```
JSON CODE(SMF:30.) STREAM(STREAM1)
FIELDS(
       SMF30JBN
       SMF30CPT
       /* More fields...
      )
CODE(SMF:30.)
  COND SMF30STP EQ 5 /* Filter to select job termination records
```

Here is a corresponding output line:

```
{"time":"2015-11-30T08:00:01.000002Z","type":"smf-30","smf30jbn":"PRDJOB1","smf30cpt":0.12300
6,…}
```

The ISPF dialog creates JCL that forwards logs from a single data source, such as a single SMF data set. You can tailor that JCL, or write JCL yourself, to forward logs from multiple data sources. For example, to add IMS transaction index records to the stream in the previous example JCL, add a **DD** statement that refers to an IMS system log data set (SLDS):

```
//LOGIN    DD DISP=SHR,DSN=IMSP.SLDS
```

then append the following lines to SYSIN:

```
IMSVRM=154
IMSINDEX
JSON CODE(IMS:CA01) STREAM(STREAM1)
FIELDS(
       TRANCODE
       PROGRAM
       USERID
       IMSID
       /* More fields...
      )
```

where 154 identifies the release level of the IMS system.

Each **CSV** or **JSON** command can only write to a single stream. However, you can specify multiple **STREAM** commands in a SYSIN data set; **CSV** and **JSON** commands in the same SYSIN data set can write to different streams.

Related concepts

Log forwarding

You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

Forwarding logs to Elastic

You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a TCP network to the Elastic Stack.

Forwarding logs to Splunk

You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format to a TCP data input on a remote Splunk indexer.

Related tasks

Adding a CA certificate to a RACF key ring

To connect to a secure TCP port on a remote system, you need the CA certificate used by that system. If you store certificates in RACF key rings, you need to add that CA certificate to a key ring.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Flat JSON Lines for ingestion by analytics platforms
You can configure the **JSON** command with parameters to create JSON with different structures for different purposes. By default, the **JSON** command creates "flat" JSON Lines that is optimized for ingestion by analytics platforms such as Elastic or Splunk.

STREAM command
Forwards log data in JSON Lines or CSV format to a TCP socket.

Chapter 48. Time stamps in CSV and JSON

CSV and JSON output by Transaction Analysis Workbench includes the time stamp of the corresponding input log record, also known as the *event* time stamp. The event time stamp is always included in the output, regardless of any **CSV** or **JSON** command parameters, such as **FIELDS**, that specify which other fields to include. Depending on the input record type and the selected fields, other fields in the output might also be time stamps. All time stamps in the output, including the event time stamp, have the same format.

Field name of the event time stamp

The field name of the event time stamp is `time`, **Time**, or **TIME**, depending on the value of the **FIELDCASE** parameter of the **CSV** or **JSON** command, and whether the output format is CSV or JSON:

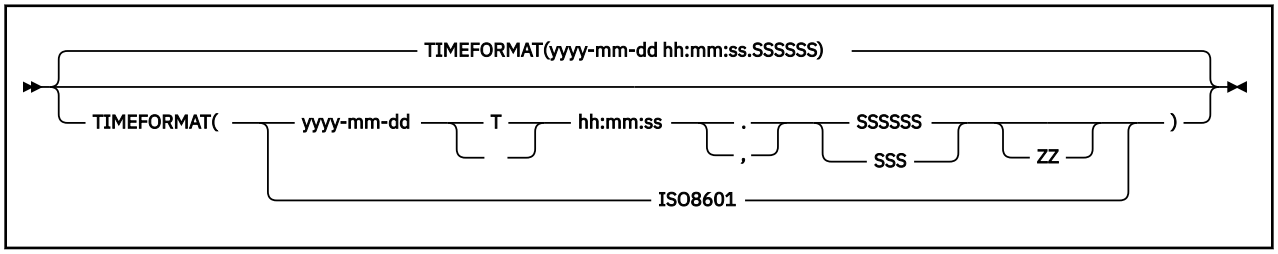
Table 12. Time stamp field name in CSV or JSON output

FIELDCASE	CSV field name (label)	JSON field name (property, key)
LOWER (default)	`time`	`time`
UPPER	`TIME`	`TIME`
ASIS	`Time`	`time`

In CSV output, the event time stamp is the first column in each row.

Format

The format of time stamps in CSV or JSON output is specified by the **TIMEFORMAT** parameter of the **CSV** or **JSON** command.



where:

yyyy-mm-dd
: Represents a calendar date.

hh*:*mm*:*ss
: Represents a time of day in the 24-hour timekeeping system.

S
: (Capital S.) Represents a digit in a fraction of a second.

ZZ
: Represents a zone designator in ISO 8601 extended format: as a UTC offset *hh*:*mm* or the UTC designator Z.

The allowed formats, while modeled on pattern syntax, are restricted to the permutations shown in the syntax diagram.

The default format is the JDBC string representation of a local time stamp with microsecond precision (no zone designator; fractions of a second to six decimal places):

```
yyyy-mm-dd hh:mm:ss.SSSSSS
```

For example:

```
2015-12-31 13:59:00.123456
```

The default format *almost* matches the ISO 8601 international standard. There is one difference: the default JDBC format separates the date and time components with a space; ISO 8601 uses the character T.

You can use the **TIMEFORMAT** parameter to change the default format in the following ways:

- Separate the date and time components with the character T, to match ISO 8601.
- Restrict precision to milliseconds: fractions of a second to three decimal places, instead of the default six.
- Append a zone designator.
- Use a comma (,) instead of a period (.) as the decimal sign ("point") that separates the integer number of seconds from the decimal fraction.

  If you use a comma as the decimal sign with the **CSV** command, then you cannot use a comma to delimit fields; in this case, the **CSV** command uses a semicolon as the default field delimiter. To specify a different delimiter, use the **DELIMITER** parameter of the **CSV** command.

`TIMEFORMAT(ISO8601)` matches the ISO 8601 date and time of day representation extended format with microsecond precision and a zone designator. The value `ISO8601` is equivalent to the following pattern-based value:

```
yyyy-mm-ddThh:mm:ss.SSSSSSZZ
```

For example:

```
2015-12-31T13:59:00.123456+08:00
```

Precision

By default, time stamps in CSV or JSON output include fractions of a second to microsecond precision; six decimal places.

Consider which applications will use the CSV or JSON data, and check the format and precision of the data types that those applications use for time stamps. Some applications support time stamps only to millisecond precision; three decimal places. To restrict time stamps in the output CSV or JSON data to milliseconds, specify a **TIMEFORMAT** value with SSS instead of the default SSSSSS precision.

Time zone

By default, time stamps in CSV or JSON output are local times; they do not include a trailing *zone designator* that identifies their time zone.

Typically, without specific configuration, applications interpret time stamps without zone designators as local times according to the time zone of the system on which that application is running. If you forward logs to an application in a different time zone, time stamps without a zone designator might get corrupted; the application might interpret the time stamps using the wrong time zone.

To append a zone designator to each time stamp, specify a **TIMEFORMAT** value with a trailing ZZ.

The zone designator is a UTC offset in the ISO 8601 extended format: +*hh*:*mm* or -*hh*:*mm*, with one exception: if the time zone is UTC, the zone designator is the character Z.

The value of the zone designator depends on two global commands: **ZONE** and **OUTZONE**.

Related concepts

Differences between CSV and JSON

CSV and JSON are common data exchange formats. Inherent differences between these formats affect, and in some cases dictate, differences in the CSV and JSON output by Transaction Analysis Workbench.

Event time stamps versus log record time stamps
Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

ZONE command
Specifies the time zone that Transaction Analysis Workbench uses for the default time zone, the output time zone, and the time zone for the **START** and **STOP** commands.

OUTZONE command
Specifies the time zone of time stamps in output CSV and JSON data.

Chapter 49. Which log records to extract to CSV or JSON

You can extract to CSV or JSON any of the log record types supported by Transaction Analysis Workbench. You must decide which records to extract according to your specific needs.

Each **CSV** or **JSON** command processes log records of one specific combination of log type and code. To specify the log type and code, you must use either a **CODE** parameter or a **FORM** parameter.

Transaction accounting records for different subsystems

Transaction accounting is a typical use case for extracting logs to CSV or JSON. You can use Transaction Analysis Workbench to extract accounting records for DB2, CICS, IMS, IBM MQ, and WebSphere Application Server (WAS) for z/OS.

Every transaction can be accounted for, with every piece of performance detail available for analysis. You can use correlation tokens to match the work done by each subsystem on behalf of the transaction.

Table 13. Transaction accounting records for different subsystems

Subsystem	**Accounting record type**	**Transaction Analysis Workbench log *type*:*code***	**Description**
DB2	SMF 101	`DTR:003`	DB2 accounting records (IFCID 003) are cut when accounting classes 1, 2, 3, 7, and 8 are activated.
CICS	SMF 110	`CMF:6E13`	CICS monitoring facility (CMF) performance class records are cut to SMF when the performance monitor is started.
IMS	Not applicable	`IMS:CA01`	IMS does not write a specific accounting record. Transaction Analysis Workbench creates *IMS transaction index* records from a pass of the IMS log. Transaction Analysis Workbench processes the resulting records in the same way as any other supported record type.
MQ	SMF 116	`SMF:74`	The MQ accounting records are cut to SMF when accounting class 1 (brief) or 3 (detailed) is activated.
WAS	SMF 120.9	`SMF:7809`	You can request the inbound request activity record at WAS startup.

Filtering which records to extract based on a time range

By default, the **CSV** and **JSON** commands select all records of the specified type and code from the input logs.

To filter input log records by event time stamp, use the **START** and **STOP** commands.

Filtering which records to extract based on field values

To filter input log records based on field values, follow the **CSV** or **JSON** command with one or more **CODE** commands and **COND** statements.

Limiting the number or volume of records to extract

You can restrict the amount of data output by a **CSV** or **JSON** command in either of two ways:

- To restrict the *number of output records*, specify the following parameter on the **CSV** or **JSON** command:

  ```
  OUTLIM(n)
  ```

  where *n* is the maximum number of records.
- To restrict the *volume of output data*, specify the following two parameters:

  ```
  OUTLIM(n) MB
  ```

  where *n* is the maximum integer number of megabytes.

Related concepts

Token fields to correlate activity within and between subsystems
When extracting logs to CSV or JSON format, you can optionally append token fields that enable you to correlate activity for each transaction within and between subsystems.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

START, STOP (FROM, TO) commands
Specify the report interval. Only log records with an event time stamp within the report interval are included in the output.

CODE command and COND statement
The **CODE** command filters log records by including or excluding them based on their log type and code. To refine the filter based on field values in a log record, follow the **CODE** command with one or more **COND** statements.

Chapter 50. Which fields from the input log records to include in CSV or JSON output

You must decide which fields from the input log records you want to include in the CSV or JSON output.

By default, the **CSV** and **JSON** commands select all fields from the input log records. This is typically more than you need, unless you plan to use the CSV or JSON for the type of "deep dive" analysis that you can perform using the log browser in the Transaction Analysis Workbench ISPF dialog.

For a few record types, ISPF dialog option 5 **Analytics** provides a curated list of fields as a suggested starting point.

Tip: To list all field names for a log code, create a form: on the primary option menu of the Transaction Analysis Workbench ISPF dialog, select option 2 **Forms**. Enter **NEW** on the command line. Specify the log type and code, and then press Enter. A new form is displayed, containing all fields in that log code. To exit without saving the new form, press the Cancel function key (F12).

Selecting fields individually by name

To select fields to include in CSV or JSON output, use either of the following methods:

- Immediately after the **CSV** or **JSON** command, insert a **FIELDS** command that lists the field names.
- Create a form, and then use the **FORM** parameter of the **CSV** or **JSON** command to refer to the name of the form.

Selecting all numeric fields

In some log record types, most or all numeric fields might be useful for performance analysis, while only some non-numeric fields might be useful for this purpose. For these log record types, it can be useful to include all numeric fields in the output, and then select non-numeric fields individually by name as required.

Including all numeric fields without naming them individually is useful when numeric fields are frequently added to a log record type to reflect new features in the system that generate those logs. New numeric fields are automatically included in output.

To include all numeric fields without naming them individually, specify the **ALLNUMBERS** parameter on the **CSV** or **JSON** command.

To select additional, non-numeric, fields by name, use the **FIELDS** command or **FORM** parameter.

Writing bit flags as separate fields

Some fields contain bit flags, where each bit conveys a different meaning. Each bit flag has its own field name.

To write bit flags as separate fields in the output, use the **FIELDS** command to explicitly specify the bit flag field names.

To list bit flag field names, use the log browser in the Transaction Analysis Workbench ISPF dialog to zoom on the corresponding parent field.

The value of an individual bit field is output as `true` or `false` for JSON, 1 or 0 for CSV. If the value is an accumulation of multiple instances of that field from repeating sections, the value is an integer that represents the number of instances where the bit flag is on.

Excluding missing fields from JSON output

By default, the **JSON** command excludes fields that are selected for output but are missing from the input log record.

This default behavior reduces data volume, which is especially useful if the pricing plan of your analytics platform is based on data volume.

To always include all fields that are selected for output, even when fields are missing from an input log record, specify the `MISSING(INCLUDE)` parameter on the **JSON** command. Fields that are missing from the input log record will have the JavaScript value `null`.

If you specify `MISSING(INCLUDE)`, be aware that some JSON parsers interpret the value `null` as the string value `"null"`. That behavior is typically undesirable. For example: if you use numeric functions to process the field, then a string value is inappropriate, and might cause errors.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 51. Field names in CSV and JSON output

The default field names in CSV and JSON output are from the corresponding Transaction Analysis Workbench knowledge module for each record type. There is one exception: for CICS monitoring facility (CMF) performance records, the default field names in CSV and JSON output are from the CICS dictionary. You can override the default field names, with some specific exceptions. You can also control the case of field names in CSV and JSON output.

Overriding default field names

If you use the **FIELDS** command to specify fields, then you can override default field names by following each field name with a colon and an alias. For example:

```
FIELDS(
  SMF30CPU:"CPU time"
)
```

Forms do not support aliases.

CICS monitoring facility (CMF) performance fields

For CMF performance class (SMF type 110, subtype 1, class 3) records, rather than using the knowledge module field names, the default field names in CSV and JSON output are the informal names, also known as nicknames, from the CICS dictionary. If the informal name is not unique, Transaction Analysis Workbench substitutes a unique name.

In the **FIELDS** parameter of a `CSV CODE(CMF)` or `JSON CODE(CMF)` command, you can specify the knowledge module field names, the informal names, or a mix of both. Regardless of which names you specify in the **FIELDS** parameter, the output uses the informal names, unless you explicitly override the default field names with aliases.

For a list of informal names in the default CICS dictionary entries, see the CICS product documentation.

For all other record types, the field names output by the **CSV** and **JSON** commands are the same field names used throughout Transaction Analysis Workbench, from the corresponding Transaction Analysis Workbench knowledge module.

For example, for the following CMF performance field:

CMF group and field ID	Field name in the Transaction Analysis Workbench knowledge module	Informal name (nickname) in the default CICS dictionary entries
DFHTASK S008	UserCPU	USRCPUT

- **FIELDS(UserCPU)** and **FIELDS(USRCPUT)** have the same effect: the output CSV field label or JSON property name will be the informal name, USRCPUT
- **FIELDS(UserCPU:CPU)** and **FIELDS(USRCPUT:CPU)**, both specifying the alias CPU, also have the same effect: the output CSV field label or JSON property name will be the alias, CPU

Fixed field names that you cannot override

You cannot override the following field names:

- In CSV and JSON output: the field name of the event time stamp. For details, see Chapter 48, “Time stamps in CSV and JSON,” on page 239.
- In JSON output:
  - `type`: the log record (or "event") type.

- `data`: a container for log data. Only applies if you specify the `FLAT(NO)` parameter on the **JSON** command.
- Correlation tokens.

Field names in lowercase, uppercase, or "as is"

To control the case of field names in CSV and JSON output, including fixed field names, specify `FIELDCASE(LOWER)`, `FIELDCASE(UPPER)`, or `FIELDCASE(ASIS)` on the **CSV** or **JSON** command.

The default is `FIELDCASE(LOWER)`: field names are output in all lowercase.

`FIELDCASE(ASIS)` uses the field names as defined in the knowledge modules, as displayed by the Transaction Analysis Workbench ISPF dialog log browser. These field names can be uppercase, lowercase, or mixed case, depending on each knowledge module.

The order of fields in output is fixed

The order of fields in the output CSV or JSON matches the order in the corresponding knowledge module. Typically, the order of fields in a knowledge module matches the order in the original log record.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 52. Field types and values

When Transaction Analysis Workbench captures and processes packets, it can extract various types of data from packets. The types of data can be time stamps, elapsed times, floating-point values, bit fields, strings, and byte flags.

Transaction Analysis Workbench represents each of these data types as follows:

Time stamps
Transaction Analysis Workbench represents time stamps as the number of seconds elapsed since January 1, 1970, 00:00:00 UTC. Transaction Analysis Workbench displays time stamps in a human-readable format, such as YYYY-MM-DD HH:MM: SS.SSS, where YYYY is the year, MM is the month, DD is the day, HH is the hours, MM is the minutes, SS is the seconds, and SSS is the milliseconds.

Elapsed times
Transaction Analysis Workbench represents elapsed times as the number of seconds elapsed between two time stamps. The format of elapsed time is HH:MM: SS.SSS, where HH is the number of hours, MM is the number of minutes, SS is the number of seconds, and SSS is the number of milliseconds.

Floating-point values
Transaction Analysis Workbench represents floating-point values as binary numbers in IEEE 754 format. Transaction Analysis Workbench can also display floating-point values in decimal format with a specified number of decimal places.

Bit fields
Transaction Analysis Workbench represents bit fields as a series of 1s and 0s, where each bit represents a specific field.

Strings
Transaction Analysis Workbench represents strings as a series of characters. Transaction Analysis Workbench can also display strings in ASCII or Unicode format.

Byte flags
Transaction Analysis Workbench typically represents byte flags as a series of 1s and 0s, where each bit represents a specific flag. Transaction Analysis Workbench can also display byte flags in hexadecimal format.

Chapter 53. Log records that contain repeating sections

When converting logs to CSV or JSON, Transaction Analysis Workbench offers several methods for handling *repeating sections*: groups of fields that occur more than once in a single record.

Many log records have simple, flat structures where each field occurs only once. Such flat structures translate easily to the two-dimensional CSV format. However, some log records contain repeating sections, where each instance of a repeating section contains its own values for the same set of fields.

Transaction Analysis Workbench offers several methods for handling repeating sections. To choose a method, you specify the following section parameters of the **CSV** or **JSON** command:

Table 14. Methods for handling log records that contain repeating sections

Method	Supported output formats	Description	Section parameter
Merging	CSV and JSON	The separate field values from each instance of a repeating section are merged into a single value according to the following rules: • Character fields are concatenated • Numeric fields (times and counts) are accumulated • Timestamp fields take the value from the first instance of the repeating section	**SECTIONS** or **EXCLUDESECTIONS**
Vertical separation	CSV and JSON	Each instance of a repeating section is written to a separate output record. The field values of any other sections in these output records are fixed, and are repeated in each output record. Only one section can be vertically separated in a CSV or JSON output file. If a log record contains multiple sections that can repeat, not just multiple instances of a single section, and you want to vertically separate the values of each of those sections, then you need to generate an output file for each section. For example, DB2 system statistics trace (IFCID 001) records contain the repeating sections QWSA and QWSB. To extract these records to CSV files and maintain each instance of the repeating sections as separate values, you need to create one CSV file for the QWSA section and another for QWSB. Typically, you would also create a third CSV file, without vertical separation, containing only non-repeating sections. To correlate the records in the different files, use the **TOKENS** parameter to append token fields to each record.	**VERTICALSECTION**

Table 14. Methods for handling log records that contain repeating sections (continued)			
Method	**Supported output formats**	**Description**	**Section parameter**
Horizontal separation	JSON only	Each instance of a repeating section is represented as an array element.	**HORIZONTALSECTIONS**

Related reference

Section parameters

The section parameters of the **CSV** and **JSON** commands specify which sections of a log record to include in the output, and how to handle repeating sections.

Chapter 54. Scrambling character field values in CSV and JSON output

When converting logs to CSV or JSON, you can specify which character fields to scramble. Scrambling a field makes its original value unrecognizable.

Scrambling substitutes characters in the ranges 0-9, a-z, and A-Z with a different character from the same range. For example, the original field value "bB9" might become "pP5".

Scrambling changes each character to a different, fixed, and unique value. Field values that were unique in the unscrambled input remain unique in the scrambled output.

Scrambling is useful for creating presentation material from sensitive data.

Important: This scrambling method is reversible. Do not use this method to scramble data for security or compliance purposes.

1. Create JCL to convert logs to CSV or JSON.
2. Use the **FIELDS** qualifying command to specify which fields to include in the output.
3. For each character field that you want to scramble, insert the keyword **SCRAMBLE** after the field name and any label.

   For example:

   ```
   FIELDS(

     SMF30RUD SCRAMBLE

   )
   ```

   SCRAMBLE is ignored for non-character fields.

The following JCL reads SMF type 30 records for job terminations from a dumped SMF data set, selects three fields, renames them, scrambles the user ID, and then writes the data as CSV to SYSOUT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV      DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(SMF:30.) LABELS FIELDCASE(ASIS)
FIELDS(
  SMF30RUD:"User ID" SCRAMBLE /* Sensitive
  SMF30JBN:"Job name"         /* Not sensitive: report original value
  SMF30CPT:"CPU time"
)
CODE(SMF:30.)
  COND SMF30STP EQ 5          /* Job termination
/*
```

Related reference

FIELDS command

Specifies which fields to include in the output of the **CSV** and **JSON** commands.

Chapter 55. Token fields to correlate activity within and between subsystems

When extracting logs to CSV or JSON format, you can optionally append token fields that enable you to correlate activity for each transaction within and between subsystems.

To request correlation tokens, specify the **TOKENS** parameter of the **CSV** or **JSON** command. The tokens are in character format because they are a composite of two or more character, hexadecimal, or integer field values.

The following table describes the tokens and when to use them for correlation:

Table 15. Tokens to correlate activity within and between subsystems

Token field name	Source	Connect to	Description	DB2 schema
ACCTOKEN	CICS	DB2	The accounting token is a composite token that consists of two fields: the network name and the network unit of work ID. For example: `FTS3.SC0TCP06.6F59D4A671B6`	CHAR(0030)
IMSTOKEN	IMS	DB2, CICS, MQ	The IMS recovery token is a composite token that consists of two fields: the subsystem ID and the unit of work ID. For example: `IDDG/0000000400000000`	CHAR(0025)
LUWID	DB2, MQ	DB2, MQ	The logical unit of work ID is a composite token that consists of four fields: the network ID, the LU name, the uniqueness value, and the commit count. For example: `FTS3/DBA6LU/CC17F01EF3DA/2` Not required for accounting correlation. Used in DB2 to correlate accounting with the DB2 log.	CHAR(0036)

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 56. JSON-format metadata that describes the CSV or JSON data

When extracting logs to CSV or JSON, you can optionally request metadata that describes the fields in the CSV or JSON output. This metadata is in JSON format.

You can use the metadata to programmatically parse the CSV or JSON data. For example, as you traverse the JSON data, you can look up each field in the metadata to get its field type and a short description.

To request this metadata, specify the **METADATA** parameter of the **CSV** or **JSON** command.

The format of this metadata is specific to Transaction Analysis Workbench. To request metadata for use with DB2, Elastic (Logstash), or Hadoop, use the **SCHEMA**, **LOGSTASHCONFIGURATION**, or **HCATALOG** parameter.

Format

For each field in the corresponding CSV or JSON output, the metadata contains a property name/value pair. The property name is the field name. The property value is an object that contains the following properties:

type
The field type:

SECTION
The start of a new section in the record; colloquially referred to as a DSECT name in z/OS.

CHAR
Character string.

HEX
Hexadecimal character format, used when the field is not human readable or requires programmatic interpretation.

FLAG
Flag bytes represented as a hexadecimal character string.

INTEGER
Count field.

DECIMAL
Duration in seconds, with fractions of seconds after the decimal point and a maximum precision of one microsecond.

TIMESTAMP
Date and time field.

STCK
MVS STCK (store clock). This type occurs only under exceptional circumstances, when a field is not populated in any of the input records; for example, when the input records were generated by an earlier release of software that writes only a subset of the fields generated by later releases. Normally, Transaction Analysis Workbench interprets STCK values, and then assigns either of the more specific types DECIMAL (elapsed/CPU time) or TIMESTAMP.

TOKEN
A correlation token in character format.

section
The name of the section in which the field occurs, or `null` if the field is not in a section.

sequence
An integer that identifies the position, starting from 1, of the field in the **FIELDS** parameter of the **CSV** or **JSON** command.

The log record time stamp field has no **sequence** property.

description
A short description of the field.

Example

The following example snippet of metadata describes an integer field named DBCALLS in the DATABASE section of a log record:

```
"DBCALLS":
{
  "type": "INTEGER",
  "section": "DATABASE",
  "sequence": 6,
  "description": "Database call count"
}
```

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Part 10. Analyzing system and subsystem instrumentation

These topics provide tips for using Transaction Analysis Workbench to analyze some types of log records from specific systems and subsystems.

These tips do not cover all of the log record types supported by Transaction Analysis Workbench.

Related concepts

Configuring subsystems to collect data for analysis
Some subsystems, such as DB2 and IMS, write log records without requiring additional configuration. However, for other subsystems, or for specific types of log record such as performance, accounting, or trace-related data, you need to configure the subsystems to write log records.

Related reference

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Chapter 57. CICS instrumentation

You can use Transaction Analysis Workbench to analyze CICS monitoring facility (CMF) performance class records (SMF type 110 records), and CICS trace entries that have been written to z/OS generalized trace facility (GTF) data sets.

CICS performance records

Typically, the last two digits of a 4-digit hexadecimal log code identify the record subtype. However, CMF records have several classes, each with its own record format. The third digit of a CMF log code identifies the subtype, and the fourth digit identifies the class. Hence, the log code 6E13 identifies CMF performance class data (SMF type 110, subtype 1, class 3).

In Transaction Analysis Workbench, CMF records belong to the specific log type CMF, whereas other SMF records belong to the log type SMF.

Related concepts

Event time stamps versus log record time stamps
Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Related reference

CMF: CICS monitoring facility codes
The CMF log type consists of log codes that identify MVS System Management Facilities (SMF) type 110, subtype 1 records written by the CICS monitoring facility.

CICS trace

You can specify CICS auxiliary trace data sets or z/OS generalized trace facility (GTF) data sets as log files and analyze CICS trace entries in those files.

In Transaction Analysis Workbench, CICS trace entries are known as log type CTR.

Related concepts

Trace levels versus filters
Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Chapter 58. CICS-DBCTL transactions

You can analyze response time problems in CICS-DBCTL transactions using CICS monitoring facility (CMF) performance class records (SMF type 110 records), IMS log records, or a combined view of both.

CICS-DBCTL transactions are CICS transactions that issue DL/I requests to access IMS databases. As for any CICS transaction, you can analyze CICS-DBCTL transactions using CMF records. However, CMF records do not contain all of the details of the IMS events that are triggered by DL/I requests. Those details are in the IMS log records. Transaction Analysis Workbench enables you to analyze CICS-DBCTL transactions using CMF records, IMS log records, or a combined view of both. In particular, Transaction Analysis Workbench enables you to analyze *exceptions*: CICS-DBCTL transactions that abended or had a response time greater than a duration that you specify.

Analyzing exceptions using a combination of CMF and IMS data enables you to identify problematic CICS-DBCTL transactions, and then analyze whether delays occurred in CICS or IMS.

The following figure shows an overview of the three methods that you can use to analyze CICS-DBCTL transactions: using CMF records, IMS log records, or both.

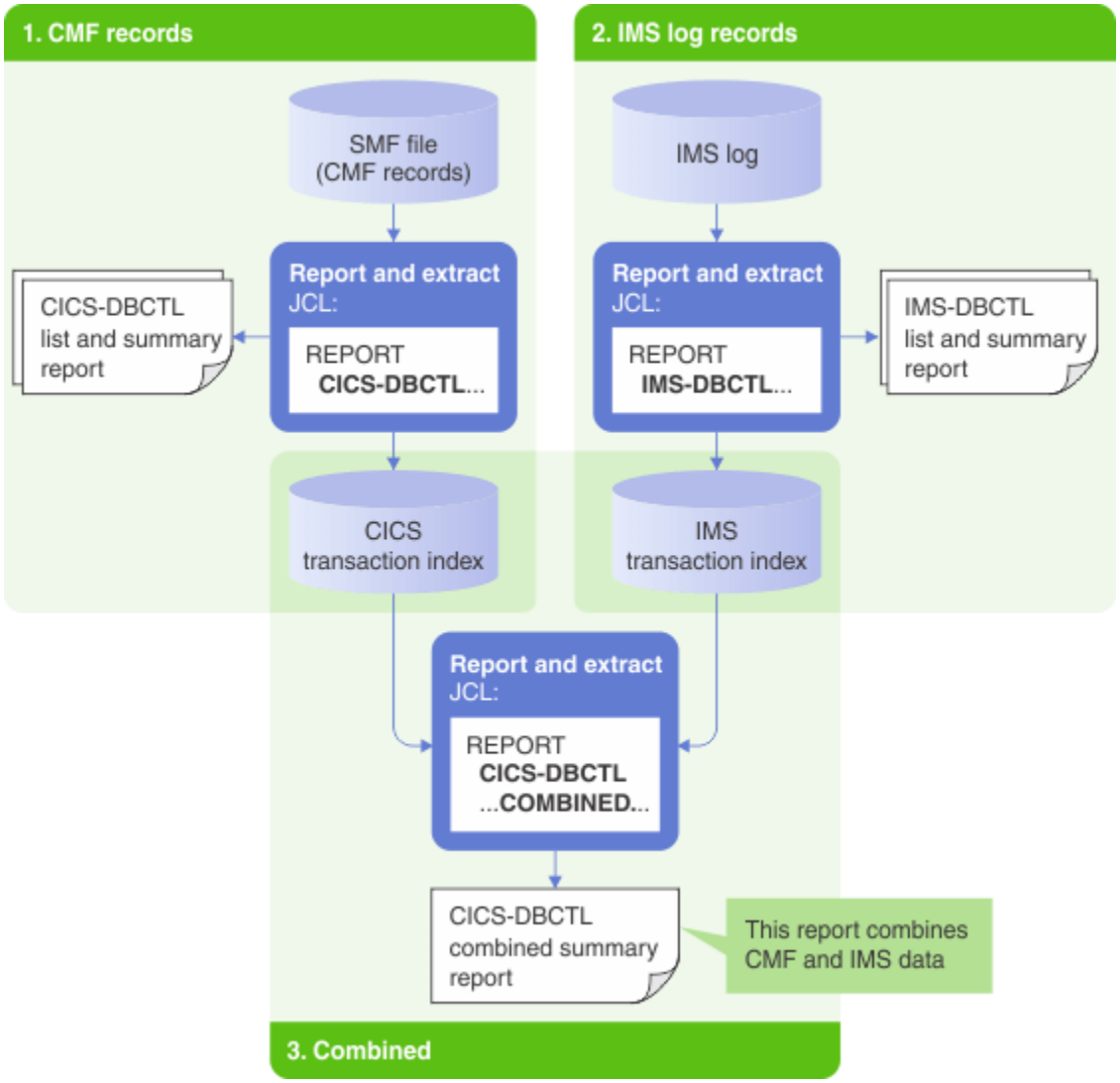


Figure 71. Batch reports for analyzing CICS-DBCTL transaction exceptions using CMF records, IMS log records, or both

You can create the following batch reports to analyze CICS-DBCTL transactions:

1. CICS-DBCTL list and summary reports, using CMF records
2. IMS-DBCTL list and summary reports, using IMS log records
3. CICS-DBCTL combined summary report, combining CMF and IMS data

To create these batch reports and extracts, you submit JCL that calls the **REPORT** command with a **CICS-DBCTL** parameter or an **IMS-DBCTL** parameter. To create the combined summary report, you need a CICS transaction index and an IMS transaction index (sorted extracts) created by earlier `REPORT CICS-DBCTL` and `REPORT IMS-DBCTL` batch commands.

You can also use the Transaction Analysis Workbench log browser to analyze CMF records and IMS log records. If you browse the SMF file (or CICS transaction index) and the corresponding IMS log (or IMS transaction index) together, then you can enter TX next to a CMF record to display the related IMS log records for that transaction.

Related concepts

Exception analysis
Transaction Analysis Workbench exception analysis identifies log records that meet your exception criteria, such as long response time or abnormal conditions, and creates a transaction index of those log records for further analysis.

Related tasks

Analyzing a CICS-DBCTL transaction problem
A user has reported that a CICS transaction failed. As a technical support staff member, you happen to know that this particular transaction is a CICS-DBCTL transaction. We will use Transaction Analysis Workbench CICS-DBCTL reports to help diagnose the problem, and then use the log browser to investigate the problem in detail.

Related reference

CICS-DBCTL reports
CICS-DBCTL reports process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.

REPORT command
Requests a report.

CICS-DBCTL transaction analysis using CMF records

You can use Transaction Analysis Workbench to analyze CICS-DBCTL transaction exceptions using CICS monitoring facility (CMF) performance class records (SMF type 110 records).

The following figure shows an overview of analyzing CICS-DBCTL transactions using CMF records.

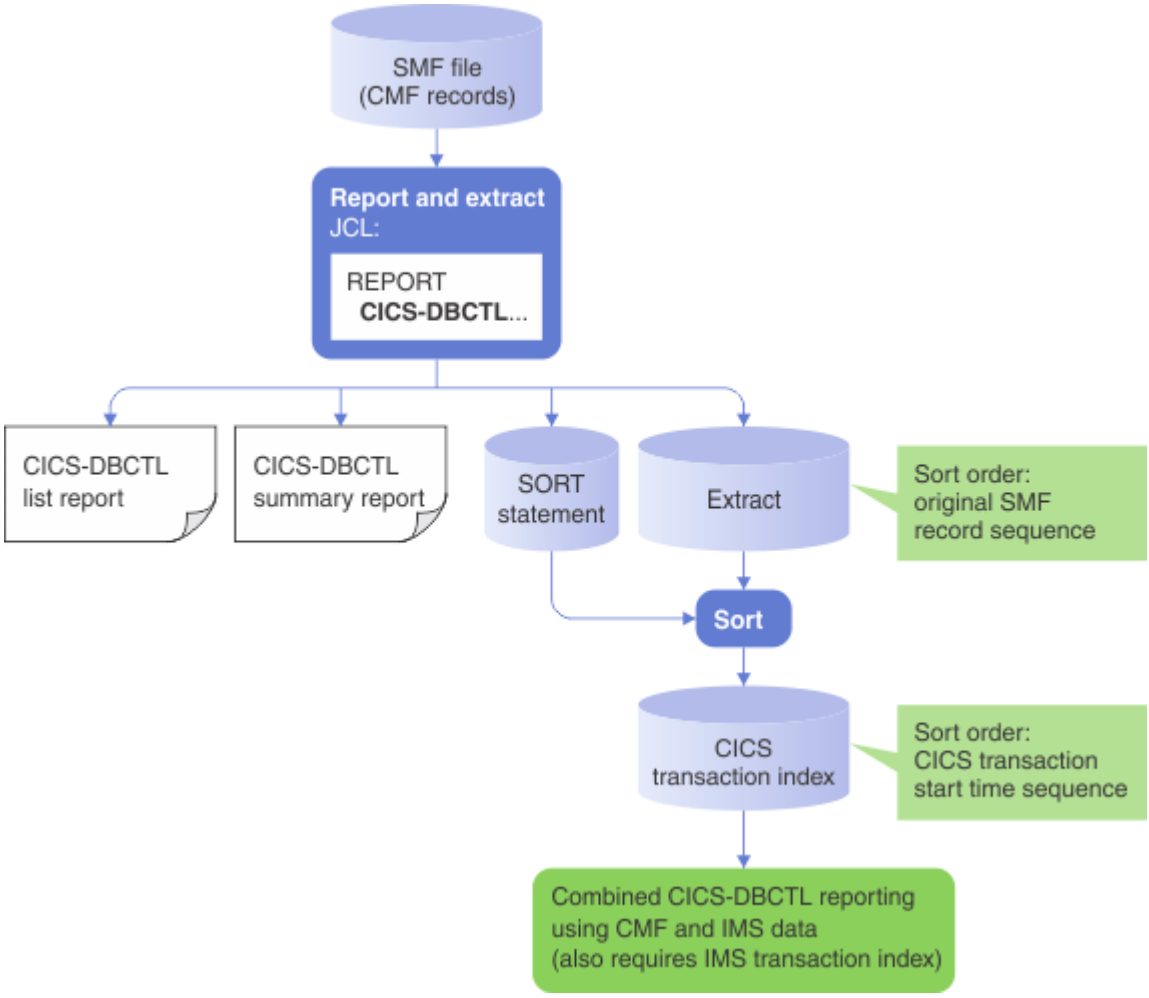


Figure 72. Analyzing CICS-DBCTL transactions using CMF records

If you have an SMF file containing CMF records for CICS-DBCTL transactions, then you can write JCL that uses the `REPORT CICS-DBCTL` command of the Transaction Analysis Workbench report and extract utility to create the following outputs:

CICS-DBCTL list report
Shows the performance characteristics of individual instances of CICS transactions, with a focus on DBCTL.

CICS-DBCTL summary report
Summarizes the data shown in the list report by CICS transaction ID and APPLID.

Extract
The extract is a copy of the input SMF file, filtered to contain only CMF records. If you specified exception criteria, the extract contains only those CMF records that are exceptions.

SORT statement
You can use the **SORT** statement created by the `REPORT CICS-DBCTL` command with the DFSORT program to convert the extract into a *CICS transaction index*, which is a set of CMF records sorted by CICS transaction start time.

You can use a CICS transaction index and corresponding IMS transaction index with a `REPORT CICS-DBCTL ... COMBINED ...` command to create a CICS-DBCTL combined summary report that combines CMF and IMS log data.

You can use the original SMF file, the extract, or the CICS transaction index as input to either CICS Performance Analyzer reporting or the Transaction Analysis Workbench log browser.

CICS-DBCTL transaction analysis using IMS log records

You can use Transaction Analysis Workbench to analyze CICS-DBCTL transaction exceptions using IMS log records. IMS log records contain the details of the IMS events that are triggered by the DL/I requests issued by CICS-DBCTL transactions.

The following figure shows an overview of analyzing CICS-DBCTL transactions using IMS log records.

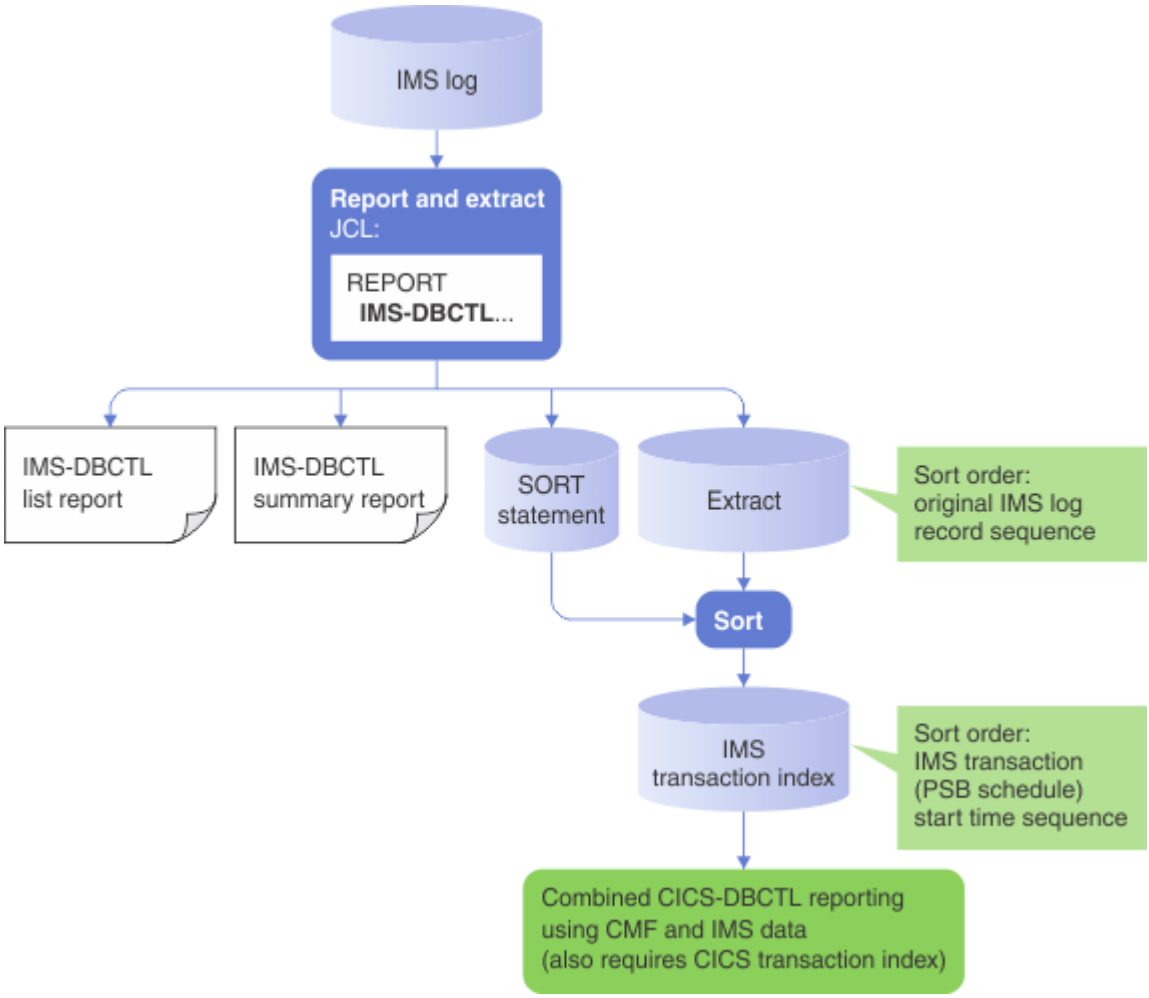


Figure 73. Analyzing CICS-DBCTL transactions using IMS log records

If you have an IMS log containing records for CICS-DBCTL transactions, then you can write JCL that uses the `REPORT IMS-DBCTL` command of the Transaction Analysis Workbench report and extract utility to create the following outputs:

IMS-DBCTL list report
Shows the performance characteristics of individual DBCTL threads. Each DBCTL thread corresponds to one CICS-DBCTL transaction instance. For each DBCTL thread, the report shows the corresponding CICS transaction ID and task number.

IMS-DBCTL summary report
Summarizes the data shown in the list report by CICS transaction ID, IMS PSB name, and CICS generic APPLID.

IMS transaction index
A specialized extract file, created from an IMS log, that consolidates multiple IMS log records for an IMS transaction into one record per transaction. Each record in an index represents an IMS transaction and contains cumulative information from the IMS log about that transaction. You can use an IMS transaction index as input to the following processes:

- IMS Performance Analyzer reporting
- Analyzing CICS-DBCTL exceptions by combining CMF and IMS log data in a `REPORT CICS-DBCTL ... COMBINED ...` command
- Analyzing transactions interactively using the Transaction Analysis Workbench log browser

CICS-DBCTL transaction analysis combining data from CMF and IMS

You can use a CICS transaction index created by an earlier **REPORT CICS-DBCTL** command and an IMS transaction index created by an earlier **REPORT IMS-DBCTL** command to create a CICS-DBCTL combined summary report. This report combines the performance data from CMF records with more detailed information from IMS about DBCTL processing.

The following figure shows an overview of analyzing CICS-DBCTL transactions using a combination of CMF and IMS data.

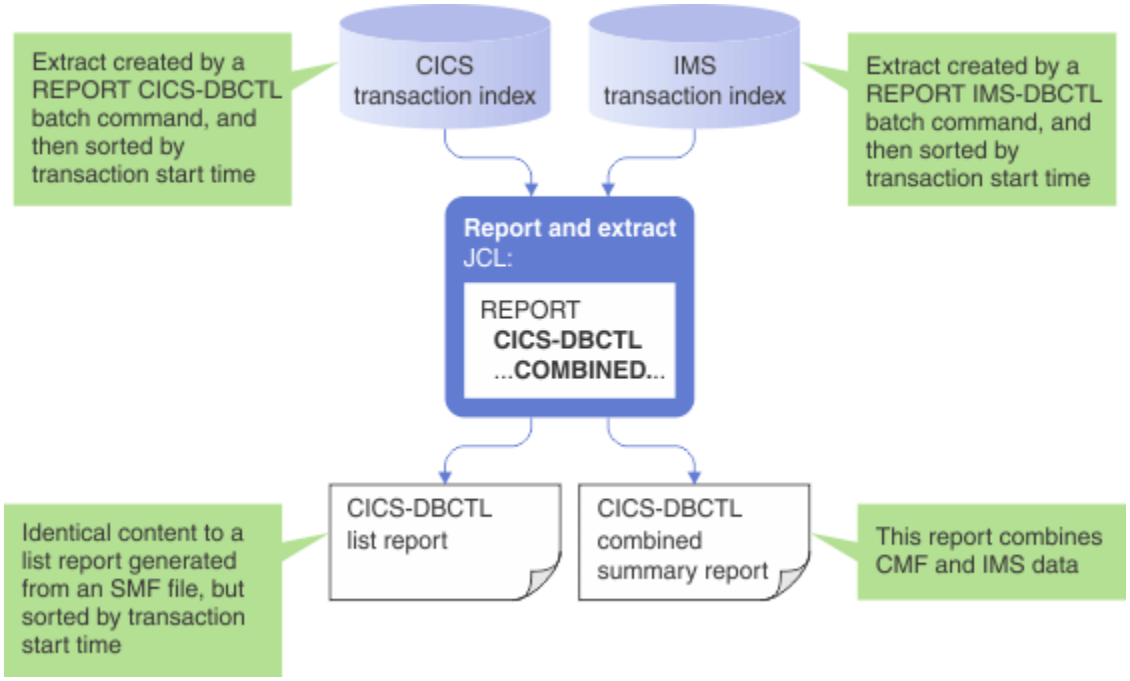


Figure 74. Analyzing CICS-DBCTL transactions using a combination of CMF and IMS data

Chapter 59. DB2 log records

DB2 log records contain detailed recovery information about each change operation performed against DB2 database objects. These changes are recorded as database *undo* (backout) records, *redo* (reapply) records, or a combination of both.

DB2 log records allow DB2 to either commit or back out in-doubt database changes encountered during DB2 system restart or after an application or transaction failure. DB2 log records are initially written to a VSAM linear data set, and then later offloaded, via an automatic archiving process, to a sequential data set called a DB2 archive log file. The format of DB2 log records is mapped by the DB2-supplied macro **DSNDQJ00** in data set DSN.SDSNMACS.

To locate DB2 log files for analysis, use the Transaction Analysis Workbench automated file selection utility.

Related tasks

Analyzing an IMS-DB2 transaction problem
Users have reported long response times from an application. As a technical support staff member, you know that the application runs IMS transactions that perform DB2 updates.

Related reference

DB2 log codes
The DB2 log type consists of log codes for DB2 log records.

DB2 log record formatting

Transaction Analysis Workbench formats DB2 log records with only a subset of the information available in the log record. The information provided is useful for the DB2 systems programmer or database administrator to identify a problem, then to use other tools such as the DB2 Log Analyzer to investigate further.

Transaction Analysis Workbench includes the following information when it formats DB2 log records:

- Database identifier (DBID), pageset identifier (PSID), page number, and the action performed (insert/delete/update). These become useful when investigating a problem for a particular transaction update.
- The complete DB2 table row being inserted or deleted, including the row header clearly identified. However, not all DB2 database changes include the full row image in the log and often only a partial update image is available; DB2 only provides what it needs to recover the update. Full or partial update data is shown in hexadecimal dump notation for clarity.
- Compressed data. The log data may be compressed (if the database administrator has selected that functionality) and Transaction Analysis Workbench will only be able to show what is actually in the log (that is, compressed data).
- Compensation log records (CLR): records written during system recovery to compensate for backed out log changes or committed in-doubt changes.

DB2 log record fields used for tracking

DB2 log records uses three fields as correlation tokens to uniquely identify DB2 transactional updates: RBA, URID, and LUWID. Transaction Analysis Workbench uses these tokens to track transactions, connecting DB2 log records with other log records to provide the complete end-to-end picture for a transaction. For example, you can track the activity of an IMS-DB2 transaction across both environments.

Transaction Analysis Workbench uses each token in the following way:

RBA: relative byte address
Transaction Analysis Workbench reports the DB2 log record RBA as the log sequence number, not the usual record sequence number associated with other record types such as IMS log records. DB2 log records are blocked into 4K physical records. MQ uses the RBA to directly locate records in the

log. RBA values enable you to pinpoint particular areas of interest in the DB2 log, relevant when investigating problems using other complementary DB2 log reporting products.

URID: unit of recovery identifier

DB2 unit of recovery records are cut by transactional DB2 database change (insert, delete, update) operations. A DB2 unit of recovery is uniquely identified by a URID. The URID represents the position (RBA) of the first log record for this unit of work. The sequence of DB2 records for a unit of recovery is bound by a begin unit of recovery record and an end commit phase 2 record. Each DB2 log record associated with the unit of recovery contains the identifying URID.

In Transaction Analysis Workbench, URID is a global field that you can use for filtering across all log record types.

Tracking IMS-DB2 transactions: The IMS recovery token (global field RECTOKEN) is present in the DB2 "unit of recovery control - end commit phase 1" record, effectively tying related DB2 and IMS event records together and allowing you to track an IMS-DB2 transaction.

LUWID: logical unit of work identifier

The DB2 logical unit of work ID (LUWID) is a composite correlation token that also uniquely identifies the transaction (along with the URID).

You can use the LUWID to correlate the analysis of transactions using Transaction Analysis Workbench with more in-depth DB2 log analysis using tools such as the DB2 Log Analyzer. The type 0020 DB2 begin unit of recovery record contains both the URID and LUWID tokens. See the RBA in the LSN column of the ISPF dialog.

Chapter 60. IMS instrumentation

IMS instrumentation is provided by IMS itself and also by other products.

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Related reference

IMS log codes

The IMS log type consists of log codes for IMS log records. Not all log codes are applicable to all releases of IMS.

IMS log records

IMS logs contain many types of record. To analyze IMS problems, you need to know which IMS log record types are useful for analyzing particular areas of IMS processing.

For each area of IMS processing, these topics provide the following information:

1. A list of IMS log records that contain information relating to problems in this area.
2. An example of a relevant log record with a description of the information in the log record that can help your investigation.
3. Sample formatted record report showing:
   - A starting section that is common to all IMS log records including the log code and description, and the STCK time stamp and log sequence number from the suffix.
   - The record is broken down into its various segment components. Each segment starts with a blank line and descriptive heading.

     Where applicable, segments are broken down into their various sub-segments.

     Transaction Analysis Workbench not only maps the basic log record, but when necessary, includes additional macro mappings to provide the complete picture of log record fields.
   - Long fields, such as message text, are dumped in the usual hexadecimal and character format.

   The Transaction Analysis Workbench ISPF dialog displays log records in a similar format but has the added advantage of interactive viewing features such as navigation, tracking, **FIND**, labels and **LOCATE**, tags, adjusting filters, switch to ISPF Browse mode, field zoom, and so on.

   Zoom to any field allows you to:

   a. See a descriptive explanation

   b. Break down flags into their bit settings

   c. Format UTC and STCK time stamps

   For more information on these interactive features, see Part 4, “Browsing logs interactively,” on page 133.

Related reading: When analyzing IMS-related problems, refer to the *IMS Diagnosis Guide* or *IMS Diagnosis Reference* to determine whether the IMS log contains the required information.

Transaction message processing

IMS log records related to transaction message processing.

IMS log code	Description
01/03	IMS message placed on message queue

IMS log code	Description
30	Message prefix was changed
31	A GET UNIQUE was issued for a message
32	A message was rejected; it was presumed to have been the cause of an application program ABEND
33	The queue manager released a record
34	A message was cancelled
35	A message was enqueued or re-enqueued
36	A message was dequeued or saved or deleted
37	Records marked as NO INPUT and NO OUTPUT are written by the syncpoint coordinator when all resource managers have completed Phase 1
3730	Syncpoint End of Phase 1
38	An input message was put back on the input queue when the application abnormally terminated
39	The output queue was freed during cleanup processing of a RELEASE call

Example

The following type 01 log record is for an OTMA message.

The formatted record includes:

- A starting section that is common to all log records including the log code and description, and the STCK time stamp and log sequence number from the suffix.
- The record is broken down into its various segment components. Each segment starts with a blank line and descriptive heading.

  Notice the APPC segment for OTMA (DFSYPRE). The OTMA segment is itself broken down into its various sub-segments.
- Long fields, including the message text, are dumped in the usual hexadecimal and character format.

```
+0004  Code... 01    IMS Message
+0456  STCK... B786D28F6DFA8345      LSN.... 0000000002C6E1E1
       Date... 2011-04-24 Sunday     Time... 11.02.57.853352

+0000  MSGLRLL.... 0466        MSGLRZZ.... 0000        MSGLCODE... 01          MSGFLAGS... D1          MSGDFLG2... 81
+0007  MSGFPADL... 94          MSGMDRRN... 08000009    MSGRDRRN... 08000009    MSGPRFLL... 03DE        MSGCSW..... 00
+0013  MSGDFLG3... 00
+0014  MSGUOW..... Unit of Work (UOW) - Tracking
+0014  MSGORGID... 'IMSP    '  MSGORGTK... B786D28F6DF73A85                    MSGPROID... 'IMSP    '
+002C  MSGPROTK... B786D28F6DF73A85                    MSGUFLG1... 00          MSGUFLG2... 00
+0036  MSGRSQTY... 00          MSGDOFS.... 0000        MSGDRBN.... 00000000

+0040  MSGSSEGM... Message Prefix System Segment; Item ID = 81
+0040  MSGSILL.... 0040        MSGSIID.... 81          MSGCFLG1... 00          MSGCFLG2... C8          MSGCFLG3... 00
+0046  MSGCQSF1... 00          MSGCQSF2... 00
+0048  MSGPTERM... Full Physical Input Terminal ID
+0048  MSGILINE... 00000000    MSGITERM... 00000000
+0050  MSGTISEQ... 1EAC        MSGPREFO... +0          MSGPREFI... +0          MSGSETS.... 00          MSGRES01... 00
+005C  MSGRECCT... +0          MSGIDSTN... FDFFFFFF1CB1CC60                    MSGODSTN... 'STOCK   '
+0070  MSGIHSQN... 0000000000000000                    MSGFMTNM... '        '

+0080  MSGEPHDR... Extended Prefix System Segment; Item ID = 86
+0080  MSGSILL.... 0010        MSGSIID.... 86          MSGEPTL.... 035E        MSGEPFL1... FC          MSGEPFL2... 00

+0090  DFSYPRE.... MVS APPC System Segment for OTMA; Item ID = 87
+0090  LUY_LENGTH......... 0270                       LUY_ZZ..... 0000        LUY_MSG_PREFIX_TYPE........ 87
+0095  LUY_CONVERSATION_TYPE...... ' '                LUY_SYNC_LEVEL..... 'C' LUY_MSG_TYPE....... 80
+0098  LUY_MSG_FLAG....... 84  LUY_RACF_OPT....... 'C'                         LUY_TMAMCRSI....... 20
+009C  LUY_LTERM.......... 'FUN87047'                 LUY_TRANCODE....... 'STOCK   '
+00AC  LUY_TOKEN.......... B786D28F6DE591C5           LUY_FLAGS.......... 00000000
+00B8  LUY_LUMBLK_TOKEN... 1CB1CC60                   LUY_MEMBER_NAME.... 'OTMA0001        '
+00CC  LUY_FRONTEND_SMQNM......... 0000000000000000  LUY_ASYNC_TOKEN.... 0000000000000000

+00E0  TMAMCTL.... IMS XCF Message Control and Header Definitions
+00E0  TMAMCALV... 01          TMAMCMGT... 40          TMAMCRSI... 00          TMAMCCCI... 00          TMAMCTYP... 00
+00E5  TMAMCPFG... 00          TMAMCTNM... 'FUN87047'  TMAMCCHN... A0          TMAMCPFL... F0          TMAMCSSN... 0000097A
+00F4  TMAMCSNS... 00000000    TMAMCSNC... 0000        TMAMCRSC... 0000        TMAMCRSQ... 00000000    TMAMCSEQ... 0001

+0100  TMAMHDR.... State Data Common Section
+0100  TMAMHLEN... 0048

+0102  TMAMHORG... State Data for Transaction messages
+0102  TMAMHIST... 00          TMAMHSYN... 20          TMAMHSLV... 01          TMAMHMAP... '        '
+010E  TMAMHTOK... 00000000000000000000000000000000    TMAMHCOR... 7558C01000000000B786D28F6DB41F85
+012E  TMAMHCID... 00000000000000000000000000000000    TMAMHLTM... '        '  TMAMHLIU... 0000

+0148  TMAMSEC.... Security Data
+0148  TMAMSLEN... 006A        TMAMSFLG... C3          TMAMSFLN... 52

+019E  TMAMSUDS... Security Data for User IDs
+019E  TMAMSULN... 09          TMAMSUTY... 02          TMAMSUID... 'STEVEN  '

+01A8  TMAMSGDS... Security Data for Groups
+01A8  TMAMSGLN... 09          TMAMSGTY... 03          TMAMSGRP... 'CENTRAL '

+014C  TMAMSFDS... Security Data for Utokens
+014C  TMAMSRLN... 51          TMAMSRTY... 00
+014E  TMAMSPRF...
       +0000  50018055 55559555 55555555 55555555 55555555 55555555 55555555 55555555   *&.....n.........................*
       +0020  55555555 55555555 55555555 55555555 55555555 55555555 55555555 55555555   *................................*
       +0040  B7939683 B0928615 0EAC9491 B1A5A5A5                                       *.loc.kf...mj.vvv               *
```

Figure 75. Transaction message processing: IMS log code 01 (part 1 of 2)

```
+01B2  TMAMUSR.... User Data
+01B2  TMAMULEN... 014E
+01B4  TMAMUDTA...
       +0000  C3D64040 00000001 00000040 00000008 FFFFFFFF 00000000 00000311 00000025   *CO  ....... .....................*
       +0020  D4D8C9D4 E2D74040 00000000 00000001 414D5120 4C50514D 20202020 20202020   *MQIMSP  .........(..<&.(........*
       +0040  3CC296F8 05D923D3 00000000 00000000 00000000 00000000 00000000 00000000   *.Bo8.R.L.........................*
       +0060  00000000 C3C1E3C1 4BD5E64B C3E4E2E3 D6D4C5D9 4BD9C5D7 D3E84BD8 E4C5D9E8   *....CATA.NW.CUSTOMER.REPLY.QUERY*
       +0080  40404040 40404040 40404040 40404040 40404040 D4D8D3D7 40404040 40404040   *                    MQLP        *
       +00A0  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
       +00C0  40404040 D5E6C3E4 E2D9D840 40404040 00000000 00000000 00000000 00000000   *    NWCUSRQ     ................*
       +00E0  00000000 00000000 00000000 00000000 40404040 40404040 40404040 40404040   *................                *
       +0100  40404040 40404040 40404040 40404040 00000000 40404040 40404040 40404040   *                ....            *
       +0120  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
       +0140  40404040 D4D8C9D4 E2E5E240                                                 *    MQIMSVS                     *

+0300  MSGSEC..... Security System Segment; Item ID = 88
+0300  MSGSILL.... 0014        MSGSIID.... 88          MSGRACUS... 'STEVEN  '  MSGSAFNM... 'SAFRQ001'  MSGUSIDI... 00

+0314  MSGWLM..... Workload Manager (WLM) System Segment; Item ID = 89
+0314  MSGSILL.... 0010        MSGSIID.... 89          MSGWLMSC... 00178000    MSGMGATM... B786D28F6DFA0145

+0324  MSGMSE..... Message System Extension; Item ID = 8A
+0324  MSGSILL.... 0018        MSGSIID.... 8A
+0328  MSGUTC..... Coordinated Universal Time (UTC)
+0328  MSGUDATE... 2011114F    MSGUTIME... 110257853343                        MSGUZONE... 020D

+033C  MSGMSCE.... MSC Extension System Segment; Item ID = 8B
+033C  MSGMSELL... 005C        MSGSIID.... 8B
+0340  MSGMSCEX... Coordinated Universal Time (UTC)
+0340  MSGMDATE... 00000000    MSGMTIME... 000000000000                        MSGMZONE... 0000
+034C  MSGMSEFL... Flags Multiple-System use
+034C  MSGMSFL5... 00          MSGMSFL6... 00          MSGMSFL7... 00          MSGMSFL8... 80
+0350  MSGSQSID... 0000        MSGESINM... 0000000000000000                    MSGESONM... 0000000000000000
+0364  MSGMETRA... 0000000000000060                    MSGMEVID... 00000000000000000000000000000060    MSGDSSID... 0060
+037E  MSGSRSID... 0060        MSGMORID... 'IMSP    '  MSGMMOTK... B786D28F6DF73A85

+0398  MSGMSC..... TMR System Segment; Item ID = 8C
+0398  MSGSILL.... 0046        MSGSIID.... 8C          MSGMSCZ2... 00          MSGMSTRA... 00000060
+03A0  MSGMSVID... 0000000000000001                    MSGMSONM... 'STOCKBMP'  MSGMSINM... FFFFFFFF1CB1CC60
+03B8  MSGMSOID... 60          MSGMSIID... 60          MSGMSFL1... 01          MSGMSFL2... 4C          MSGMSFL3... 40
+03BD  MSGMSFL4... 08          MSGMSUID... 0000000000000000                    MSGMSGID... 0000
+03C8  MSGMSPAD... 0000000000000000
+03D0  MSGMSCTS... Date and Time - Standard UTC
+03D0  MSGMSCDT... 00000000    MSGMSCTM... 000000000000                        MSGMSCZN... 0000
+03DC  MSGIMSR.... 06          MSGIMSL.... 10

+03DE  QLOGMSGD... Message Text
+03DE  MSGXDLEN... 0078        MSGXFLG1... 03          MSGXFLG2... 00
+03E2  MSGXSTXT...
       +0000  E2E3D6C3 D2C2D4D7 40E38885 408195A2 A6859940 A39640A3 88854083 A4A2A396   *STOCKBMP The answer to the custo*
       +0020  94859940 E2A39683 9240D8A4 8599A840 89A240A3 8881A340 D78199A3 40D5A494   *mer Stock Query is that Part Num*
       +0040  82859940 E7F0F1F2 F0F4F740 89A24081 A5818993 81829385 40819584 40998581   *ber X012047 is available and rea*
       +0060  84A84086 96994084 859389A5 8599A840 40404040                            *dy for delivery                *
```

Figure 76. Transaction message processing: IMS log code 01 (part 2 of 2)

Full-function database processing

IMS log records related to full-function database processing.

IMS log code	Description
20	A database was opened
21	A database was closed
24	The buffer handler detected an I/O error
25	An EEQE was created or deleted
26	An I/O toleration buffer was created
27	A data set was extended
49	RSR Tracking
50	The database was updated. This log record contains the new data on an insert and update call as well as the old data and FSE updates on a delete call
57	Database updates in an RSR environment

Example

The following type 50 log record is for a full-function database update.

The formatted record includes the following useful information:

- User ID and Program
- Database name and RBA
- Undo and Redo data so you can see what has changed

```
+0004  Code... 50    Database Update
+009C  STCK... B96A36B719A19C00      LSN.... 00000000002D5CFD
       Date... 2008-05-13 Tuesday    Time... 22.22.50.259993

+0000  DLENGTH.... 00AC        DLOGZZ..... 0000        DLOGCODE... 50          DLOGSCDE... 50          DLOGPSTN... 0008
+0008  DLOGRTKN... Recovery Token
+0008  DLOIMSID... 'IMSP    '  DLOOASN.... 0000F0DB    DLOCOMN.... 00000000
+0018  DLOGSTCK... B96A36B719A15F00                    DLOGVIMS... 82          DLOGDBF1... 63          DLOGDBF2... 00
+002A  DLOGDBOR... 40          DLOGDSOR... 88          DPGMNAME... 'STOCKP  '  DDBDNAME... 'STOCKDB1'  DDSID..... 01
+003D  DDSID2..... 00          DLOGSLVL... 01          DLOGCALL... 20          DLOGRBA.... 00002000    DLOGBLKO... 0000
+0048  DLOGSEQ.... 000001CC    DLOGXTOF... 0000        DLOGDSOF... 006C        DLOGIDOF... 007A        DLOGTKOF... 0000
+0054  DLOGDLOF... 0000        DLOGKYOF... 0000        DLOGSPOF... 0000        DLOGUNOF... 0082        DLOGREOF... 008C
+0060  DDSTMP..... UTC Time Stamp
+0060  DDATE...... 2008133F    DTIME...... 222250258437                        DZONE...... 008C

+006C  DLOGDSHR... Data Sharing Section
+006C  DLOGDSSN... 00000000    DLOGLSN.... 00000000000018E2                    DLOGUSID... 18E2E298

+007A  DLOGID..... RACF Userid Section
+007A  DLOGUSER... 'STEVEN  '
+0082  DLOGUNDO... Undo Data
Section
+0082  DLOGDFLG... 80          DLOGDFUN... 40          DLOGDOFF... 01CC        DLOGDLEN...
0004
+0088
DLOGDDAT...
       +0000  C7D6D6C4
*GOOD                           *


+008C  DLOGREDO... Redo Data
Section
+008C  DLOGDFLG... 80          DLOGDFUN... 40          DLOGDOFF... 01CC        DLOGDLEN...
0004
+0092
DLOGDDAT...
       +0000  C2C1C45A
*BAD                            *


+0096  DBCKCHN....
0000E6C71670
```

Figure 77. Full-function database update: IMS log code 50

Application program processing

IMS log records related to application program processing.

IMS log code	Description
08	An application program was scheduled
07	An application program terminated
09	SB (Sequential Buffering) statistics
0A08	A CPI communications driven application program was scheduled
0A07	A CPI communications driven application program terminated
11	A conversational program started
12	A conversational program terminated
41	A batch program or BMP program issued a checkpoint
56FA	Transaction statistics - similar data to X'07' record but at transaction, not schedule, level

Example

The following type 07 log record is for a Program Termination.

The formatted record includes the following useful information:

- Transaction Code, Program, Job Name and User ID
- Completion Code
- CPU time
- Accounting data, including DLI call counts

```
+0004  Code... 07    Application Terminate
+01B8  STCK... C62D26AFC2BF7802      LSN.... 00000000000009DB
       Date... 2010-06-24 Thursday   Time... 16.06.38.093303.500

+0000  DLRLL..... 01C8        DLRZZ..... 0000        DLRLTYP.... 07
+0005  DLRNPSB.... 'CEXTPGM '  DLRTRNCD... 'CEXTCONV'  DLRPRTY.... 08
+0016  DLRTYPE.... 01          DLRTIME.... 00000CCC    DLREXTIM... 0.088457
+0028  DLRCMP..... 00          DLRCOMP.... 00000000
+002C  DLRNJOB.... 'IBEHMPP8'  DLRNSTP.... 'REGION  '  DLRMCNT.... +7

+0040  DLRACCT.... Accounting data from DFSDACCT
+0040  DLRGU1..... +0          DLRGN...... +0          DLRGNP..... +0
+004C  DLRGHU..... +14         DLRGHN..... +0          DLRGHNP.... +0
+0058  DLRISRT.... +7          DLRDLET.... +7          DLRREPL.... +7
+0064  DLRCLCNT... +35         DLRGUMES... +8          DLRGNMES... +7
+0070  DLRISMES... +14         DLRPUMES... +0          DLRTSTNQ... +0
+007C  DLRTSTWT... +0          DLRTSTDQ... +0          DLRQCONQ... +0
+0088  DLRQCOWT... +0          DLRQCODQ... +0          DLRSUPNQ... +0
+0094  DLRSUPWT... +0          DLRSUPDQ... +0          DLREXCNQ... +0
+00A0  DLREXCWT... +0          DLREXCDQ... +0          DLRCMD..... +0
+00AC  DLRGCMD.... +0          DLRCHNG.... +0          DLRAUTH.... +0
+00B8  DLRSETO.... +0          DLRAPSB.... +0          DLRDPSB.... +0
+00C4  DLRGMSG.... +0          DLRICMD.... +0          DLRRCMD.... +0
+00D0  DLRCHKP.... +0          DLRXRST.... +0          DLRROLB.... +0
+00DC  DLRROLS.... +0          DLRSETS.... +0          DLRSETU.... +0
+00E8  DLRINIT.... +0          DLRINQY.... +0          DLRSLOG.... +0
+00F4  DLRDEQ..... +0          DLRVSAMR... +0          DLRVSAMW... +7
+0100  DLROSAMR... +0          DLROSAMW... +0          DLRTOTIO... +7
+010C  DLRESAF.... +98         DLRFLD..... +0          DLRPOS..... +0
+0118  DLRRLSE.... +0          DLRXSAVE... +0          DLRXRSTR... +0
+0124  DLRXCOPY... +0          DLRICAL.... +0
+0140  DLRUSID.... 'FUNTRM15'  DLRTSKID... 0000001A    DLRFLAG2... 20
+0155  DLRFLAG3... 80          DLRPSTNR... 0001
+015A  DLRTOKN.... C9C1C4C7404040400000001600000007
+016A  DLRNPGM.... 0000000000000000
+0174  DLRCPUID... 003521D120660000                    DLRABRSN... 00000000
+018C  DLRUSSN.... 00000000    DLRTMEIO... 0.009358    DLRTMEPL... 0
+01A0  DLRIOCNT... +0          DLRSQ6TM... 180.100000
+01A8  DLRACCQ6... 191.200000
+01AC  DLRUTC..... UTC Time Stamp
+01AC  DLRDTE..... 2010175F    DLRTME..... 080638093296
+01B6  DLROFF..... 032C
+0148  DLRNWID.... '        '
```

Figure 78. Application Program termination: IMS log code 07

Fast Path message and database processing

IMS log records related to Fast Path message and database processing.

Table 16. IMS log records related to Fast Path message and database processing

IMS log code	Description
5901	An input message was received
5903	An output message was sent
5910	FP VSO I/O from Dataspace/CF
5911	An input message was inserted on an EMHQ structure
5912	FP VSO CIs hardened to DASD
5916	An output message was inserted on an EMHQ structure

Table 16. IMS log records related to Fast Path message and database processing (continued)

IMS log code	Description
5920	An MSDB was updated
5921	FP DEDB ADS Open
5922	DEDB area data set was closed
5923	DEDB area data set status was changed
5924	An ADS error queue element FLSD (EQE) was created
5936	An output message was dequeued
5937	A synchronization point operation completed
5938	A synchronization point operation was unsuccessful
5942	FP DMHR Dequeue
5945	FP Statistics
5947	Contains a bit map of CIs that have updates in an HSSP image copy data set
5950	A DEDB was updated
5951	Nonrecoverable suppression has taken place
5953	An online utility updated a DEDB
5954	A log record was created each time an area containing sequential dependent buffers was opened
5955	A new buffer for sequential Dependent segments was obtained
5956	Indoubt SDEP buffer from the resynchronization process
5957	Local/Global portion of DMAC logged
5958	An SDEP buffer was successfully written
5970	The MSDB relocation factor for XRF is shown
59FF	Track internal IMS/FP information

Example

The following type 5937 log record is for a Fast Path Transaction Sync Point.

The formatted record includes the following useful information:

- Program
- Completion Indicator
- Accounting data, including DEDB call counts

```
+0004  Code... 5937  Fast Path Sync Point
+00B0  STCK... B7E910B012BB2560      LSN.... 00000000A4F5F18A
       Date... 2011-07-11 Monday     Time... 14.22.43.801010

+0000  SYNCRLL.... 00C0        SYNCRZZ.... 0000        SYNCCODE... 59          SYNCSUBC... 37          SYNCRTYP... 20
+0007  SYNCRSV1... 00          SYNCRGID... 002F        SYNCFAIL... 00          SYNCRSV3... 00          SYNCNAME... 'STOCKP  '
+0014  SYNCDATE... Sync Point Time Stamp (UTC)
+0014  SYNCDATE... 2011192F    SYNCTIME... 142243801008016D
+0020  SYNCPTYP... 40          SYNCPCON... 00
+0022  SYNCSTST... Start of Statistics Area
+0022  SYNCDECL... +34         SYNCDERD... +7          SYNCDEWT... +0          SYNCOVFN... +0          SYNCBFWT... +0
+002C  SYNCBSTL... +0          SYNCNBA.... +10         SYNCOTHR... +2          SYNCNRDB... +1          SYNCUOWC... +0
+0036  SYNCUOWW... +0          SYNCMSCL... +0
+003A  SYNCFLG1... 00          SYNCFLG2... 00          SYNCRTKN... C9D4E2D74040404001C947840000865B    SYNCPBUF... +0
+004E  SYNCPBWT... +0          SYNCASIO... +0          SYNCVSOR... +2          SYNCVSRD... +0          SYNCVSWT... +2
+005C  SYNC#GU.... +14         SYNC#GN.... +10         SYNC#GNP... +0          SYNC#GHU... +5          SYNC#GHN... +0
+0066  SYNC#HNP... +0          SYNC#ISR... +2          SYNC#DEL... +1          SYNC#REP... +2          SYNC#FLD... +0
+0070  SYNC#POS... +0          SYNCCNTN... 0000000000000000                    SYNCLGCI... +0          SYNCCOMB... +0
+007E  SYNCSTCK... B7E910B012BB1460                    SYNCLKS.... +0          SYNCAIOW... +0
+008C  SYNCUOW.... Unit of Work Identifier
+008C  SYNCOIMS... 'IMSP    '  SYNCOTKN... B7E910B012BADD40                    SYNCPIMS... 'IMSP    '
+00A4  SYNCPTKN... B7E910B012BADD40                    SYNCUSF1... 00          SYNCUSF2... 00
```

Figure 79. Fast Path Sync Point: IMS log code 5937

IMS checkpoint processing

IMS log records related to IMS checkpoint processing.

IMS log code	Description
40*xx*	Checkpoint information
45*xx*	Checkpoint statistics
47	PSTs that were active in the system

Example

The following type 4506 log record is a Scheduling Statistics record written at Checkpoint time.

Statistics records provide a useful guide to the performance of your IMS resources including the IMS Message Queue, various Buffer Pools, TCBs, IRLM and many others.

```
+0004  Code... 4506  Scheduling Statistics
+0080  STCK... B96A33D8DF9AC100      LSN.... 00000000002D59C7
       Date... 2008-05-13 Tuesday    Time... 22.10.00.367532

+0000  STATLEN.... 0090        STATZZ..... 0000        STATCODE... 45          STATTYPE... 06          STATID..... 'AS'

+0008  ST4506_STATS....... Scheduling statistics
+0008  ST4506_NOSCHD...... +0                         ST4506_NSDB........ +0
+0010  ST4506_NSCH........ +175                       ST4506_BMPACT...... +1
+0016  ST4506_MPPACT...... +0                         ST4506_NSMISC...... +83
+001C  ST4506_SKCNT....... 0000F0D1
+0030  ST4506_PSBGETSTATS..... PSB Pool GET statistics
+0030  ST4506_PSBGRQSTS... +20                        ST4506_PSBGNSPCCSA..... +0
+0038  ST4506_PSBGCOCSA... +0                         ST4506_PSBGFAILCSA..... +0
+0040  ST4506_PSBGNSPCDLI..... +0                    ST4506_PSBGCODLI... +0
+0048  ST4506_PSBGFAILDLI..... +0                    ST4506_PSBGNSPCBOTH..... +0
+0050  ST4506_PSBGTIMENCO..... 000000000001C5F00    ST4506_PSBGTIMECO..... 0000000000000000
+0060  ST4506_PSBFREESTATS..... PSB Pool Free statistics
+0060  ST4506_PSBFRQSTS... +110                       ST4506_PSBFTIME.... 00000000008EE400
+0070  ST4506_PSBWGETSTATS..... PSB Work Area Pool GET statistics
+0070  ST4506_PSWGRQSTS... +111                       ST4506_PSWGFAIL.... +0
+0078  ST4506_PSWGTIME.... 000000000068EA00
```

Figure 80. Scheduling Statistics at Checkpoint: IMS log code 4506

Security

IMS log records related to security.

IMS log code	Description
10	A security violation occurred

Example

The following type 10 log record is for a security violation.

SCERROR has a value of 66, indicating that an invalid password was entered.

```
+0004  Code... 10    Security Violation
+002C  STCK... B96AB0ADDA8EF600      LSN.... 00000000002E3DB6
       Date... 2008-05-14 Wednesday  Time... 07.28.29.690095

+0000  SCDL..... 003C        SCDZZ..... 0000       SCRFLAG.... 10          SCFLAG1.... 40          SCERROR.... +66
+0008  SCNODE..... 'CENTRAL1'  SCTRAN..... 'STEVEN  '
+0018  SCDST..... Violation Time Stamp (UTC)
+0018  SCDATE..... 2008134F    SCTIME..... 072829690078                      SCOFFSET... 008C
+0024  SCPGM...... 1CCB585000182740
```

Figure 81. Security violation: IMS log code 10

External subsystem processing

IMS log records related to external subsystem processing.

IMS log code	Description
55	Reserved for external subsystem information
56*xx*	IMS external subsystem support recovery

Example

The following type 5612 log record is an ESAF Sync Point event.

Information available includes:

- User ID and Program
- Recovery Token. You can use Tracking to link all events for this recoverable unit-of-work.

```
+0004  Code... 5612  External Subsystem - End of Phase 2 Syncpoint
+0058  STCK... B96A3411C73BFA01      LSN....
00000000002D5C20
       Date... 2008-05-13 Tuesday    Time...
22.11.00.036543


+0000  TPCPLL..... 0068        TPCPZZ..... 0000       TPCPCDE.... 56          TPCPSSTY... 12          TPCPSBCD...
0000
+0008  TPCPOSSN... 'IMSP
'


+0010  TPCP55X.... Data Area for ESAP IMS
use
+0010  TPCPNSS.... +0          TPCPSTN.... 0000       TPCPPSB.... 'STOCK   '  TPCPUSID... 'STEVEN  '  TPCPGRPN...
'CENTRAL1'
+002C  TPCPRTKN... Recovery
Token
+002C  TPIMSID.... 'IMSP'      TPOASN..... 0000F0D2    TPCOMN.....
00000000
+003C  TPCPFLGS... 00          TPCPCPUI... 7202746596720000                  TPCPSIDD... 0000        TPCPSIDS...
0000


+004C  TPCPDATA... Variable
Data
+004C  TPCPCKTZ... 12 byte Time Stamp in STCK
format
+004C  TPCPCLCK... B96A3411C73B9101                  TPCPZONE...
00001AD2
```

Figure 82. ESAF Sync Point: IMS log code 5612

IMS system events

IMS log records related to IMS system events.

IMS log code	Description
06	IMS was started or stopped, plus other events
42	OLDS Switch

Example 1

The following type 06 log record identifies an IMS system event.

ACIDENT has a value of 04, indicating a FEOV on a System Log.

```
+0004  Code... 06    IMS
Accounting
+0044  STCK... B7830830E9AD6943      LSN....
00000000000012EF
       Date... 2002-04-21 Sunday     Time...
10.41.36.352982


+0000  ACLENG..... 0054        ACSPACE.... 0000        ACLFLAG.... 06          ACIMSID.... 'IMSA'      ACFFFFF....
FFFFFF
+000F  ACIDENT.... 04          ACRSENM.... 0000000000000000                    ACPRILOG_51........
0000000000000000
+0024  ACDATE_51.......... 00000000                    ACTTIME_51.........
00000000
+002C  ACPRILOG... PRILOG Start Time
(UTC)
+002C  ACPDATE.... 2002111F    ACPTIME.... 103507300000                        ACPZONE....
016D
+0038  ACDATE..... Current Time
(UTC)
+0038  ACTDATE.... 2002111F    ACTTIME.... 104136352980                        ACTZONE.... 016D
```

Figure 83. IMS system event: IMS log code 06

Example 2

The following type 42 log record is a Log Control record.

It includes information about the active IMS Log and the Restart Checkpoint Table, showing when system checkpoints were last taken.

```
+0004  Code... 42    Log Buffer Control
+0248  STCK... B78306DF269C1B41      LSN....
00000000000000B6
       Date... 2002-04-21 Sunday     Time...
10.35.42.183873


+0000  ATLEN..... 0258        ATZZ....... 0000        ATFLAG..... 42          ATSYST..... 00          ATSYS......
38
+0007  ATBATCH.... 00          ATBUFLEN... 00000600    ATDATE..... 00000224    ATRLMNM.... 00000000    ATIMSNM....
'IMSA    '
+0020  ATRLID..... 0000        ATRLMID.... 00          ATSHFL1.... 40          ATADDR..... 00B81000    ATDATESW...
00000230
+0030  ATLOGCTL... Start of
section
+0030  ATPRITOK... 00000000    ATPRISTR... 0000023C    AT1STLSN... 0000000000000001                    ATGSGNM....
'        '
+004C  ATSGNM..... '        '  ATIMSID.... 'IMSA    '  ATRSENM.... '
'


+0064  BCPBEGIN... Restart Checkpoint
Table
+0064  BCPRVET.... 'RVET'      BCPOLRBN... 0200        BCPOFBYT... 0200        BCPOLBYT... 02B4        BCPBCPT....
'BCPT'
+0078  BCPTADDR... 0C26C000    BCPTRBN.... 00000001    BCPTLRBN... 000000B4    BCPSIDX.... 'SIDX'      BCPLCRE....
'LCRE'
+00A4  BCPRRE..... 'RRE '      BCPRVEND... FFFFFFFF    BCPIMSID... 'IMSA    '  BCPIMSRL... 0154        BCPROLE....
C1
+00E7  BCPCRTYP... 20          BCPSHDST... 01          BCPIRLM.... 00          BCPSKCNT... 00000000    BCPSSIDX...
00010040
+00F4  BCPDSETA... 00000002    BCPASTAB... 00000000    BCP42BLK... 00000000    BCP42OFF...
0000
+010C  BCPTOKEN... B78306DF269BBC01                    BCPRSTID...
00000000000000000000000000
+0120  BCPOLDLG... 00000000000000000000000000          BCPTIME0...
2002111F103541835653016D
+013C  BCPSTCK.... 0000000000000000                    BCPVMUID... 0000000000000000                    BCPRDSSW...
00000000
+0150  BCPRSENM... 'IMSA    '  BCPCURR.... 'CURR'      BCPCCKID... 2002111F103541835653016D            BCPCOLD....
'COLD'
+0178  BCP0CKID... 2002111F103541835653016D            BCPOLCRE... 'LCRE'      BCPLCKID...
2002111F103541835653016D
+0194  BCPOBMP.... 'OBMP'      BCPBMPID... 2002111F103541835653016D            BCPCCRBN... 00000002    BCP0CRBN...
00000002
+01AC  BCPLCRBN... 00000002    BCPBMRBN... 00000002    BCPFPTH.... 'FPTH'      BCPFCKID...
000000000000000000000000
+01C4  BCPCBCC.... 'CBCC'      BCPCMPID... 000000000000000000000000            BCPOLDC....
'OLDC'
+01D8  BCPLDCID... 2002111F103541835653016D            BCPSNPQ.... 'SNPQ'      BCPSNPID...
000000000000000000000000
+01F4  BCPFCRBN... 00000000    BCPCMPBN... 00000000    BCPLDRBN... 00000002    BCPSNRBN... 00000000    BCPDMPQ....
'DMPQ'
+0208  BCPQCKID... 2002111F103541835653016D            BCPQCRBN...
00000002
+0224  ATDATE..... Date/Time of
Log
+0224  ATDATE.....
2002111F103201603517016D
+0230  ATDATESW... Time last Log
Switch
+0230  ATDATESW...
2002111F103507300000016D
+023C  ATPRISTR... PRILOG Start
Time
+023C  ATPRISTR...
2002111F103507300000016D
```

Figure 84. IMS system event: IMS log code 42

Traces

IMS log records related to traces.

Table 17. IMS log records related to traces

IMS log code	Description
67*xx*	Service trace. DFSERA10 and its associated exits can also format these records.

Analyzing individual IMS trace table entries

IMS log records of type 67FA, and 67FF records for IMS trace tables, contain IMS trace table entries. Rather than displaying these 67FA or 67FF IMS trace table records, the log browser formats each IMS trace table entry individually (with the log record type ITR).

To show these IMS trace entries in the log browser, enter the primary command `TRACE ON`.

If the presence of too many IMS trace records becomes distracting and interferes with your analysis of other records in the log browser, you can hide IMS trace records by entering the command `TRACE OFF`.

The **TRACE** setting is specific to each user; its initial value is OFF. If **TRACE** is OFF, and you select for browsing an IMS log file that contains 67FA or 67FF IMS trace records, then Transaction Analysis Workbench displays a pop-up window that enables you to change the setting.

When creating extracts, you can use the **TRACE** batch command to specify whether to process 67FA and 67FF IMS trace table records in their original format or as individual trace entries.

Limitations:

- The time stamp that Transaction Analysis Workbench displays for individual IMS trace table entries is the time stamp of the original 67FA or 67FF IMS trace table record, not the time of the particular trace event. Transaction Analysis Workbench displays the same time stamp for all trace table entries that are from the same 67FA or 67FF IMS trace table record.
- IMS trace table entries contain a 1-byte PST number. This limits accurate reporting of the trace to 255 dependent regions. From version 13, IMS allows 4095 dependent regions; earlier versions allow 999. Unfortunately, due to the 1-byte PST number in IMS trace table entries, if you have more than 255 dependent regions, then Transaction Analysis Workbench will report and track IMS trace table entries incorrectly.

Example

The following listing shows an IMS trace table entry (from an IMS type 67FF record) formatted individually as a record with log code AA.

```
+0018  Code... AA    DLI Database call: DLET
+0038  STCK... C7945AC9BED07D21      LSN.... 00000000000015FA
       Date... 2011-04-06 Wednesday  Time... 08.51.16.132103.820

+0004  Type..... 67          Subtype.... FF
+0006  SNELIDEN... 'DL/ITRAC'  SNCALLID... 'PSAB'      SNCOND..... '0777'
+0018  Entry..... Trace Table entry
       +0000  AA01A729 1BD49748 C4D3C5E3 1BB44104   *..x..Mp.DLET....*
       +0010  01280000 00006919 00044040 2178676C   *.........  ...%*
```

Figure 85. IMS trace table entry (log code AA) from an IMS type 67FF

For an example of using IMS trace table entries to analyze problems, see Chapter 66, “Analyzing a CICS-DBCTL transaction problem,” on page 309.

Related reference

TRACE command
Specifies how to process IMS log records of type 67FA, and 67FF records for IMS trace tables: in their original format, or as individual IMS trace table entries (log type ITR).

ITR: IMS trace table entry codes
The ITR log type consists of log codes that identify IMS trace table entries in IMS log records of type 67FA and 67FF.

Analyzing IMS user log records

IMS applications can use the DL/I **LOG** call to write records to the IMS system log. Such records are known as IMS user log records; they have a log code of X'A0' or greater, which is outside the range of log codes used by IMS itself. To enable Transaction Analysis Workbench to process and format IMS user log records, you create knowledge modules for the corresponding log codes.

To create a knowledge module, you need an assembler mapping macro (DSECT) that describes the structure of the IMS user log record.

A knowledge module is a file that describes the structure and contents of a particular type of log record. Transaction Analysis Workbench uses knowledge modules to process and format log records. Transaction Analysis Workbench is supplied with knowledge modules for many types of log record, including IMS log records written by IMS.

If you want to use Transaction Analysis Workbench to process and format IMS user log records, you must perform the following two steps:

1. Define the user log code to Transaction Analysis Workbench. Transaction Analysis Workbench stores details of user log codes in the control repository.
2. Create a knowledge module for the log code.

After you have performed these steps, you can use Transaction Analysis Workbench to process and format IMS user log records in exactly the same way as other IMS log records.

The following procedure describes these steps in detail.

1. Define the user log code:
   a) On the Transaction Analysis Workbench Primary Option Menu, select option 2 **Controls**.

      The **Controls** menu is displayed, showing the data set name of the control repository where user log codes are stored.

   b) Select option 4 **IMS user log records**.

      The **User Log Codes** panel is displayed, showing a list of existing user log codes in the control repository.

   c) To define a new user log code, enter **NEW** on the command line. To edit an existing user log code, enter S next to the code.

      A user log code must be a 2-digit hexadecimal number in the range A0-FF. Avoid using the same log code as IMS Connect Extensions or OMEGAMON TRF. You can optionally append a subcode in the range 00-FF. All of the following are valid user log codes:

      A1FF
      F0
      B211
2. Create the knowledge module:
   a) In the fields under **User Log Record Data Set**, specify the location of the assembler mapping macro (DSECT) for the user log record.

   b) In the fields under **Knowledge Module Libraries**, specify the libraries where you want to store the knowledge module and its source.

      The load library that you specify here is known as the Transaction Analysis Workbench user load library. You can also set the location of this library via ISPF dialog option 0.1 **Workbench Personal Settings**.

   c) In the **SYSLIB Macro Libraries** fields, specify the locations of the following libraries required to assemble the knowledge module source. If you do not know the locations of these libraries, ask your z/OS system administrator.

      - The Transaction Analysis Workbench samples library, containing the FUWUKMF macro used in knowledge module source.
      - The IMS macro library, SDFSMAC, required if your user log record refers to IMS macros.
      - The MVS macro library, SYS1.MACLIB, required for miscellaneous system macros, such as YREGS.

      Note: When Transaction Analysis Workbench generates the JCL to assemble and link the knowledge module, it includes the **User Log Record Data Set** (the library that contains the mapping macro) in the SYSLIB concatenation.

d) Select option 1 **Create source from Mapping Member**.

Transaction Analysis Workbench creates a knowledge module source member named FUW*nnss*, where *nn* is the user log code and *ss* is the optional subcode. The member is stored in the source library that you specified under the **Knowledge Module Libraries** heading.

e) Select option 2 **Edit source**.

Transaction Analysis Workbench displays the knowledge module source member in the ISPF editor, with message lines indicating lines that require editing. Review the source to decide whether or not you need to edit it before assembling and linking (the next step). For details, see "Reasons to edit knowledge module source" on page 619.

f) Select option 3 **Assemble and Link** to generate JCL to assemble and link the knowledge module, and then enter **SUB** to submit the job.

The job stores the knowledge module in the user load library that you specified under the **Knowledge Module Libraries** heading. The knowledge module has the same name as its source member. Do not rename it. Transaction Analysis Workbench locates knowledge modules by searching the user load library for members with these generated names.

Related concepts

Knowledge modules

A *knowledge module* is an executable load module that Transaction Analysis Workbench uses to interpret and process a particular type of log record.

Related reference

Knowledge module source reference

Knowledge module source members are assembler source. They consist of your original mapping macro, calls to the knowledge module macro FUWUKMF, and exit routines that enable you to programmatically alter field attributes.

Tips for using knowledge modules created by IMS Problem Investigator

Transaction Analysis Workbench can use knowledge modules for IMS user log records that were created by IMS Problem Investigator.

Location and name of the knowledge modules

Knowledge modules for IMS user log records must be stored in the Transaction Analysis Workbench user load library. Transaction Analysis Workbench and IMS Problem Investigator can refer to the same user load library. For details, see Transaction Analysis Workbench dialog option 0.1 **Workbench Personal Settings**.

If Transaction Analysis Workbench and IMS Problem Investigator use different user load libraries, you must copy the IMS Problem Investigator knowledge module members to the Transaction Analysis Workbench user load library.

When searching the user load library for a knowledge module for user log code *nnss* (where *ss* is an optional subcode), Transaction Analysis Workbench looks first for a member named FUW*nnss* (as created by Transaction Analysis Workbench), and then for a member named ALZ*nnss* (as created by IMS Problem Investigator).

Definition of the user log codes

User log codes must be defined in the Transaction Analysis Workbench control repository. Transaction Analysis Workbench can use an existing IMS Problem Investigator control data set (CDS) as a control repository. For details, see Transaction Analysis Workbench dialog option 0.2 **Repositories**.

If Transaction Analysis Workbench does not use the IMS Problem Investigator control data set that contains the user log code definitions, you must define the user log codes in the Transaction Analysis Workbench control repository. Just define the user log code and enter a description; this is enough to

register the user log code, and prompt Transaction Analysis Workbench to look for the corresponding existing knowledge module. You do not need to re-create the knowledge module.

IMS Connect Extensions journal records

You can use Transaction Analysis Workbench to analyze IMS Connect events that have been recorded in an IMS Connect Extensions journal data set. IMS Connect Extensions is an IBM tool that provides instrumentation for IMS Connect. The tool collects real-time data about IMS Connect events, which it then regularly archives.

You can analyze these records in Transaction Analysis Workbench to perform the following tasks:

- Pinpoint TCP/IP performance problems.
- Debug transactions by tracking event flow through IMS Connect and IMS.
- Audit TCP/IP security.

Related reference

CON: IMS Connect event codes
The CON log type consists of log codes for IMS Connect events that have been recorded by IMS Connect Extensions.

IMS IRLM long lock records

The IMS internal resource lock manager (IRLM) writes information about long locks to SMF type 74-15 records, known in Transaction Analysis Workbench as log type SMF, log code 4F0F. These records contain information about the task that is holding, or waiting for, the lock.

The IRLM long lock record contains information about a database lock that occurs during an update.

The global field LongLock is sourced from field R79FDLKC in the record, which is the deadlock cycle number (in hexadecimal format) generated by IRLM and passed to IMS. As IMS requests long lock data for two cycles, this value is used to uniquely identify when the data was gathered.

The global field TranCode is sourced from field R79FTRNM in the record, which is the name of the transaction in a BMP or MPP region, or the job name for all of the remaining region types.

Typically two records are created for each long lock: a Waiter and a Blocker, although there can be more than one of each.

Chapter 61. SMF

The System Management Facilities (SMF) log contains codes identifying MVS SMF records

SMF user records can be an asset to monitor, manage, and analyze the performance and activities of your systems. By including these records in your problem analysis sessions, you can gain deeper insights into the workings of systems and identify opportunities for optimization and improvement. With SMF user records, you can proactively manage systems and ensure your systems are operating efficiently.

Analyzing SMF user records

You can include Systems Management Facilities (SMF) user records in your problem analysis sessions to monitor, manage, and analyze the performance and activities of the systems.

You need an assembler mapping macro (DSECT) that describes the structure of the SMF user record to create a knowledge module.

If you want to use Transaction Analysis Workbench to process and format SMF user records, you must perform the following tasks:

1. Define SMF user records to Transaction Analysis Workbench.

   Transaction Analysis Workbench stores details of SMF user records in the control repository.
2. Create a knowledge module for the SMF user record.

After you complete the tasks, you can use Transaction Analysis Workbench to process and format SMF user records like other IMS log records.

Perform the following steps to define the SMF user record and create the knowledge module:

1. From the **Transaction Analysis Workbench Primary Option Menu**, select option 2 **Controls**.

   The **Controls Menu** is displayed.
2. Select option 5 **SMF user records**.

   The **SMF User Record Types** panel lists existing SMF user records (if any) in the control repository.
3. To edit an existing SMF user record, go to step “8” on page 287. Otherwise, to register a new SMF user record, continue to the next step.
4. Enter `NEW` on the command line.
5. Specify the **Type** of the new SMF user record.

   A **Type** of SMF user record must be within the `128-255` decimal range.
6. Optional: Specify the **Subtype** of the SMF user record.

   The **Subtype** of the SMF user record must be within the `0-255` decimal range. You can leave a blank space if there is no **Subtype** for the SMF user record.
7. Press enter to save the new SMF user record, and then go to step “9” on page 287.
8. Enter S next to the SMF user record to edit an existing record.
9. Choose one of the following options in the **Knowledge Module processing** section:

   1. Create source from Mapping Member
   Transaction Analysis Workbench creates a knowledge module source member named FUW*nnss*.

   Where *nn* is the SMF user record and *ss* is the optional subcode.

   The member is stored in the source library specified under the **Knowledge Module Libraries** section.

   2. Edit source
   Transaction Analysis Workbench displays the knowledge module source member in the ISPF editor, with message lines indicating lines that require editing. Review the source to decide

whether to edit it before assembling and linking. For more information, see “Reasons to edit knowledge module source” on page 619.

3. Assemble and Link
Generate JCL to assemble and link the knowledge module, and then enter **SUB** to submit the job.

The job stores the knowledge module in the user load library that you specified under the **Knowledge Module Libraries** heading. The knowledge module has the same name as its source member. Do not rename it. Transaction Analysis Workbench locates knowledge modules by searching the user load library for members with these generated names.

10. Specify the location of the assembler mapping macro (DSECT) for the SMF user record in the **SMF User Record Data Set** section.
11. Specify the libraries where you want to store the knowledge module and its source in the **Knowledge Module Libraries** section.
12. Specify the locations of the following libraries required to assemble the knowledge module source in the **SYSLIB Macro Libraries** section:
    - **1.** Transaction Analysis Workbench samples library that contains the SFUWSAMP macro used in the knowledge module source.
    - **2.** IMS macro library, SDFSMAC, is required if your SMF user record refers to IMS macros.
    - **3.** The MVS macro library, SYS1.MACLIB, is required for miscellaneous system macros, such as YREGS.

    Notes:

    - If you do not know the locations of these libraries, you can ask your z/OS system administrator for more details.
    - When Transaction Analysis Workbench generates the JCL to assemble and link the knowledge module, it includes the **SMF User Record Data Set** (the library that contains the mapping macro) in the SYSLIB concatenation.

Chapter 62. WebSphere Application Server for z/OS request activity performance statistics

You can use Transaction Analysis Workbench to analyze WebSphere Application Server for z/OS request activity performance statistics (SMF type 120, subtype 9 records).

Example

The following figure shows a list panel of request activity performance statistics records (log type SMF, log code 7809):

```
__ 7809 URI=/FUW/                                                  18.12.56.260000
        Resp=0.006457 InQ=0.000979 Disp=0.004941 OutQ=0.000535 CPU=0.004960
   ------------------------------------------------------------------------------
__ 7809 WAS request-MBean                                          18.13.01.280000
        Resp=0.002281 InQ=0.000375 Disp=0.001513 OutQ=0.000392 CPU=0.002297
   ------------------------------------------------------------------------------
__ 7809 URI=/FileTransfer/transfer/logs%2Fsysout                   18.13.01.670000
        Resp=0.383650 InQ=0.000906 Disp=0.381789 OutQ=0.000954 CPU=0.508912
   ------------------------------------------------------------------------------
__ 7809 WAS request-MBean                                          18.13.01.690000
        Resp=0.001990 InQ=0.000505 Disp=0.001128 OutQ=0.000356 CPU=0.001659
   ------------------------------------------------------------------------------
__ 7809 URI=/FileTransfer/transfer/logs%2Fsyserr                   18.13.02.040000
        Resp=0.337414 InQ=0.000835 Disp=0.334898 OutQ=0.001680 CPU=0.504275
   ------------------------------------------------------------------------------
__ 7809 URI=/FUW/srvRun                                            18.13.04.840000
        Resp=0.040930 InQ=0.000908 Disp=0.039083 OutQ=0.000939 CPU=0.016149
```

Figure 86. Panel: Browsing a list of WebSphere Application Server for z/OS activity request performance statistics (SMF type 78, subtype 9 records)

The following formatted record report shows the details of a single record:

```
+0005  Code... 7809  URI=/FileTransfer/transfer/logs%2Fsysout
+000A  STCK... CA71D70549750000      LSN.... 0000000000000002
       Date... 2012-11-09 Friday     Time... 18.10.01.010000.000

+0000  Length..... 0A78        Flag....... 5E          Type....... +120
+0006  Time....... 0063CB45    Date....... 0112314F    SMF ID..... 'FTS1'
+0012  Subsystem.... 'WAS '    Subtype.... +9

+0000  SM120ST9... WAS Request
+0018  Subtype version number... +2
+0020  Index of this record... +1
+0024  Total number of records.... +1
+0028  Record continuation token.... 0000005000050F03

+00CC  SM120S91... Platform Neutral Server information
+00CC  Version.... +1          Cell short name.... 'H1BASEA '
+00D8  Node short name.... 'H1NODEA '
+00E0  Cluster short name... 'H1SR00  '
+00E8  Server short name.... 'H1SR00A '
+00F0  Server/Controller PID.... 02010514
+00F4  Version of WAS... 08000004

+0118  SM120S92... z/OS Server information
+0118  Version.... +2          System name (CVTSNAME)... 'FTS1    '
+0124  Sysplex name... 'FTS1PLEX'
+012C  Controller Jobname... 'H1SR00A '
+0134  Controller Jobid... 'STC47885'
+013C  Controller SToken.... 00000B5000004D8F
+0144  Controller ASID.... 02D4
+0148  Cluster UUID... C95D356CDE171303000006900000 1FD9AC114519
+015C  Server UUID.... C95D356CD38CAB220000069000001FD9AC114519
+0170  Daemon Group name.... 'H1BASEA '
+0178  LE GMT Offset (hours).... +0
```

```
+017C  LE GMT Offset (min).... +0
+0180  LE GMT Offset (sec).... +0
+0188  System GMT Offset.... 00006B49D2000000
+0190  Service level.... 'cf041228.01'

+01B4  SM120S93... Platform Neutral Request information
+01B4  Version.... +1          Dispatch Servant PID... 00010444
+01BC  Dispatch Task ID... 217AF20000000033
+01C4  Dispatch CPU time.... 0.000126
+01CC  Completion minor code.... 00000000
+01D4  Request type... 00000003

+01F8  SM120S94... z/OS Request information
+01F8  Version.... +2          Arrival.... 18.10.00.601487
+020C  Queued..... 18.10.00.603250
+021C  Dispatched... 18.10.00.603760
+022C  Completed.... 18.10.01.010729
+023C  Finished... 18.10.01.011503                    Response... 0.410015
       InputQ..... 0.002272    Dispatch... 0.406968    OutputQ.... 0.000773
+024C  Dispatch Servant Jobname... 'H1SR00AS'
+0254  Dispatch Servant Jobid... 'STC47908'
+025C  Dispatch Servant SToken.... 00000D2000000896
+0264  Dispatch Servant ASID.... 0348
+0268  Dispatch TCB address... 008CB168
+026C  Dispatch TCB TToken.... 00000D200000089600000036008CB168
+027C  Dispatch CPU offloaded to specialty CP... 0.000119
+0284  Enclave Token.... 0000005000050F03
+02AC  Enclave CPU.... 0.517668                       Enclave zAAP CPU... 0
+02BC  Enclave zAAP Eligible on CP.... 0
+02C4  Enclave zIIP on CPU.... 0.028129
+02CC  Enclave zIIP Qual Time... 0
+02D4  Enclave zIIP CPU... 0.487597
+02DC  zAAP normalization factor.... +418
+02E0  Enclave CPU time... 0.518143
+02E8  Enclave CPU time on zAAP... 0
+02F0  zAAP normalization factor.... +418
+02F8  Enclave CPU time on zIIP... 0.488072
+0300  Enclave CPU SUs on zIIP.... +10878
+0308  Enclave CPU SUs on zAAP.... +0
+0310  Enclave CPU SUs.... +11549
+0318  Response Time Ratio.... +0                     GTID....... 00000000
+0374  Dispatch timeout value... +300
+0378  Transaction class.... '        '              Flag....... 80F0
+03A4  stalled_thread_dump_action... 00000003
+03A8  cputimeused_dump_action.... 00000003
+03AC  dpm_dump_action.... 00000003
+03B0  timeout_recovery... 00000001
+03B4  dispatch_timeout classification.... +300
+03B8  Queue timeout.... +297
+03BC  request_timeout classification... +180
+03C0  cputimeused_limit classification... +0
+03C4  dpm_interval classification.... +0
+03C8  message_tag classification... +0
+03D0  Obtained affinity RNAME.... 00
+0454  Routing affinity RNAME... 00

+04D8  SM120S95... z/OS Formatted Timestamps
+04D8  Arrival.... '2012/11/09 10:10:00.601487'
+04F2  Queued..... '2012/11/09 10:10:00.603250'
+050C  Dispatched... '2012/11/09 10:10:00.603760'
+0526  Completed.... '2012/11/09 10:10:01.010729'
+0540  Finished... '2012/11/09 10:10:01.011503'

+055C  SM120S96... Network data - HTTP/SIP and IIOP transport
+055C  Version.... +1          Bytes received... +1398
+0568  Bytes sent... +0        Target port.... +18046
+0578  Origin string.... 'ip addr=172.17.69.74 port=63869'

+0618  SM120S97... Classification data
+0618  Version.... +1          Type....... 00000006
+0624  Data....... '/FileTransfer/transfer/logs%2Fsysout'

+06A4  SM120S97... Classification data
+06A4  Version.... +1          Type....... 00000007
+06B0  Data....... 'PROD10.FUNDI.PRIV'

+0730  SM120S97... Classification data
+0730  Version.... +1          Type....... 00000008    Data....... '18046'

+07BC  SM120S98... Security data
+07BC  Version.... +1          Type....... 00000001
+07C8  Identity String.... 'server:h1basea_h1nodea_h1sr00a'
```

```
+0808  SM120S98... Security data
+0808  Version.... +1          Type....... 00000003
+0814  Identity String.... 'TWM'

+0854  SM120S99... CPU Usage breakdown
+0854  Version.... +1          Type....... 00000002    CPU time... 0.000125
+0864  Elapsed time... 0.000000
+086C  Invocation count... +1
+0874  String 1... 'filetransfer#filetransfer.war'
+0978  String 2... 'transfer'
```

Chapter 63. IBM MQ log extracts

You can use Transaction Analysis Workbench to analyze log extracts from IBM MQ.

Transaction Analysis Workbench does not directly support the native IBM MQ log. You must use the IBM MQ log print utility CSQ1LOGP to create a log extract. Transaction Analysis Workbench formats IBM MQ log extract records according to the mapping supplied with MQ in the C language header file CSQ4LOGD in the library SCSQC370.

Fields used as correlation tokens for tracking

IBM MQ log extract records use the following three fields as correlation tokens to uniquely identify IBM MQ transaction events. Transaction Analysis Workbench uses these tokens to track transactions, connecting IBM MQ log extract records with other log records to provide the complete end-to-end picture for a transaction.

RBA: relative byte address
Transaction Analysis Workbench reports the IBM MQ log extract record RBA as the log sequence number, not the usual record sequence number associated with other record types such as IMS log records. IBM MQ uses the RBA to directly locate records in the log. RBA values enable you to pinpoint particular areas of interest in the DB2 log, relevant when investigating problems using other complementary DB2 log reporting products.

URID: unit of recovery identifier
IBM MQ application requests cut unit of recovery records, typically send/receive (MQPUT/MQGET) and object change (MQSET) calls. Each IBM MQ unit of recovery is uniquely identified by its unit of recovery identifier (URID). The URID is the RBA of the first record cut for the unit of recovery. Every IBM MQ log record in the unit of recovery has the same URID. You can use IMS PI extended tracking (TX line action) to display all of the records that are associated with an MQ unit of recovery.

IMS adapter environments: Every persistent IBM MQ message has its own URID because it is committed when the message is issued. Therefore, the URID cannot be used to track this request with other messages that are associated with the transaction. In this case, Transaction Analysis Workbench uses the dependent region partition specification table (PST) ID to identify a IBM MQ request with its IMS transaction.

LogToken: message or correlation identifier
For the LogToken global field value, Transaction Analysis Workbench uses the STCK value from the trailing bytes of one of the following IBM MQ fields:

- If available, the correlation ID (MQMD_CORRELID). This allows the outgoing message to be tied back to the original incoming request.
- Otherwise, the MQ message ID (MQMD_MSGID).

Tracking with combined IMS and MQ logs

The result of tracking can change depending on several factors. You must note the following information when tracking transactions within combined IMS and MQ logs:

- Tracking requests initiated against an MQ log record uses the URID value to correlate the IMS log to the MQ log. The result includes all of the MQ log records associated with an MQ unit of recovery.
- Tracking request initiated against an IMS log record uses the LogToken value to correlate the IMS log to the MQ log. Only MQ PUT and MQ GET log records that are directly related to the tracked transaction's input and output messages are displayed.

IBM MQ log time stamps

Some IBM MQ time stamps are estimates only, as explained in the *IBM MQ z/OS System Administration Guide*:

> A time stamp can only be extracted from a Begin-UR record or from an MQPUT request. If there is only a long running transaction which is getting messages, the times when the gets occurred are all at the time the transaction started (the Begin-UR record). If there are many short units of work, or many messages being put, the time is reasonably accurate (within milliseconds). Otherwise the times become less and less accurate.

Related reference

MQ: IBM MQ log codes

The MQ log type consists of log codes that identify IBM MQ log extract records.

Locating IBM MQ log files

To locate IBM MQ log files, run the IBM MQ-supplied print log map utility, CSQJU004.

Run the MQ print log map utility to generate a list of MQ log files and their time spans, using information in the MQ bootstrap data set (BSDS).

For example:

```
//CSQJU004 JOB,NOTIFY=&SYSUID
//*
//STEP1    EXEC PGM=CSQJU004
//STEPLIB  DD  DISP=SHR,DSN=CSQ900.SCSQANLE
//         DD  DISP=SHR,DSN=CSQ900.SCSQAUTH
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=CSQ900.CSQ6.BSDS01
```

The utility produces the following report showing the data set names of the archive log data sets and the time ranges that they span.

```
ARCHIVE LOG COPY 1 DATA SETS

START RBA/TIME/LRSN      END  RBA/TIME/LRSN      CREATED     DATA SET INFORMATION
----------------------  ----------------------  ----------  --------------------
 0000001D3000 /          00000060AFFF /         2007-12-14   DSN=CSQARC1.CSQ6.A0000005
2007-12-05  14:32:42.4  2007-12-14  15:36:30.2        15:36  PASSWORD=********
      / C1994770A288          / C1A4A67C3330                 VOL=DEV029 UNIT=DASD
                                                               CATALOGUED
```

Determine which IBM MQ log file you want to analyze, and then run the IBM MQ log print utility, CSQ1LOGP, to extract that log into a log extract file that you can analyze with Transaction Analysis Workbench.

Extracting IBM MQ log files

To extract records from an IBM MQ log file into a log extract file that you can analyze with Transaction Analysis Workbench, run the IBM MQ-supplied print log utility, CSQ1LOGP.

Run the IBM MQ-supplied print log map utility, CSQJU004, to locate the log file that contains the records you want to extract.

Run the MQ print log map utility with the EXTRACT(YES) option specified. The ARCHIVE DD statement identifies the native MQ log, while the CSQBOTH DD statement identifies the extract data set.

For example:

```
//CSQ1LOGP JOB,NOTIFY=&SYSUID
//*
//STEP1    EXEC PGM=CSQ1LOGP
//STEPLIB  DD  DISP=SHR,DSN=CSQ900.SCSQANLE
//         DD  DISP=SHR,DSN=CSQ900.SCSQLOAD
//SYSPRINT DD  SYSOUT=*
//ARCHIVE  DD  DISP=SHR,DSN=CSQARC1.CSQ6.A0000005
//CSQBOTH  DD  DISP=(NEW,CATLG),DSN=MY.MQ.EXTRACT,
//         SPACE=(CYL,(10,5)),UNIT=SYSDA
//SYSIN DD *
EXTRACT(YES)
/*
```

Figure 87. JCL to extract IBM MQ log records

Example: IBM MQ requests submitted by an IMS transaction via the IBM MQ-IMS adapter

This example shows an IMS transaction that submits requests to IBM MQ via the IBM MQ-IMS adapter.

This example shows a single IMS transaction that performs the following activity:

1
Full-function IMS database updates.

2
DB2 SQL calls.

3
IMS adapter GET and PUT requests to IBM MQ.

```
   . . .
 01   Input Message TranCode=MQATREQ1                        09.49.26.679852
 35   Input Message Enqueue TranCode=MQATREQ1                      +0.000023
 31   DLI GU TranCode=MQATREQ1 Region=0001                         +0.000137
 5E   SB Handler requests Image Capture Region=0001               +0.000262
 50   Database Update Database=DI21PART Region=0001               +0.000643  1
 50   Database Update Database=DI21PART Region=0001               +0.000720
 50   Database Update Database=DI21PART Region=0001               +0.000771
 5600 Sign-on to ESAF Region=0001 SSID=DB3A                       +0.001604
 0020 DB2 Unit of Recovery Control - Begin UR                     +0.023043  2
 0020 DB2 Update In-Place in a Data Page                          +0.023059
 0010 DB2 Savepoint                                               +0.023347
 0020 DB2 Delete from a Data Page                                 +0.023459
 0020 DB2 Insert into a Data Page                                 +0.023683
 5600 Sign-on to ESAF Region=0001 SSID=CSQ6                       +0.145085
 0002 MQ Get Region=0001                                          +0.145870  3
 0006 MQ Commit Phase 1 Region=0001                               +0.145870
 0007 MQ Commit Phase 2 Region=0001                               +0.145870
 0002 MQ Get Region=0001                                          +0.148405
 0007 MQ Commit Phase 2 Region=0001                               +0.154640
 0002 MQ Get Region=0001                                          +0.156635

    . . . [Multiple additional MQ calls]

-------------------------------------------------------------------------------
 07   Application Terminate                                       +1.073791
      UTC=10.37.00.753639 TranCode=MQATREQ1 Program=MQATPGM Region=0001
      RecToken=IADG/0000000700000005
      RegTyp=MPP MCNT=5 DBDLI=10 DCDLI=10 CPU=0.129896 ESAF=220  4
-------------------------------------------------------------------------------
```

Figure 88. IMS adapter: IMS transaction issues IBM MQ requests

4
Observe the high number of MQ calls (in excess of 200). This might be the cause of a problem.

The expanded record details of the IMS 07 record are shown in this screen capture for illustration purposes only: in practice, the log browser does not display a mix of expanded and condensed record details. To expand or condense record details in the log browser, scroll left or right: press the Left function key (F10) or the Right function key (F11).

Example: IMS transaction submitted via the IBM MQ-IMS bridge

In this example, a IBM MQ message to the IBM MQ-IMS bridge submits an IMS transaction (TranCode=PART).

The sequence of events that is shown in the following screen image is as follows:

1
IBM MQ puts the message on the IMS bridge queue.

2
IBM MQ commits the message.

3
IMS initiates the PART transaction.

The Source field of the IMS input message indicates that the transaction was initiated by IBM MQ; the QMgr field identifies the IBM MQ queue manager; and the RepQ field identifies the IBM MQ reply-to queue (where the IMS bridge will put the reply from the IMS transaction).

4
IMS processes the transaction.

5
IMS notifies OTMA that the transaction reply is available.

6
OTMA gets the transaction reply from the IMS message queue, and then sends it to the IMS bridge; the IMS bridge puts it on the specified IBM MQ reply-to queue.

```
Code Description                Date 2008-02-20 Wednesday  Time (Local)
---- ---------------------------------------------------- ----------------
0001 MQ Put                                               16.31.12.275748 1
     Userid=TWM LogToken=C1FA31888C8C6881 SSID=CSQ6 QMgr=CSQ6
     Delta=52 ConnType=BATCH RepQ=MQIADG_REPLY_Q QName=MQIADG_REQ_Q
     URID=000000AAAB29
-------------------------------------------------------------------------------
0006 MQ Commit Phase 1                                    16.31.12.275748 2
     SSID=CSQ6 Delta=52 ConnType=BATCH URID=000000AAAB29
-------------------------------------------------------------------------------
0007 MQ Commit Phase 2                                    16.31.12.275748
     SSID=CSQ6 Delta=52 ConnType=BATCH URID=000000AAAB29
-------------------------------------------------------------------------------
01   Input Message                                        16.31.12.477973 3
     UTC=17.31.12.477964 TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027
     OrgUOWID=IADG/C1FA3188BDEE0023 LogToken=C1FA31888C8C6881 SSN=017
     QMgr=CSQ6 Source=MQ RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
```

Figure 89. MQ-IMS bridge: MQ message initiates an IMS transaction (part 1 of 2)

```
Code Description                Date 2008-02-20 Wednesday  Time (Local)
---- ---------------------------------------------------- ----------------
35   Input Message Enqueue                                  16.31.12.477997
     UTC=17.31.12.477964 TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027
     OrgUOWID=IADG/C1FA3188BDEE0023 LogToken=C1FA31888C8C6881 SSN=017
     QMgr=CSQ6 RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
5607 Start of UOR                                           16.31.12.552256 4
     Program=DFSSAM02 Region=0001 IMSID=IADG RecToken=IADG/0000000C00000000
-------------------------------------------------------------------------------
31   DLI GU                                                 16.31.12.552283
     UTC=17.31.12.552280 TranCode=PART Region=0001
     OrgUOWID=IADG/C1FA3188BDEE0023 RecToken=IADG/0000000C00000000
-------------------------------------------------------------------------------
5E   SB Handler requests Image Capture                     16.31.12.567547
     UTC=17.31.12.552133 Region=0001
-------------------------------------------------------------------------------
5610 Start Phase 1 Syncpoint                                16.31.12.569615
     Region=0001 IMSID=IADG RecToken=IADG/0000000C00000000
-------------------------------------------------------------------------------
03   Output Message Response                                16.31.12.569666
     UTC=17.31.12.477964 TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027
     OrgUOWID=IADG/C1FA3188BDEE0023 LogToken=C1FA31888C8C6881 SSN=017
     QMgr=CSQ6 Source=MQ RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
35   Output Message Enqueue                                 16.31.12.569689
     UTC=17.31.12.569680 TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027
     Region=0001 OrgUOWID=IADG/C1FA3188BDEE0023
     RecToken=IADG/0000000C00000000 LogToken=C1FA31888C8C6881 SSN=017
     QMgr=CSQ6 RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
37   Syncpoint                                              16.31.12.569715
     Region=0001 RecToken=IADG/0000000C00000000
-------------------------------------------------------------------------------
37   Syncpoint message transfer                             16.31.12.569750 5
     TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027 Region=0001
     OrgUOWID=IADG/C1FA3188BDEE0023 RecToken=IADG/0000000C00000000
     LogToken=C1FA31888C8C6881 SSN=017 QMgr=CSQ6 RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
33   Free Message                                           16.31.12.569780
     OrgUOWID=IADG/C1FA3188BDEE0023
-------------------------------------------------------------------------------
5612 End of Phase 2 Syncpoint                               16.31.12.569824
     Program=DFSSAM02 Region=0001 IMSID=IADG RecToken=IADG/0000000C00000000
-------------------------------------------------------------------------------
31   Communications GU                                      16.31.12.588964 6
     UTC=17.31.12.588956 TranCode=PART LTerm=CSQ00027 Terminal=CSQ00027
     OrgUOWID=IADG/C1FA3188BDEE0023 LogToken=C1FA31888C8C6881 SSN=017
     QMgr=CSQ6 RepQ=MQIADG_REPLY_Q
-------------------------------------------------------------------------------
0001 MQ Put                                                 16.31.12.605969
     Userid=TWM LogToken=C1FA31888C8C6881 SSID=CSQ6 QMgr=CSQ6
     QName=MQIADG_REPLY_Q URID=000000AADC23
-------------------------------------------------------------------------------
```

Figure 90. MQ-IMS bridge: MQ message initiates an IMS transaction (part 2 of 2)

Chapter 64. z/OS SMF and OPERLOG records

You can use Transaction Analysis Workbench to analyze OPERLOG records and various SMF records.

SMF data

Transaction Analysis Workbench can process SMF log streams and dumped SMF MAN*x* data sets.

No special preparation is required unless your SMF files have been post-processed so that their records are no longer in time sequence.

For example, a site might combine SMF files from individual systems, for the same time period, into a single large file, by concatenating the files rather than merging their records in time sequence. In this case, although the set of records for each system are in time sequence, the resulting file as a whole is not in time sequence. To perform batch processing on records for a particular time period from a SMF file containing records that are not in time sequence, you must specify the **UNSORTED** command. The **UNSORTED** command causes Transaction Analysis Workbench to read and check the time stamp of every record in the file.

Before processing an SMF file using the Transaction Analysis Workbench ISPF dialog, you must ensure that the records are in time sequence. To check whether the records in an SMF file are in time sequence, create an SMF Recap report. If the records in an SMF file are not in time sequence, then, rather than sorting all of the records in that (potentially very large) SMF file, it is recommended that you extract the records that you are interested in, and then sort the records in that (smaller) extract.

Related reference

MVS: OPERLOG and OTHER log stream codes
The MVS log type consists of log codes that identify records from OPERLOG and records from log streams specified using the prefix OTHER:.

SMF codes
The SMF log type consists of log codes that identify MVS System Management Facilities records.

SMF reports
Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

UNSORTED command
Indicates that records in the input log files are unsorted; their records are not in time sequence. Specifying **UNSORTED** causes Transaction Analysis Workbench to test each input log record and process it if falls within the period specified by the **START** and **STOP** commands.

Part 11. Scenarios

These step-by-step scenarios demonstrate using Transaction Analysis Workbench to analyze specific types of transaction.

Related concepts

Tutorials

Follow these step-by-step tutorials to help you get started with Transaction Analysis Workbench.

Chapter 65. Analyzing a CICS-DB2 transaction problem

Users have reported long response times from an application. As a technical support staff member, you know that the application runs CICS transactions that access DB2. We will use Transaction Analysis Workbench to identify the cause of the problem as a SQL SELECT statement run by a stored procedure. We will see that the poor performance is due to the SQL SELECT statement performing a lengthy table (or "sequential data") scan rather than a (typically faster) index scan.

The analysis procedure described here assumes that you have already performed the following steps in Transaction Analysis Workbench:

1. Created a session for the problem.
2. Run workflow tasks, inherited from the session template on which you based the session, that perform the following steps:
   a. Select the log files required to analyze the problem, containing log records written by the related CICS and DB2 systems for the period when the problem occurred.
   b. Create a CICS transaction index: an extract of an SMF file containing selected SMF 110 (CICS monitoring facility; CMF) records, sorted by CICS transaction start time.

To analyze a CICS-DB2 transaction problem, you typically need the following log files:

- Archived DB2 log
- Dumped SMF files (or SMF log stream) containing at least the following types of record:
  - DB2 accounting (SMF type 101) records (IFCIDs 003 and 239)
  - DB2 performance (SMF type 102) records (IFCIDs belonging to trace level 2, as defined by Transaction Analysis Workbench)
  - CICS monitoring facility (CMF) performance class (SMF type 110, subtype 1, class 3)

  Note: For DB2 accounting and performance records, instead of using SMF files, you can use GTF data sets.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**, and then select the session that you have created for this problem.
2. Select option 5 **Investigate**.

   The **Investigate** panel is displayed.

```
  File  Menu  Time Slicing  Help

                              Investigate              Row 1 of 4 More: < >
Command ===> ____________________________________________ Scroll ===> PAGE

                           Time Slice (ON)
               Time            Date           Duration
               HH.MM.SS.thmiju YYYY-MM-DD     HH.MM.SS  Zone   Filter +
               15.25.00.000000 2013-10-08     00.05.00  LOCAL  ________
/
________ Type  Data Set Name                                    Coverage
T_______ CMF   FUW000.QADATA.FBOSP006.CICS.D131008.INDEX        COMPLETE
________ DTR   FUW000.QADATA.FBOSP006.SMF.D131008.DB2PERF2     PARTIAL
________ DTR   FUW000.QADATA.FBOSP006.SMF.D131008.DB2NOPE2     PARTIAL
________ DB2   FUW000.QADATA.FBOSP006.DB2.D131008.ARCLOG       COMPLETE
****************************** Bottom of data ********************************
```

   *Figure 91. Panel: **Investigate**: selecting an entire log file (rather than a time slice)*

3. Enter line action T next to the CICS transaction index.

Line action T sets the time and duration of the time slice (the period of time that we want to investigate) to match the selected log file.

4. Enter line action S on the first line under the **/** heading, to browse a merged view of all the log files in the list.

   (This is equivalent to typing S next to each log file, and then pressing Enter.)

   The log browser is displayed.

   We want to filter the log browser to show only CMF records, so that we can then select a single instance of a CICS transaction for closer inspection.

5. Press the Filter function key (F18) or enter **FILTER** on the command line.

   The **Filter** panel is displayed.

6. Specify CMF under the **Log** column heading, and 6E13 under the **Code** column heading.

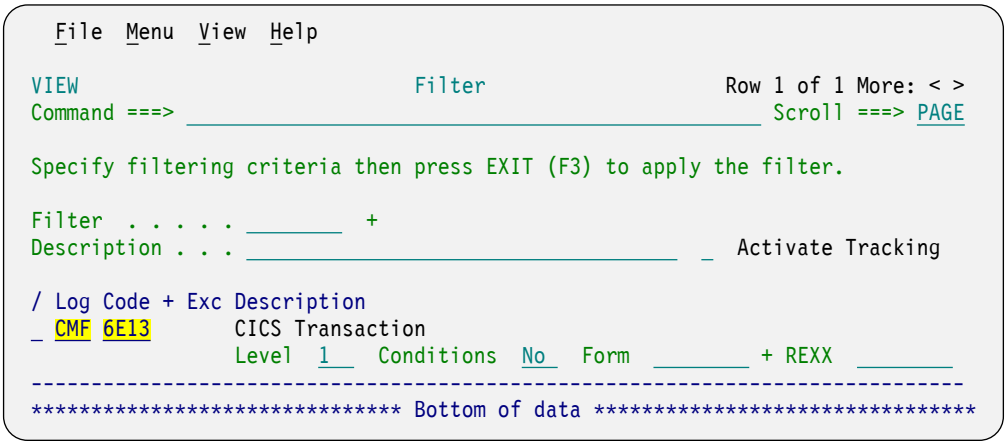


Figure 92. Panel: Specifying a filter to show only CMF records

In Transaction Analysis Workbench, 6E13 is a derived log code that identifies CICS monitoring facility (CMF) performance class records:

- The first two digits, 6E, are the hexadecimal representation of the decimal SMF record type, 110
- The third digit represent the record subtype, 1 (CICS monitoring)
- The fourth digit represents the class of data, 3 (performance)

7. Press the Exit function key (F3) to apply the filter and return to the log browser.

   The log browser now shows only the records that match the filter.

8. Press the Right function key (F11) to expand the display of each log record from a single line to multiple lines.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP006.CICS.D131008.INDEX  Record 00000741 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 14.56.25.116977
/       Filtering                           Tuesday 2013-10-08 Time (LOCAL)
TX 6E13 CICS Transaction                                        15.28.17.693992
        TranCode=FB66 Program=FBOCCP66 Userid=TWM LTerm=SC0TCP07 Terminal=CP07
        RecToken=FUWTCIC/CC145FF1F9E7C984 Resp=1.516358 CPU=0.006805 DB2=2
        ACCT=FTS3.SC0TCP07.145FF1F9E7C9 Task=251

__ 6E13 CICS Transaction                                        15.28.26.395768
        TranCode=FB66 Program=FBOCCP66 Userid=TWM LTerm=SC0TCP07 Terminal=CP07
        RecToken=FUWTCIC/CC145FFA465C6704 Resp=1.168714 CPU=0.006750 DB2=2
        ACCT=FTS3.SC0TCP07.145FFA465C67 Task=253


...
```

Figure 93. Panel: Selecting a CICS transaction to track

Notice the long response time shown by the Resp= field, and the nonzero DB2 request count shown by the DB2= field indicating that this is a CICS-DB2 transaction.

9. Enter TX next to the first CICS transaction (CMF) record.

   The TX line action activates transaction tracking, which displays the records that belong to the same transaction as the selected record.

   This shows us the complete, detailed life cycle of the CICS-DB2 transaction, starting with the CMF record for the CICS transaction, followed by the DB2 activity performed by that CICS transaction.

10. Enter E next to any of the records.

    The **Time** column displays elapsed times between each log record. To return to displaying wall-clock times, enter line action W next to any record.

11. Press the Right function key (F11) to collapse the display of each log record to a single line.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP006.CICS.D131008.INDEX  Record 00000741 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 14.56.25.116977
/       Tracking                            Tuesday 2013-10-08 Time (Elapsed)
__ 6E13 CICS Transaction TranCode=FB66 Task=251                  15.28.17.693992
__ 072  Create thread start                                DBA6        0.003592
__ 112  Thread allocate FBODCP06                           DBA6        0.000451
__ 073  Create thread end                                  DBA6        0.000052
__ 177  Package allocation FBODCP06                        DBA6        0.000187
__ 053  SQL request                SQLCODE=0 STMT=000158 DBA6        0.000165
__ 380  SP entry FBOSP006                    STMT=000196 DBA6        0.000018
__ 177  Package allocation FBOSP006                        DBA6        0.000649
__ 055  SQL set current SQLID                              DBA6        0.000563
__ 053  SQL request                SQLCODE=0 STMT=000281 DBA6        0.000041
__ 060  SQL SELECT                           STMT=000344 DBA6        0.000166
__ 058  SQL SELECT                 SQLCODE=0 STMT=000344 DBA6        1.502353
__ 061  SQL UPDATE                           STMT=000423 DBA6        0.000387
__ 0020 Begin UR                                                     0.000549
__ 0600 Savepoint                                                    0.000016
__ 0600 Update in-place in a data page                               0.000000
__ 058  SQL UPDATE                 SQLCODE=0 STMT=000423 DBA6        0.000298
__ 499  SP statement execution detail                      DBA6        0.000299
__ 380  SP exit  FBOSP006          SQLCODE=0 STMT=000196 DBA6        0.000012
__ 053  SQL request                SQLCODE=0 STMT=000196 DBA6        0.000048
__ 088  Sync start                                         DBA6        0.002284
__ 0020 Begin commit phase 1                                         0.000024
__ 0020 Phase 1 to 2 transition                                      0.000112
__ 0020 End commit phase 2                                           0.000656
__ 089  Sync end                                           DBA6        0.000184
__ 074  Terminate thread start                             DBA6        0.000090
__ 239  Package accounting-SP                              DBA6        0.000080
__ 003  Thread accounting                                  DBA6        0.000028
__ 075  Terminate thread end                               DBA6        0.000532
****************************** Bottom of Data ********************************
```

Figure 94. Panel: Tracking a CICS-DB2 transaction

Notice the jump in elapsed time between the 060 SQL SELECT DB2 trace record and its corresponding 058 end record, inside stored procedure (SP) FBOSP006.

12. Scroll the panel to display the 058 record at the top.
13. Press the Right function key (F11) to expand the display of each log record.

```
__ 058  SQL SELECT                 SQLCODE=0 STMT=000344 DBA6        1.502353
        TranCode=FB66 Userid=TWM ClientID=FUWTCIC
        ACCT=FTS3.SC0TCP07.145FF1F9E7C9 ScanSEQD=780753
        LUWID=FTS3/DBA6LU/CC145FF1FAB4/0001
```

Figure 95. Panel: Tracking a CICS-DB2 transaction

Notice the ScanSEQD field, indicating that the SQL SELECT statement performed a sequential data (SEQD) scan of 780753 rows. This high number explains the reason for the long elapsed time, and the long overall response time.

The DB2 accounting record (IFCID 003), just before the end of the DB2 thread, contains information about the thread activity. This information can offer useful insights into problems, and is often a good place to start analysis in the absence of more specific or immediately obvious indicators. For example, in addition to showing the long response and CPU time that we already know about, the DB2 accounting record for this thread shows a high number of get page requests:

```
__ 003  Thread accounting                                  DBA6        0.000028
        TranCode=FB66 Userid=TWM ClientID=FUWTCIC
        RESP=1.510268 CPU1=0.001418 CPU2=0.000968 I/O3=0.000328
        ACCT=FTS3.SC0TCP07.145FF1F9E7C9 Source=CICS SEL=1 UPD=1 CAL=1
        LogRecs=6 GetPage=14616 UpdPage=1 MaxLock=2
        LUWID=FTS3/DBA6LU/CC145FF1FAB4/0002
```

Figure 96. Panel: Tracking a CICS-DB2 transaction

This value is likely to be related to the SQL SELECT statement activity.

Note: The DB2 subsystem parameter **ACCUMACC** ("accumulate accounting") determines whether DB2 accounting records represent a single thread (ACCUMACC(NO)) or "roll up" a number of threads. If you specify an **ACCUMACC** value other than NO, you lose the per-thread granularity of this data.

The long response time was caused by an SQL SELECT statement in a stored procedure performing a table scan of over 780 thousand rows. Having diagnosed the problem, you can pass the details, including the name of the stored procedure and the SQL statement number, on to a DB2 subject-matter expert to fix the problem.

Chapter 66. Analyzing a CICS-DBCTL transaction problem

A user has reported that a CICS transaction failed. As a technical support staff member, you happen to know that this particular transaction is a CICS-DBCTL transaction. We will use Transaction Analysis Workbench CICS-DBCTL reports to help diagnose the problem, and then use the log browser to investigate the problem in detail.

The analysis procedure described here assumes that you have already performed the following steps in Transaction Analysis Workbench:

1. Created a session for the problem.
2. Run workflow tasks, inherited from the session template on which you based the session, that perform the following steps:
   a. Select the log files required to analyze the problem, containing log records for the period when the problem occurred:
      - The SMF file, or the name of the SMF log stream, that contains the records written by the CICS system
      - The IMS log
   b. Create CICS-DBCTL reports.

1. Look at the CICS-DBCTL combined summary report:

```
                                          CICS-DBCTL Summary

Tran  APPLID     CMF Count  Response  CPU Time  IMS Reqs  IMS Wait     ABEND  Rate/Sec
----  --------   ---------  --------  --------  --------  --------  --------  --------
FBOU  FUWTCIC           95  1.853189  0.348989        15  1.450194         8         0

                  08 Count   Elapsed  CPU Time  StaDelay  Schedule   IC Wait   PS Wait
                 ---------  --------  --------  --------  --------  --------  --------
                        95  1.847830  0.346223  0.004024  0.000217         0         0

                  07 Count   DB call   DB Gets   DB Upds  IO Count   IO Time  LockWait
                 ---------  --------  --------  --------  --------  --------  --------
                        95        13         5         7         3  0.000961  1.442473

                            Synctime   Phase 1   Phase 2    FP PH2   OTHREAD
                            --------  --------  --------  --------  --------
                            0.001572  0.001240  0.000331         0

other transaction/APPLID combinations…
```

Figure 97. CICS-DBCTL combined summary report

Notice the following points:

- `CMF Count` matches `08 Count`. This indicates that the problem might be in IMS rather than CICS. For every CICS-DBCTL transaction that had a problem, there was a problem in IMS.
- `LockWait`, the time that IMS waited due to a lock, accounts for most of `IMS Wait`, the time that CICS waited for IMS to respond.

This summary report gives us some idea of the likely cause of the problem: contention in IMS.

Next, we will look at individual transactions in list reports.

2. Look at the CICS-DBCTL list report. In particular, look at one of the transactions that abended, and make a note of its task number.

```
                                        CICS-DBCTL List

Tran  Task Num  Start Time       Response  CPU Time  IMS Reqs  IMS Wait  ABEND  APPLID   RMUOWID
----  --------  ---------------  --------  --------  --------  --------  -----  -------- ----------------
⋮
FBOU       965  17.30.15.908554  10.48227  4.063838        23  5.935800  ADCD   FUWTCIC  CC32A7F83B613604
⋮
```

Figure 98. CICS-DBCTL list report

CICS issues abend code ADCD when a deadlock has been detected by IMS.

3. Look at the corresponding DBCTL thread in the IMS-DBCTL list report; in this example, for CICS task number 965.

```
V1R1M0          2011-04-06 Wednesday           IMS-DBCTL List                          Page   1

Tran Task Num  PSB name  Start Time        Elapsed  CPU Time  FF Get  FF Upd  FP Get  FP Upd  ABEND  APPLID   RMUOWID
---- --------  --------  ---------------  --------  --------  ------  ------  ------  ------  -----  -------- ----------------
⋮
FBOU      965  FBOSCHED  17.30.15.912123  10.47055  4.058369       9      12                  U0777  FUWTCIC  CC32A7F83B613604
⋮
```

Figure 99. IMS-DBCTL list report

The IMS-DBCTL list report confirms the cause of the abend from the perspective of IMS: IMS issues abend code U0777 when a potential resource is in the deadlock condition.

Next, we will interactively browse the log files to examine the problem in more detail.

The workflow tasks inherited from the session template created a CICS transaction index and an IMS transaction index from the original log files.

4. Browse a combined time slice of the CICS transaction index, the IMS transaction index, and the original IMS log file (from which the IMS transaction index was created). Specify a time slice that spans the period of the transaction that we have seen in the CICS-DBCTL reports.
5. On the command line, enter TRACE 0FF to switch off the display of IMS trace records.

   For now, we want to hide IMS trace table entries from the display, so that we can concentrate on other records.
6. Scroll to the CICS transaction log record (log code 6E13) for the transaction that we identified in the reports.

   There are various ways to do this. For example, on the command line, enter the following command:

   ```
   F TASK=965
   ```
7. Enter line action TX next to the 6E13 record, to track the records for that transaction.

   The log browser shows only the log records that are related to the selected CICS transaction; in this case, the selected CICS transaction index (6E13) record, the IMS transaction index (CA01) record, and the related IMS log records.

   Note: A CICS transaction index record has the same format as the original CMF record in an SMF file, so they have the same log code. An IMS transaction index has its own unique log code (CA01) because it consolidates multiple IMS log records.
8. Enter line action R next to the 6E13 record, to show record time stamps relative to the start of the CICS transaction.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.DBCTLDLK.SMF.D131031.CMFX +  Record 00001645 More: < >
Command ===> _____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-11-01 16.59.41.444468
/       Tracking                             Friday 2013-11-01 Time (Relative)
__ 6E13 CICS Transaction TranCode=FBOU Task=965 Abend=ADCD     17.30.15.908554
__ 08   Application Start TranCode=FBOU Program=FBOSCHED             +0.003573
__ CA01 IMS Transaction Program=FBOSCHED LTerm=FUWTCIC               +0.003573
__ 5607 Start of UOR Program=FBOSCHED Region=0002                    +0.003576
__ 5050 Database DLET Database=FBOIAD01 Region=0002                  +0.006865
__ 5050 Database DLET Database=FBOIAD01 Region=0002                  +0.006890
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                  +0.007426
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                  +0.007434
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                  +0.007438
__ 5050 Database DLET Database=FBOIAD01 Region=0002                  +4.535575
__ 5050 Database DLET Database=FBOIAD01 Region=0002                  +4.535595
__
...
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                  +4.546043
__ 67FF Exception Condition SNAP - DEADLOCK                         +10.470720
__ 38   Release Input Message after Application ABEND               +10.471213
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                 +10.472584
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                 +10.472621
__ 5050 Database DLET Database=FBOIAD01 Region=0002                 +10.472636
__ 5050 Database DLET Database=FBOIAD01 Region=0002                 +10.472640
__ 5050 Database DLET Database=FBOIAD01 Region=0002                 +10.472643
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                 +10.472649
__ 5050 Database ISRT Database=FBOIAD01 Region=0002                 +10.472657
__ 5050 Database DLET Database=FBOIAD01 Region=0002                 +10.472665
__ 5050 Database DLET Database=FBOIAD01 Region=0002                 +10.472667
```

Figure 100. Panel: Tracking a CICS-DBCTL transaction

Notice the two jumps in elapsed time between records:

- A delay of 4.5 seconds between an insert and a delete
- A delay of nearly 6 seconds after an insert request, resulting in a deadlock

The first delay resolved itself, but is still worth investigating. The second delay is more serious, because it resulted in an abend. We will concentrate on that problem.

9. Press the Right function key (F11) to expand the display of each log record from a single line to multiple lines.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.DBCTLDLK.SMF.D131031.CMFX +  Record 00001645 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-11-01 16.59.41.444468
/       Tracking                             Friday 2013-11-01 Time (Relative)
__ 6E13 CICS Transaction                                        17.30.15.908554
        TranCode=FBOU Program=FBO$UPD Userid=TWM2 LTerm=SC0TCP30 Terminal=CP30
        RecToken=FUWTCIC/CC32A7F83B613604 Resp=10.482274 CPU=4.063838 IMS=23
        ACCT=FTS3.SC0TCP30.32A7F83B6136 Task=965 Abend=ADCD

...
__ CA01 IMS Transaction                                          +0.003573
        UTC=17.30.15.912123 TranCode=FBOU Program=FBOSCHED Userid=TWM2
        LTerm=FUWTCIC Region=0002 SSID=IDB1 IMSRel=131
        RecToken=FUWTCIC/CC32A7F83B613604 CPU=4.058369 Process=10.470559
        TotalTm=10.470559 RegTyp=DBC DBCalls=21 Abend=U0777

...
__ 67FF Exception Condition SNAP - DEADLOCK                     +10.470720
        UTC=17.30.26.379071 Region=0002
        Winner: IMS=IDB1 Job/Tran=FUWTCIC  PST=0001 PSB=FBOSCHED DMB=FBOIAD01
        Victim: IMS=IDB1 Job/Tran=FUWTCIC  PST=0002 PSB=FBOSCHED DMB=FBOIAD01
```

Figure 101. Panel: Tracking a CICS-DBCTL transaction: deadlock winner

For the 67FF record, the expanded display includes details of the deadlock loser (this transaction) and winner. You, or the CICS-DBCTL administrators or developers at your site, can use the PSB of the winner to identify the transaction that is contending with this transaction for the same resources, resulting in a deadlock. You might choose to change the behavior or either or both transactions to avoid deadlocks.

In this example, the winner and loser have the same PSB, because of the design of the test application used to generate these log records.

Next, we will use IMS trace records to identify the database call that caused the deadlock.

10. On the command line, enter TRACE ON to show IMS trace records.

    The IMS trace records include records for each database call (log code AA).

    The database call that caused the deadlock is the last AA record before the 67FF "SNAP - DEADLOCK" record:

```
__ AA   DLI Database call: DLET Region=0002                      +10.470692
__ 67FF Exception Condition SNAP - DEADLOCK                      +10.470720
```

Figure 102. Panel: Tracking a CICS-DBCTL transaction: IMS trace record showing database call that caused the deadlock

    You can match the sequence of AA records with your application source code to determine exactly which database call in your application caused the deadlock.

    The remaining steps in this scenario assume that IMS Performance Analyzer is installed on your system.

11. Press the Exit function key (F3) until you return to the session menu.
12. On the session menu, select option 4 **Reporting**.

    The **Reporting** menu is displayed.

13. Select option 1 **IMS**.
14. Select **Deadlock analysis**.
15. Select the IMS log file.

16. Press Enter twice to generate the JCL for the report.

    The JCL is displayed in a **Notepad** panel.
17. Submit the job: enter **SUB** on the command line.
18. View the job output in the session workflow: return to the session menu, select option 2 **Workflow**, and then enter ? next to the corresponding task. Specifically, look at the Deadlock List report in the DEADLOCL output data set.

    The following figure shows an example report.

```
Start 20Apr2010 11.01.17.75                        IMS Performance Analyzer
Page      1
                                                          Deadlock List
                                                          -------------

Pseudo abend record              Abend No = U0777      Time   11:01:17:76  Date 20Apr2010      Recno = 00000000000001E8

Deadlock Analysis Report - Lock Manager is IRLM
.....................................................................................................................
............

Resource DMB-name Lock-len Lock-name
01 of 02 DI21PART    08    D1CD77C3800601D7 (RBA = D1CD77C3, DMB# = 8006, DCB = 01, P-Lock)

Key is root key of data base record associated with lock...
000000  F0F2F6F0 F0F0F360  F1F1F840 40404040   40                                        *0260003-118      *

       IMS-name Tran/Job PSB-name PCB--DBD PST#  RGN  Call Lock  State             Lockfunc
Blcker IBB1     FUWTCIC  DFHTWM04          00001 DBT             06-P (Update,Pri)
Waiter IBB1     FUWTCIC  DFHTWM04 DI21PART 00002 DBT  GET  GRIDX 06-P (Update,Pri) 30400358 Func=Get Local and Global
Root Locks
                                                                                                 Mode=Uncond
                                                                                                 State=Update
                                                                                                 Flag=Get,Single,P-Lock
.....................................................................................................................
............

Resource DMB-name Lock-len Lock-name    ** Waiter for this resource is VICTIM **
02 of 02 DI21PART    08    00004001800601D7 (RBA = 00004001, DMB# = 8006, DCB = 01, P-Lock)

Key is root key of data base record associated with lock...
000000  F0F2F6F0 F0F0F360  F1F1F840 40404040   40                                        *0260003-118      *

       IMS-name Tran/Job PSB-name PCB--DBD PST#  RGN  Call Lock  State             Lockfunc
Blcker IBB1     FUWTCIC  DFHTWM04          00002 DBT             06-P (Update,Pri)
Waiter IBB1     FUWTCIC  DFHTWM04 DI21PART 00001 DBT  DLET GBIDP 03-P (Read,Pri)   22400358 Func=Get Global Buffer
Update Lock
                                                                                                 Mode=Uncond
                                                                                                 State=Update
                                                                                                 Flag=Get,Single,P-Lock

Deadlock Analysis Report - End of Report
```

Figure 103. Example IMS Performance Analyzer Deadlock List report

Related concepts

CICS-DBCTL transactions

You can analyze response time problems in CICS-DBCTL transactions using CICS monitoring facility (CMF) performance class records (SMF type 110 records), IMS log records, or a combined view of both.

Chapter 67. Analyzing an IMS-DB2 transaction problem

Users have reported long response times from an application. As a technical support staff member, you know that the application runs IMS transactions that perform DB2 updates.

The analysis procedure described here assumes that you have already performed the following steps in Transaction Analysis Workbench:

1. Created a session for the problem.
2. Run workflow tasks, inherited from the session template on which you based the session, that perform the following steps:
   a. Select the log files required to analyze the problem, containing log records written by the related IMS and DB2 systems for the period when the problem occurred.
   b. Create an IMS transaction index: an extract of an IMS log that consolidates multiple records for each transaction into one record per transaction.
   c. Create an IMS Performance Analyzer batch report, as a starting point for analysis.
3. Manually add to the session some related log files that were not automatically added by the tasks.

To analyze an IMS-DB2 transaction problem, you typically need the following log files:

- IMS log
- SMF file, or SMF log stream, containing at least the following types of record:
  - DB2 accounting (SMF type 101) records (IFCIDs 003 and 239)
  - DB2 performance (SMF type 102) records (IFCIDs belonging to trace level 2, as defined by Transaction Analysis Workbench)
- Archived DB2 log

Note: For DB2 accounting and performance records, instead of using SMF files, you can use GTF data sets.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 1 **Sessions**, and then select the session that you have created for this problem.
2. On the session menu, select option 2 **Workflow**.

   We will begin analyzing the problem by looking at the IMS Performance Analyzer report that has already been created by one of the workflow tasks.
3. Enter **?** next to the task for the report.

```
  File  Help

                                  Tasks                      Row 1 to 5 of 5
Command ===> ____________________________________________ Scroll ===> PAGE

 NEW   Create a new task
 AUTO  Create file selection and extract tasks
 SCHED Schedule all the tasks (or select required tasks only)

/  Task Status      Description
__   1  DONE        DB2 log file selection for DBA6
__   2  CC 0000     SMF file selection for DBA6
__   3  CC 0000     IMS log file selection for IDDG
__   4  CC 0000     Create the IMS transaction index
?_   5  CC 0000     IMS transaction and system analysis report
****************************** Bottom of data ********************************
```

*Figure 104. Panel: **Tasks**: selecting a batch job task to view its output*

The list of output data sets for that task is displayed.

```
  File  Help

                               Task Output                   Row 1 to 8 of 8
Command ===> ____________________________________________ Scroll ===> PAGE

/   DDname    StepName ProcStep      Rec-Cnt  Jobname  JobID     Max-RC
_   JESMSGLG  JES2                        32  JCH#FUW  JOB46414  CC 0000
_   JESJCL    JES2                        25
_   JESYSMSG  JES2                        91
_   SYSPRINT  IMSPA                       82
_   LOGINFO   IMSPA                        8
_   LIST0001  IMSPA                       27
S   LIST0004  IMSPA                       27
_   SYSPRINT  SUBMIT                       4
****************************** Bottom of data ********************************
```

*Figure 105. Panel: **Task Output**: list of output data sets for a batch job task*

4. Enter **?** next to the LIST0004 data set.

   The IMS Performance Analyzer "Transaction detail: External Subsystem" report is displayed:

   Figure 106. IMS Performance Analyzer Transaction detail: External Subsystem report

```
IMS Tran                                        CPU  Process    Total ESAF
Start           Trancode Userid    PST     Time     Time IMS Time Name
17.10.09.284078 FBOIAT41 FUNTRM10    2 45.69955 72.61228 72.61294 DBA6
17.15.12.276470 FBOIAT41 FUNTRM10    1 0.004247 0.007591 0.008006 -
17.15.19.060177 FBOIAT41 FUNTRM10    2 11.51239 18.10520 18.10559 DBA6
17.15.45.907312 FBOIAT41 FUNTRM10    1 11.58205 23.36967 23.37007 DBA6
17.16.20.310281 FBOIAT41 FUNTRM10    2 11.67014 26.57243 26.57280 DBA6
17.18.10.042958 FBOIAT41 FUNTRM10    1 11.57455 28.23666 28.23708 DBA6
17.18.43.971732 FBOIAT41 FUNTRM10    2 11.47511 23.54669 23.54710 DBA6
⋮
```

   The report includes several IMS transactions around the time of the reported problem. The first IMS transaction looks particularly bad: long CPU, process, and total IMS time. We will take a closer look at that transaction in the log browser.

5. Press the Exit function key (F3) until you return to the session menu.
6. On the session menu, select option 5 **Investigate**.

   The **Investigate** panel is displayed.

7. If the **Time Slice** heading shows **(OFF)**, enter SLICE on the command line to switch time slicing on.

For our time slice, we are going to use the full duration of the IMS transaction index, which is the same as the duration of the IMS log from the index was created.

```
  File  Menu  Time Slicing  Help

                                Investigate              Row 1 of 4 More: < >
Command ===> ____________________________________________ Scroll ===> PAGE

                             Time Slice (ON)
               Time            Date           Duration
               HH.MM.SS.thmiju YYYY-MM-DD     HH.MM.SS  Zone   Filter +
               17.10.09.284086 2013-10-08     00.14.45  LOCAL  ________
/
________  Type  Data Set Name                                    Coverage
T_______  IMS   FUW000.QADATA.FBOSP007.IMS.D131008.INDEX        COMPLETE
________  IMS   FUW000.QADATA.FBOSP007.IMS.D131008.SLDS         COMPLETE
________  DB2   FUW000.QADATA.FBOSP007.DB2.D131008.ARCLOG       COMPLETE
________  SMF   FUW000.QADATA.FBOSP007.SMF.D131008.FULL         PARTIAL
****************************** Bottom of data ********************************
```

*Figure 107. Panel: **Investigate**: setting the time slice to match the period covered by a log file*

8. Enter **T** next to the IMS transaction index.
9. Enter **S** on the first line under the **/** heading, to browse a merged view of all log files.

   The log files are displayed in the log browser. IMS transaction index records have the log code CA01.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
       Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/                                          Tuesday 2013-10-08 Time (LOCAL)
__ CA01 IMS Transaction TranCode=FBOIAT41 Region=0002          17.10.09.284086
__ 01   Input Message TranCode=FBOIAT41                        17.10.09.284086
__ 35   Input Message Enqueue TranCode=FBOIAT41                17.10.09.284110
__ 08   Application Start TranCode=FBOIAT41 Region=0002        17.10.09.284366
__ 5607 Start of UOR Program=FBOIAP41 Region=0002              17.10.09.284367
__ 31   DLI GU TranCode=FBOIAT41 Region=0002                   17.10.09.284390
__ 5616 Start of protected UOW Region=0002                     17.10.09.284579
__ 086  Signon start                                     DBA6  17.10.09.290369
__ 087  Signon end                                       DBA6  17.10.09.290445
__ 5600 Sign-on to ESAF Region=0002 SSID=DBA6                  17.10.09.290476
__ 5600 Thread created for ESAF SSID=DBA6                      17.10.09.290488
__ 072  Create thread start                              DBA6  17.10.09.290500
__ 112  Thread allocate FBOIAP41                         DBA6  17.10.09.291061
__ 073  Create thread end                                DBA6  17.10.09.291130
__ 122  Thread level exit from DB2                       DBA6  17.10.09.291146
__ 121  Thread level entry into DB2                      DBA6  17.10.09.291210
__ 177  Package allocation FBOIAP41                      DBA6  17.10.09.291357
__ 233  SP entry FBOSP007                    STMT=001031 DBA6  17.10.09.291592
```

*Figure 108. Panel: **Investigate**: setting the time slice to match the period covered by a log file*

In this example, the record that we want to investigate happens to be the first record in the merged view of log files, and is displayed at the top of the panel.

In practice, depending on the time slice that you specify and the periods covered by the log files, this rarely happens.

Typically, unless you know the exact time stamp of the record that you want to investigate (again, here, we do, because we have already run a batch report that gives us this information), we would filter the log browser to show only IMS transaction index records, so that we can then select a single instance of an IMS transaction for closer inspection.

We will do that now anyway, so that we can see how we might select a transaction if we did not already know exactly which one we were interested in.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
       Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/                                          Tuesday 2013-10-08 Time (LOCAL)
__ CA01 IMS Transaction TranCode=FBOIAT41 Region=0002          17.10.09.284086
__ 01   Input Message TranCode=FBOIAT41                        17.10.09.284086
__ 35   Input Message Enqueue TranCode=FBOIAT41                17.10.09.284110
__ 08   Application Start TranCode=FBOIAT41 Region=0002        17.10.09.284366
__ 5607 Start of UOR Program=FBOIAP41 Region=0002              17.10.09.284367
__ 31   DLI GU TranCode=FBOIAT41 Region=0002                   17.10.09.284390
__ 5616 Start of protected UOW Region=0002                     17.10.09.284579
__ 086  Signon start                                     DBA6 17.10.09.290369
__ 087  Signon end                                       DBA6 17.10.09.290445
__ 5600 Sign-on to ESAF Region=0002 SSID=DBA6                  17.10.09.290476
__ 5600 Thread created for ESAF SSID=DBA6                      17.10.09.290488
__ 072  Create thread start                              DBA6 17.10.09.290500
__ 112  Thread allocate FBOIAP41                         DBA6 17.10.09.291061
__ 073  Create thread end                                DBA6 17.10.09.291130
__ 122  Thread level exit from DB2                       DBA6 17.10.09.291146
__ 121  Thread level entry into DB2                      DBA6 17.10.09.291210
__ 177  Package allocation FBOIAP41                      DBA6 17.10.09.291357
__ 233  SP entry FBOSP007                    STMT=001031 DBA6 17.10.09.291592
```

*Figure 109. Panel: **Investigate**: setting the time slice to match the period covered by a log file*

10. Press the Filter function key (F18) or enter **FILTER** on the command line.

    The **Filter** panel is displayed. You can use filters to limit the log browser to displaying only the records that interest you.

11. Specify IMS under the **Log** column heading and CA01 under the **Code** column heading, and then press Enter.

```
  File  Menu  View  Help

VIEW                          Filter                  Row 1 of 1 More: < >
Command ===> ____________________________________________ Scroll ===> PAGE

Specify filtering criteria then press EXIT (F3) to apply the filter.

Filter  . . . . . ________  +
Description . . . ____________________________________  _  Activate Tracking

/ Log Code + Exc Description
_ IMS CA01       IMS Transaction
                 Level  1   Conditions  No   Form  ________ + REXX  ________
------------------------------------------------------------------------------
****************************** Bottom of data ********************************
```

*Figure 110. Panel: **Filter**: setting a filter*

12. Press the Exit function key (F3) to apply the filter and return to the log browser.

    The log browser now shows only the records that match the filter.

13. Press the Right function key (F11) to expand the display of each log record from a single line to multiple lines.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/       Filtering                           Tuesday 2013-10-08 Time (LOCAL)
TX CA01 IMS Transaction                                        17.10.09.284086
        UTC=17.10.09.284078 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
        LTerm=FUNTRM10 Terminal=SC0TCP10 Region=0002
        OrgUOWID=IDDG/CC1476B6713CB884 SSID=IDDG IMSRel=131
        RecToken=IDDG/0000000400000000
        CPU=45.699549 InputQ=0.000309 Process=72.612278 OutputQ=0.000356
        TotalTm=72.612943 RegTyp=MPP

__ CA01 IMS Transaction                                        17.15.12.276476
        UTC=17.15.12.276470 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
        LTerm=FUNTRM10 Terminal=SC0TCP10 Region=0001
        OrgUOWID=IDDG/CC1477D765FD2406 SSID=IDDG IMSRel=131
        RecToken=IDDG/0000000500000000
        CPU=0.004247 InputQ=0.000361 Process=0.007591 OutputQ=0.000054
        TotalTm=0.008006 RegTyp=MPP

__ CA01 IMS Transaction                                        17.15.19.060184
        UTC=17.15.19.060177 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
```

Figure 111. Panel: Browsing IMS transaction index records (after defining a filter)

This expanded view shows some of the significant details that we saw earlier in the batch report: long CPU, process, and total (input queue + process + output queue) times.

14. Enter TX next to the first record.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> ____________________________________________ Scroll ===> CSR
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/       Filtering                           Tuesday 2013-10-08 Time (LOCAL)
TX CA01 IMS Transaction                                        17.10.09.284086
        UTC=17.10.09.284078 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
        LTerm=FUNTRM10 Terminal=SC0TCP10 Region=0002
        OrgUOWID=IDDG/CC1476B6713CB884 SSID=IDDG IMSRel=131
        RecToken=IDDG/0000000400000000
        CPU=45.699549 InputQ=0.000309 Process=72.612278 OutputQ=0.000356
        TotalTm=72.612943 RegTyp=MPP

__ CA01 IMS Transaction                                        17.15.12.276476
        UTC=17.15.12.276470 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
        LTerm=FUNTRM10 Terminal=SC0TCP10 Region=0001
        OrgUOWID=IDDG/CC1477D765FD2406 SSID=IDDG IMSRel=131
        RecToken=IDDG/0000000500000000
        CPU=0.004247 InputQ=0.000361 Process=0.007591 OutputQ=0.000054
        TotalTm=0.008006 RegTyp=MPP

__ CA01 IMS Transaction                                        17.15.19.060184
        UTC=17.15.19.060177 TranCode=FBOIAT41 Program=FBOIAP41 Userid=FUNTRM10
```

Figure 112. Panel: Browsing IMS transaction index records (after defining a filter)

The TX line action activates tracking, which displays the records that belong to the same transaction as the selected record. This shows us the complete, detailed life cycle of the IMS-DB2 transaction, starting with the IMS transaction index record, followed by the related IMS and DB2 activity performed

15. Press the Right function key (F11) to collapse the display of each log record to a single line.
16. Enter E next to any of the records.

The **Time** column displays elapsed times between each log record. To return to displaying wall-clock times, enter line action W next to any record.

```
  File  Mode  Filter  Time  Labels  Options  Help

BROWSE    FUW000.QADATA.FBOSP007.IMS.D131008.INDEX   Record 00000226 More: < >
Command ===> _____________________________________________ Scroll ===> PAGE
        Navigate < 00.00.01.000000 >      Date/Time 2013-10-08 17.10.09.284086
/       Tracking                            Tuesday 2013-10-08 Time (Elapsed)
__ CA01 IMS Transaction TranCode=FBOIAT41 Region=0002          17.10.09.284086
__ 01   Input Message TranCode=FBOIAT41                               0.000000
__ 35   Input Message Enqueue TranCode=FBOIAT41                       0.000023
__ 08   Application Start TranCode=FBOIAT41 Region=0002               0.000256
__ 5607 Start of UOR Program=FBOIAP41 Region=0002                     0.000000
__ 31   DLI GU TranCode=FBOIAT41 Region=0002                          0.000022
__ 5616 Start of protected UOW Region=0002                            0.000189
__ 5600 Sign-on to ESAF Region=0002 SSID=DBA6                         0.005896
__ 5600 Thread created for ESAF SSID=DBA6                             0.000012
__ 112  Thread allocate FBOIAP41                          DBA6        0.000572
__ 073  Create thread end                                 DBA6        0.000068
__ 177  Package allocation FBOIAP41                       DBA6        0.000227
__ 380  SP entry FBOSP007                      STMT=001031 DBA6        0.000023
__ 177  Package allocation FBOSP007                       DBA6        0.000184
__ 061  SQL UPDATE                             STMT=000001 DBA6        0.000141
__ 0020 Begin UR                                                      0.001034
__ 0600 Savepoint                                                     0.000000
__ 0600 Update in-place in a data page                                0.000000
__ 058  SQL UPDATE                 SQLCODE=0 STMT=000001 DBA6        0.000338
__ 065  SQL OPEN C1                            STMT=000001 DBA6        0.000090
__ 058  SQL OPEN                   SQLCODE=0 STMT=000001 DBA6        0.000021
__ 499  SP statement execution detail                     DBA6        0.000039
__ 380  SP exit  FBOSP007          SQLCODE=0 STMT=001031 DBA6        0.000012
__ 053  SQL request               SQLCODE=466 STMT=001031 DBA6        0.000083
__ 053  SQL request                SQLCODE=0 STMT=001082 DBA6        0.000824
__ 053  SQL request                SQLCODE=0 STMT=001085 DBA6        0.000119
__ 059  SQL FETCH C1                           STMT=001090 DBA6        0.000107
__ 0600 Savepoint                                                     1.437546
__ 0600 Savepoint                                                     0.257680
__ 0600 Savepoint                                                     1.059456
__ 0600 Savepoint                                                     0.000032
__ 0600 Savepoint                                                     0.000016
__ 0600 Savepoint                                                     0.000016
__ 058  SQL FETCH                  SQLCODE=0 STMT=001090 DBA6     1.09.840951
__ 053  SQL request                SQLCODE=0 STMT=001090 DBA6        0.000112
__ 059  SQL FETCH C1                           STMT=001090 DBA6        0.000295
__ 058  SQL FETCH                 SQLCODE=100 STMT=001090 DBA6        0.000036
__ 053  SQL request               SQLCODE=100 STMT=001090 DBA6        0.000022
__ 5600 Commit Prepare starting Region=0002 SSID=DBA6                 0.001033
__ 084  Prepare start                                     DBA6        0.000604
__ 0020 End commit phase 1                                            0.000223
__ 085  Prepare end                                       DBA6        0.000519
__ 03   Output Message Response LTerm=FUNTRM10                        0.000082
__ 35   Output Message Enqueue LTerm=FUNTRM10 Region=0002             0.000012
__ 3730 Syncpoint End of Phase 1 Region=0002                          0.000016
__ 074  Terminate thread start                            DBA6
```

Figure 113. Panel: Tracking an IMS-DB2 transaction (showing elapsed time between records)

Notice the elapsed time caused by the first SQL FETCH C1 statement.

17. Scroll the display to set the 058 record for the first SQL FETCH statement at the top of the panel.

    Tip: To position a record at the top of the panel, enter T next to the record.

18. Press the Right function key (F11) to expand the records.

```
__ 058  SQL FETCH                    SQLCODE=0 STMT=001090 DBA6 17.11.21.890327
        TranCode=FBOIAP41 Userid=FUNTRM10 ClientID=IDDG
        ScanINDX=1280799 ScanSEQD=1280373 ScanSEQW=2734180
        LUWID=FTS3/DBA6LU/CC1476B672D1/0001
```

Figure 114. Panel: Tracking an IMS-DB2 transaction (SQL FETCH details)

Notice the Scan... fields, which show the number of rows scanned by this SQL statement. These high numbers explain the reason for the long elapsed time, and the long overall response time.

Let us look more closely at that SQL FETCH C1 statement; specifically, at the C1 cursor that it uses. Looking back at previous records in the DB2 thread, we can see that this cursor was declared earlier by a stored procedure. So, the SQL FETCH statement is manifesting a problem that was actually caused by the earlier stored procedure. If we look at the SQL statement source for that stored procedure, we can see that the real problem is an inefficient SQL SELECT statement used by that cursor.

The long response time was caused by an SQL FETCH statement. The SQL FETCH statement took a long time because it used a cursor, defined in a stored procedure, that used an inefficient SQL SELECT statement. Having diagnosed the problem, you can pass the details on to a DB2 subject-matter expert to fix the problem. (That expert might, for example, use the SQL EXPLAIN statement to explain the SQL SELECT statement.)

Related concepts

DB2 log records

DB2 log records contain detailed recovery information about each change operation performed against DB2 database objects. These changes are recorded as database *undo* (backout) records, *redo* (reapply) records, or a combination of both.

Part 12. Troubleshooting

Use these topics to diagnose and correct problems that you experience with Transaction Analysis Workbench.

Chapter 68. Messages

Use the information in these messages to help you diagnose and solve problems running Transaction Analysis Workbench batch utilities. For information on messages issued by the Transaction Analysis Workbench ISPF dialog, see the online help.

Message format

Transaction Analysis Workbench message identifiers have the following format:

```
FUWnnnnx
```

where:

FUW
Indicates that the message was issued by Transaction Analysis Workbench.

nnnn
Is the message identification number.

Message identification numbers are divided into the following ranges:

Range	Description
0001 - 0019	Transaction Analysis Workbench
0020 - 0049	Log record field processing: filters and forms
0050 - 0094	Knowledge modules
0095 - 0099	Export/Import - Filters and Forms
0100 - 0199	Dialog Browse API
0200 - 0299	Command input
0500 - 0599	Automated file selection
3000 - 3099	Control repository

x
Indicates the severity of the message:

I
Indicates that the message is informational only.

W
Indicates that the message is a warning to alert you to a possible error condition.

E
Indicates that an error occurred, which might or might not require operator intervention.

S
Severe. Transaction Analysis Workbench processing is suspended until you have taken action.

C
An internal Transaction Analysis Workbench error. Transaction Analysis Workbench immediately stops processing. Contact IBM Software Support.

Each message also includes the following information:

Explanation
Describes what the message text means, why the message occurred, and what its variables represent.

System action
Describes what the system will do in response to the event that triggered this message.

User response
Describes whether a response is necessary, what the appropriate response is, and how the response will affect the system or program.

Messages from other products

Transaction Analysis Workbench can invoke other products, resulting in messages with the following message identifier prefixes. For more information about these messages, refer to the documentation for that product.

Prefix
Product

ALZ
IMS Problem Investigator

CPA
CICS Performance Analyzer (batch reports)

DFS
IMS (when calling the IMS DBRC API to perform automated file selection for IMS logs)

DSN
DB2 (when running the DB2-supplied print log map utility, DSNJU004, to perform automated file selection for DB2 logs)

FUN
IMS Connect Extensions (functional support message) or Common Services Library server

IPI
IMS Performance Analyzer (batch reports)

FUW-prefixed messages

These are the messages from the Transaction Analysis Workbench batch utilities.

FUW0001E SYSPRINT DD is missing

Explanation

SYSPRINT DD was not specified in the JCL. SYSPRINT contains the system messages and runtime event log.

System action

Transaction Analysis Workbench immediately stops processing with RC=16. This message is issued by a WTO to the JOBLOG because no message output file is available.

User response

Specify SYSPRINT DD in the JCL then retry request.

FUW0002E SYSIN DD is missing

Explanation

SYSIN DD was not specified in the JCL. SYSIN specifies commands for the Report and Extract requests.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Specify the SYSIN DD in the JCL with the required report request commands then retry request.

FUW0003E OUTPUT DDname is missing from JCL; DDname=*ddname*

Explanation

The OUTPUT DDname was not specified in the JCL. The OUTPUT DDname is where report output or extract data is written.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Specify the OUTPUT DDname in the JCL then retry request.

FUW0004E **A supported IMS VRM is not specified in EXEC PARM or IMSVRM command**

Explanation:
A supported release of IMS is not specified in the JCL EXEC PARM= or the IMSVRM command. The IMS version is required so that the log records can be correctly interpreted.

System action:
Processing stops.

User response

Specify either:

1. PARM='Vvvr' on your job EXEC statement, or
2. IMSVRM=Vvvr command in the SYSIN deck.

Supported releases of IMS include version 11 upwards; V111, V121 and V131.

FUW0005I **Log File processing has started; DDname=*ddname,* Time=*yyyy-mm-dd hh.mm.ss.thmiju,* LSN=*lsn***

Explanation

The first Log record in the specified Log file data set has been read. Record processing does not commence until the Start time (if specified) is reached. The log record suffix (STCK time stamp and LSN) are formatted to help identify the start of this Log file.

System action

Processing continues.

User response

None required.

FUW0006I **Record processing has started; Time=*yyyy-mm-dd hh.mm.ss.thmiju,* LSN=*lsn***

Explanation

The first Log record after the required start time has been read and is about to be processed. The log record suffix (STCK time stamp and LSN) are formatted to help identify the first record.

System action

Processing continues.

User response

None required.

FUW0007I **Report processing has ended; DDname=*ddname*, Record count=*count***

Explanation:
Report processing to the specified output file ddname has completed. The Record count indicates the number of records reported. The message severity is escalated to a warning when the record count is zero.

System action:
Processing continues.

User response:
None.

FUW0008I **Extract processing has ended; DDname=*ddname*, Record count=*count***

Explanation:
Extract processing to the specified output file ddname has completed. The Record count indicates the number of records that were extracted. The message severity is escalated to a warning when the record count is zero. Subsequent job steps (for example sorting) that rely on the extract to contain data can be skipped with a condition code check, for example: `EXEC PGM=SORT,COND=(0,NE,EXTRACT)`.

System action:
Processing continues.

User response:
None, unless a warning is issued. Verify the date/time range and filtering criteria to ensure the input files contains matching records.

FUW0009I **Processing has completed, maximum RC=*xx***

Explanation

Processing of all requests has completed. The maximum condition code (return code) represents the highest severity of the messages that were issued:

- I=0=Processing completed successfully

- W=4=Processing completed with a warning
- E=8=Processing did not complete due to an error.

System action:
This message indicates the end of all processing.

User response:
The reasons for a non-zero completion can be determined from the warning or error messages issued.

FUW0010W **Report processing has reached page limit; DDname=*ddname***

Explanation:
Report processing for the specified output data set ddname has reached its page limit. No more records will be written to this file. The **PAGELIM** command specifies the page limit, and if not specified, defaults to 10,000.

System action:
Log file processing stops for this report. Other processing continues.

User response:
Increase the **PAGELIM** specification to meet your requirements. The format of the command is `PAGELIM(nnnnn)`.

FUW0011E **REXX exec not found; exec=*execname***

Explanation:
REXX processing for the specified exec could not be done. The exec may not exist in the specified SYSEXEC.

System action:
Processing stops.

User response:
Check SYSEXEC and the exec name are correct.

FUW0012E **REXX exec not processed; exec=*execname***

Explanation:
REXX processing for the specified exec could not be done. There may be an error with the arguments (ARG parameter).

System action:
Processing stops.

User response:
Check SYSEXEC, the exec name and the arguments are correct.

FUW0013E **REXX environment could not be initialised**

Explanation:
The REXX environment ALZEXEC failed to initialize.

System action:
Processing stops.

User response:
Internal error. Contact IBM Software Support.

FUW0014E **SYSTSPRT DD is missing**

Explanation:
REXX request is specified and the SYSTSPRT DD was not specified in the JCL.

System action:
Processing stops.

User response:
Specify SYSTSPRT DD in the JCL then retry request.

FUW0015E **Extract record length exceeds data set BLKSIZE; DDname=*ddname*, BLKSIZE=*blksize*, RecLen=*reclen***

Explanation

A log record to be extracted to the specified extract file has a record length exceeding the maximum allowed (4 less than BLKSIZE).

System action

Transaction Analysis Workbench processing terminates.

User response

Increase the Extract data set BLKSIZE specification.

FUW0016E **DB2 active log file error; DDname=*ddname*, Service=*service*, Ret=*ret*, Reas=*reas***

Explanation

An error occurred processing the specified DB2 active log file. The service for processing VSAM linear data sets failed with a bad return and reason code.

System action

Transaction Analysis Workbench processing terminates.

User response

Verify that the failing DDname refers to a valid DB2 active log file (VSAM linear data set). If the data set specification is correct, refer to the return and reason codes for the specified service to identify the cause of the problem.

FUW0017E **DB2 log record processing error; DDname=*ddname*, RBA=*rba*, STCK=*stck***

Explanation:
An error occurred during reconstruction of a DB2 log record from the specified DB2 log file. The specified RBA and STCK values (located at the end of the 4K physical record in the DB2 log) uniquely identify the record being processed.

System action:
Transaction Analysis Workbench processing terminates.

User response:
Verify that the failing ddname refers to a valid DB2 log (either active or archive). Use the **DSN1LOGP** utility to report and verify the contents of the DB2 log. Otherwise, contact IBM Software Support.

FUW0018I **Log File processing has ended; DDname=*ddname*, Time=*yyyy-mm-dd hh.mm.ss.thmiju*, Records read=*count*, LSN=*lsn***

Explanation

The specified log file data set has been closed. Either the reporting stop time or end of file has been reached. The log record suffix (STCK time stamp and LSN) are formatted to help identify the end of this Log file. The number of records that were read from the file is not indicative of the number of records that were reported; see message FUW0007I for the records reported count.

System action

Processing continues.

User response

None required.

FUW0019E **GETDSAB macro error; Ret=*xx*, Reas=*xx***

Explanation

Transaction Analysis Workbench received a bad return code from the GETDSAB macro.

System action

Transaction Analysis Workbench processing terminates.

User response

Transaction Analysis Workbench error. Contact IBM Software Support.

FUW0020E **Field name is unknown; Code=*code*; Name=*name***

Explanation:
The specified field name is not known for this log record code.

System action:
Processing stops.

User response:
Correct the field name and retry your request. If you are using the Transaction Analysis Workbench ISPF dialog to build a filter, press the Prompt function key (F4) to select from the list of allowed field names.

FUW0021E **Log Record Code is not supported; Code=*code***

Explanation

The specified Log Record Code is not supported by Transaction Analysis Workbench. Transaction Analysis Workbench supports most IMS system record types, but does not support user defined log records. If the Log Record Code is unknown to Transaction Analysis Workbench, requests that require field level interpretation cannot be honored:

- The record cannot be formatted, only dumped.
- Filter Conditions cannot specify field names, only offsets.
- A Form cannot be defined.

Some log record types require a subcode. For example, the Fast Path log record types have a code of 59 and subcode to further identify them. In this case, specifying 59 is insufficient and the subcode (for example, 5937) must be specified for Transaction Analysis Workbench to accept it.

System action

Requests for Field level information cause Transaction Analysis Workbench to immediately stop processing. Other requests may continue. For example, Record formatting requests cause Transaction Analysis Workbench to display the record in Dump format only.

User response

Ensure that you have specified the required log record code. If you are using the Transaction Analysis Workbench dialog, tab to a Log Code field (for example, on **Filter** panel), and then press the Prompt

function key (F4) to select from the list of supported Log Codes.

FUW0022E Offset is invalid; Offset=*xx*

Explanation

The specified record or field offset is invalid.

Log record offsets must be either:

- A decimal number in the range 1 to 32760. The offset is taken from the first byte of the log record and is relative to 1. For example, 5 represents the fifth byte of the record.
- A hexadecimal number in the range X'00' to X'7FF7'. The offset is taken from the start of the log record and is relative to 0. For example, X'04' or X'0004' represents the fifth byte of the record. The hexadecimal number represents the same offset as shown in assembler output for the log record macro.

Field offsets indicate an offset past the beginning of the specified log record field. The offset must be a decimal number in the range 1 to 32757 where (1) indicates the first byte of the field, and so on. For example:

```
TRANCODE(5) EQ 'IN'
```

will *match* PARTINQY but will *not match* PINQUIRY or ORDERPIN.

System action

Processing stops.

User response

Specify a valid offset and retry the request.

FUW0023E Comparison Operator is invalid; Operator=*xx*

Explanation

The specified Comparison Operator is invalid. Allowed operators are:

EQ
Equal to

NE
Not equal to

GT
Greater than

LT
Less than

GE
Greater than or equal to

LE
Less than or equal to

For Flag bit checking, allowed operators are:

ON
Bits are On

OFF
Bits are Off

System action

Transaction Analysis Workbench immediately stops processing.

User response

Specify a valid Comparison Operator and retry your request.

FUW0024E Character value end quote is missing; Value=*value*

Explanation

The specified character value requires an end quote to identify the end of the string. This can be caused by truncation when the length of the specified value (including the enclosing quotes) exceeds the allowed maximum length.

System action

Processing stops.

User response

Specify a quote at the end of the value ensuring that the maximum length of the value is not exceeded, and retry your request.

FUW0025E Hexadecimal value is invalid; Value=*xx*

Explanation

The specified hexadecimal value is invalid. A hexadecimal string is specified as X'nn...nn' where nn is a hexadecimal number from 00 to FF.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Specify a correct hexadecimal value and retry your request.

FUW0026E **Numerical value is invalid; Value=*xx***

Explanation

The specified numerical value is invalid. Numbers must be in the range 0 to 99999999.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Specify a correct numerical value and retry your request. If the intention is for this value to be a character string that starts with a digit, then use quotes.

FUW0027E **Time stamp has invalid format (YYYY-MM-DD-hh.mm.ss.thmiju); TIME=*date-time***

Explanation

The specified time stamp is invalid. The time stamp format is YYYY-MM-DD-hh.mm.ss.thmiju. Date must be a valid calendar date and within the range allowed by your system. Time is optional and defaults to 00.00.00.000000. Time can also be abbreviated. For example, specify 2008-06-24-16.47 for 04.47 PM on June 24, 2008.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the time stamp specification and retry your request.

FUW0028E **Log Record Code is invalid; Code=*code***

Explanation

The specified Log Record Code is invalid. The code must be a one (code) or two (code and subcode) byte hexadecimal number. For example, 01 or 5937.

The log record code identifies the log record type. For IMS log records, the record type is always located at the start of the log record (after the 4-byte record descriptor word).

System action

Processing stops.

User response

Correct the Code specification and retry your request.

FUW0029E **Log Sequence Number is invalid; LSN=*lsn***

Explanation

The specified Log Sequence Number is invalid. The LSN is an 8-byte hexadecimal number. For example, 000000000D451C28.

The Log Sequence Number is located at the end of every log record, and between cold starts of IMS, uniquely identifies the record.

System action

Processing stops.

User response

Correct the LSN specification and retry your request.

FUW0030E **VSAM RBA/OSAM RBN is invalid; RBA=*xx***

Explanation

The specified RBA is invalid. The RBA is a 4-byte hexadecimal number representing a VSAM RBA or OSAM RBN.

System action

Processing stops.

User response

Correct the RBA specification and retry your request.

FUW0031E **PST (Region) ID is invalid; PST=*xx***

Explanation

The specified PST (Region) ID is invalid. The PST is a 2-byte hexadecimal number representing the Region PST ID.

System action

Processing stops.

User response

Correct the PST specification and retry your request.

FUW0032E **IMS Release not supported; IMSREL=*xxx***

Explanation

The specified IMS Release is not supported.

System action

Processing stops.

User response

Correct the IMS Release specification and retry your request.

FUW0033E **Originating Tracking UOW ID is invalid; ORGUOWID=*xx***

Explanation

The specified Originating Tracking Unit of Work ID is invalid. The format is *Originating IMS ID*/*time stamp token*. For example IMSA/ B7ADFE54E81F3680.

In a sysplex where originating IMS ID is not known, specify */*time stamp token*.

System action

Processing stops.

User response

Correct the Tracking UOW ID and retry your request.

FUW0034E **Recovery Token is invalid; RECTOKEN=*xx***

Explanation

The specified Recovery Token is invalid. The format is *IMS ID*/*OASN*+*Commit Number*. For example, IMSA/ 0000002300000001.

System action

Processing stops.

User response

Correct the Recovery Token specification and retry your request.

FUW0036E **Value not allowed; Field=*field***

Explanation

You cannot set a value for the specified Field name because it is a Flag bit setting. For example, consider the following flag definition in the type 01 log record macro QLOGMSGP.

```
MSGFLAGS DS     X          MESSAGE FLAGS
MSGFFRST EQU   X'80'    FIRST RECORD OF MSG
```

To test for MSGFFRST, you can specify the following conditions:

```
COND MSGFLAGS ON X'80'
or
COND MSGFFRST ON
```

MSGFFRST cannot specify a value because its value is pre-determined by its definition, in this case X'80'.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Remove the Field value and retry your request.

FUW0037E **Object List name is invalid; Name=*name***

Explanation

The specified Object List name is invalid. The Object List name must be a valid 1 to 8 character member name.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the Object List name and retry your request.

FUW0038E **PORT number is invalid; PORT=*portnumber***

Explanation

The specified Port number is invalid. It must be a numeric value less than 65536.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the Port number and retry your request.

FUW0039E **Connect Logon Token is invalid; LOGTOKEN=*logtoken***

Explanation

The specified Connect Logon Token is invalid. The format is a time stamp token. For example, B7ADFE54E81F3680.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the Connect Logon Token and retry your request.

FUW0040E **Logical Unit of Work ID is invalid; LUWID=*luwid***

Explanation

The specified logical unit of work ID is invalid. The format is: *Network ID*/*LU name*/*Unique value*/*Commit count*. For example, NET1/DB2LUA/B7ADFE54E81F/0001.

System action

Processing stops.

User response

Correct the logical unit of work ID and retry your request.

FUW0041E **Input DD is missing (LOGIN or SMFIN)**

Explanation

No input files were specified in the JCL.

System action

Processing stops.

User response

Specify the input DD in the JCL then retry request.

Related reference

Report and extract utility JCL

You can use the Transaction Analysis Workbench ISPF dialog to create JCL for the utility, or you can write the JCL yourself.

FUW0042I ***request type* processing has ended, record count=*count* DD=*ddname* DSN=*dsname* PATH=*z/OS UNIX file path***

Explanation

The extract, CSV, JSON, or MWP request has completed processing to the specified output data set or z/OS UNIX file path. The record count indicates the number of records written to the extract file.

If the record count is zero, the message severity is escalated to a warning, indicating that a problem might have occurred. In this case, an error message will be issued describing the problem.

System action:
Processing continues.

User response:
None when the message is informational. In the event of an error, see the accompanying error message to resolve the problem.

FUW0043E **Extract did not proceed; REAS=ABENDS*xxx* DSN=*dsname***

Explanation

Extract processing either failed or ended prematurely due to an ABEND or system service failure. The request might have failed during one of the following periods:

1. Prior to commencement of I/O, at data set open time. For example, ABENDS913 - 38: RACF security violation.
2. During I/O processing after some, but not all of the required records were written. For example, ABENDS*x*37 - 04: data set is too small.

System action:
Extract processing stops.

User response

Correct the problem and retry your request. Common ABEND resolutions include:

S913-*xx*
Request authorization from your security administrator, or change the extract data set name to one you have permission to use.

S*x*37
You might have deliberately limited the size of the extract data set to prevent a runaway extract request, in which case you can ignore this

message. If you want to collect more records, reallocate the extract data set with a larger space allocation.

FUW0043W **Extract was interrupted; REAS=ABENDS*xxx* after *nnnn* records DSN=MY.EXTRACT**

Explanation

Extract processing either failed or ended prematurely due to an ABEND or system service failure. The request might have failed during one of the following periods:

1. Prior to commencement of I/O, at data set open time. For example, ABENDS913 - 38: RACF security violation.
2. During I/O processing after some, but not all of the required records were written. For example, ABENDS*x*37 - 04: data set is too small.

System action:
Extract processing stops.

User response

Correct the problem and retry your request. Common ABEND resolutions include:

S913-*xx*
Request authorization from your security administrator, or change the extract data set name to one you have permission to use.

S*x*37
You might have deliberately limited the size of the extract data set to prevent a runaway extract request, in which case you can ignore this message. If you want to collect more records, reallocate the extract data set with a larger space allocation.

FUW0044E **Record processing has failed; ABENDS*xxx*- *cc* - see SYSOUT file SYS*nnnnn* for diagnostics**

Explanation:
An ABEND has occurred while processing a log record. This can happen when a record with an unsupported format is being interpreted for display.

System action:
Diagnostics are written to the specified SYSOUT file under your TSO user id. Processing continues, although the offending log record may be formatted incorrectly.

User response:
Ensure that the type of log file you are processing is a supported type, for example an IMS log or SMF file. If the file is an IMS log, ensure that you have specified the correct release of IMS, for example 121 for IMS version 12. If the problem persists then report the problem to IBM, supplying the diagnostics in the SYSOUT file.

FUW0050E **LOAD failed; Module=*module* ABEND=*xxxxxxxx-xxxxxxxx,* Reason=*reason*[, Type=*log type*]**

Explanation

The specified module could not be loaded into storage for any of the following reasons:

- Missing Transaction Analysis Workbench load library module.
- Missing knowledge module for the type of log file.

System action:
Processing stops.

User response

If the ABEND code is S806-04 and the module name has a prefix of FUW, then verify that the correct STEPLIB or Transaction Analysis Workbench load library has been specified.

If the ABEND code is S806-04 and the module name has a prefix of DFS, then verify that the correct IMS RESLIB library has been specified.

If the ABEND code is not S806-04, then an environmental error caused the LOAD SVC to fail. Look for associated system error messages or check the system code for the failure reason. You might need to increase the REGION size of your job.

If you cannot correct the problem, contact IBM Software Support.

FUW0051E **Log Record formatting failed; Code=*code* LSN=*lsn* KM=*km* Reason=*reason* Field=*field***

Explanation

The log record with the specified Log Sequence Number could not be formatted. The specified Knowledge Module (KM) could not interpret the fields in the record for any of the following reasons:

01
The named field is beyond the end of the current record

02
The named field is before the start of the current record

03
KM has undefined Field type (Extraction)

04
Log record has more than 4096 fields

05
KM has undefined Field type (Formatting)

06
KM has undefined Field type (Zoom)

This error can occur when the format of the log record changes due to IMS maintenance.

System action

Transaction Analysis Workbench does not format this log record. The record is reported in dump format only.

User response

Log record formats can change from IMS release to release, so ensure that you have specified the correct IMS release for this log file.

If you cannot resolve the problem, contact IBM Software Support.

FUW0052W **Form cannot be used to format record; Record Code=*code* Form=*form* Field=*field***

Explanation

The specified Form cannot be used to format the log record for either of the following reasons:

01
The Form is not for this log record code

02
The log record Knowledge Module is not compatible with the Form.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the Form specification and retry your request.

FUW0053E **Unrecoverable CICS CMF error; REAS=*reason***

Explanation

A problem occurred during the preparation of a CICS monitoring performance class record (SMF 110). Reason codes:

01
The SMF record specified a Connector that is not defined in the default dictionary. The record is mapped by the dictionary record contained in the `DICT=` module.

02
The CSRCESRV macro used to expand a compressed CMF failed with the specified return code.

System action:
Log file processing stops.

User response:
This problem can only be caused by an invalid CMF record or an internal logic error. Contact IBM Software Support.

FUW0054E **Extract file is missing from the JCL; DDname=*ddname*, Request=*type***

Explanation:
The extract file for the `REPORT CICS-DBCTL` or `REPORT IMS-DBCTL` request was not specified in the JCL. Exception transaction records could not be written.

System action:
The report request continues without writing exceptions to the extract file.

User response:
If you need to collect the exceptions for further analysis, then specify the extract file in the JCL.

FUW0055W **CICS/IMS-DBCTL request parameter is invalid; PARM(*invalid parameter***

Explanation:
An incorrect request parameter was specified for the `REPORT CICS-DBCTL` or `REPORT IMS-DBCTL` command. Exception transaction records could not be written.

System action:
The report request continues using the valid request parameters.

User response:
Correct the `PARM(...)` specification, then re-run the job.

Related reference
REPORT command
Requests a report.

FUW0056I **Extract processing has ended; DDname=*ddname*, TOTAL=*transactions*, EXCEPT=*exceptions*, CUT=*written***

Explanation

`REPORT CICS-DBCTL` or `REPORT IMS-DBCTL` exception extract processing has completed.

TOTAL
Total number of transactions.

EXCEPT
Total number of exception transactions that failed at least one of the criteria.

CUT
Total number of transaction records cut to the extract file. If exception criteria were specified, then only exception transactions are cut. If exception criteria were not specified, then all transactions are cut.

System action:
Processing continues.

User response:
None required.

FUW0057W CMF records (in SMFIN) are not in ascending start time sequence

Explanation:
Combined CICS-DBCTL reporting detected that the CMF performance class records in the SMF file are not in ascending (transaction) start time sequence. The correlation of CMF records (in SMFIN) with the IMS transaction index records (in LOGIN) might not always occur. The resulting report might be incomplete.

System action:
Processing continues.

User response
Ensure that CMF input into the combined CICS-DBCTL report has been prepared correctly:

1. Use the CICS-DBCTL extract process to create an extract of exception transactions from the original SMF file.
2. Sort the extract file into start time sequence.
3. Use the extract file as (SMFIN) input into the combined CICS-DBCTL report.

FUW0058W IMS transaction index records (in LOGIN) are not in start time sequence

Explanation:
Combined CICS-DBCTL reporting detected that the IMS transaction index records in the log input file are not in ascending (transaction) start time sequence. The correlation of CMF records (in SMFIN) with the IMS transaction index records (in LOGIN) might not always occur. The resulting report might be incomplete.

System action:
Processing continues.

User response
Ensure that IMS input into the combined CICS-DBCTL report has been prepared correctly:

1. Use the IMS-DBCTL extract process to create an extract of exception transactions from the original IMS log file.
2. Sort the extract file into start time sequence.
3. Use the extract file as (LOGIN) input into the combined CICS-DBCTL report.

FUW0059I CICS-DBCTL reporting has ended; DDname=*ddname*, CMF=*count*, IMS=*count*, Match=*count*

Explanation
Combined CICS-DBCTL reporting has ended successfully.

CMF
Total number of CICS tasks that were reported.

IMS
Total number of IMS transaction index records that were processed. Note that only matched IMS records are reported. You can use the IMS-DBCTL request to report all the IMS DBCTL transactions.

Match
Total number of IMS transaction index records that were matched to the corresponding CMF record (and hence reported).

System action:
Processing continues.

User response:
None required.

FUW0060W SMF 42.6 request parameter is invalid; PARM(*x*)

Explanation
An incorrect request parameter PARM(...) was specified for the SMF 42.6 DASD Data Set I/O report. Supported parameters are: Exception criteria:

IORATE>n
The data set is an exception when the I/O count per second exceeds the specified threshold value. The IORATE in itself may not indicate a problem, but in conjunction with RESPONSE>n, allows you to focus on data sets with high I/O activity that had substandard response time.

RESPONSE>ss.thmiju
The data set is an exception when the average I/O operation response time exceeds the specified threshold value, which may be a value from 0.000001 to 999999 seconds inclusive. For example RESPONSE>0.5 is an exception when the average data set I/O time exceeds 0.5 seconds.

Selection criteria:

JBN=/DSN=
JBN (job name) and DSN (data set name) are selection criteria that restrict reporting to the required job names and data sets. Multiple names can be specified (spanned across multiple lines when necessary). Wildcard characters (*) are allowed.

System action:
The report request continues using the valid request parameters.

User response:
Correct the PARM(...) specification, then re-run the job.

FUW0061I **DB2 exception recap; No exceptions were detected**

FUW0062I **DB2 exception recap; Exception=*exception* Count=*nnnn***

Explanation

The DB2 accounting exception report and extract issues one of more messages at the completion of processing. Either:

1. One message is issued to indicate that no exceptions were detected, or
2. One or more messages are issued for each exception that was triggered.

System action:
Processing continues.

User response:
The DB2 exception list report provides the detail about each exception. The DB2 exception extract contains the DB2 accounting records (SMF 101) that generated an exception. You can process this extract using the ISPF dialog to obtain additional detail.

FUW0063W **DB2 exception request parameter is invalid; PARM(x)**

Explanation:
An incorrect request option was specified in PARM(...) for the DB2 exception report and extract.

System action:
The report request continues using the valid request parameters.

User response:
Refer to the User's Guide for the PARM options supported by REPORT DB2X. Correct the PARM(...) specification, then re-run the job.

FUW0090E **Virtual Storage obtain request failed, RC=*xx***

Explanation

Transaction Analysis Workbench could not obtain Virtual Storage required for Log file processing. The STORAGE OBTAIN return code identifies the failure reason.

System action

Log file processing stops.

User response

Ensure that the TSO region size is sufficient to run Transaction Analysis Workbench and other ISPF applications in parallel. About 10 megabytes is required for large log files, to keep IO buffers and filtering and tracking information.

Transaction Analysis Workbench must share the region with other ISPF applications running in split screen mode. Closing other sessions may alleviate the shortage of virtual storage. Otherwise increase the REGION size in your logon procedure or at logon time.

FUW0095I ***Type Name Action***

Explanation:
Export/Import information message giving the type of data processed, the name, and the action performed.

System action:
Processing continues.

User response:
Check that the required action was completed.

FUW0096E **AMATERSE has returned an error**

Explanation

Export/Import has encountered an error response back from AMATERSE.

System action

Processing terminates.

User response

Check the messages in AMAPRINT to determine the error and required action.

FUW0100E **Log record processing failed, RC=*xx***

Explanation

Transaction Analysis Workbench encountered a problem reading a log record from the log file data set.

System action

Log file processing stops.

User response

Verify that the log File has valid DCB attributes, specifically RECFM=VB.

If the DCB attributes are correct, then browse the data set using ISPF Browse. Scroll to the bottom of data to verify that all records are read successfully. Otherwise, contact IBM Software Support.

FUW0101E **PDS Member does not exist; Name=*name* BLDL RC=*xxxx-xxxx***

Explanation

The Log input file is a PDS but the specified member name does not exist. The BLDL return and reason codes indicate the failure reason.

System action

Log file processing stops.

User response

None required.

FUW0102E **Log input file is not RECFM=VB; DCBRECFM=*xx***

Explanation:
The Log input file does not have variable length records, or the dataset has spanned records. Only Log files with RECFM=VB are supported. DCBRECFM is the unsupported RECFM from the DCB.

System action:
Log file processing stops.

User response:
Ensure that the specified Log input file is a valid IMS Log data set with RECFM=VB.

FUW0103E **Log file processing failed; Reason=*reason text***

Explanation

The log file, or files, could not be processed. Common problems include:

1. The data set is empty
2. The data set is not a supported type of log file

System action:
Log file processing stops.

User response:
Ensure that the specified log input file is a type supported by Transaction Analysis Workbench and contains data.

FUW0104W **Attention Interrupt has stopped Log File processing**

Explanation

Transaction Analysis Workbench has stopped reading the Log Input file because an Attention Interrupt was received.

System action

Transaction Analysis Workbench stops reading the Log file and displays only data read to this point.

User response

Press Enter to resume Log Input file processing.

FUW0105E **Log input file is not available. DDname *ddname* allocation error; RDJFCB RC=*xx***

Explanation

The RDJFCB system service determined that the Log input file is not allocated to the specified DDname.

System action

Log file processing stops.

User response

Verify that the Log file data set name is specified correctly. The data set must reside on an online DASD volume. If the data set is cataloged, it must reside on the cataloged VOLSER. If the data set is not cataloged, it must reside on the specified VOLSER.

FUW0106E **Log input file does not reside on the specified volume; VOLSER=*volser*; OBTAIN RC=*xx***

Explanation

The DADSM OBTAIN system service determined that the Log input file does not reside on the required volume, as indicated in the Catalog or the specified VOLSER.

System action

Log file processing stops.

User response

Verify that the Log file data set name is specified correctly. The data set must reside on an online DASD volume. If the data set is cataloged, it must reside on the cataloged VOLSER. If the data set is not cataloged, it must reside on the specified VOLSER.

FUW0107E Zone specification is invalid; ZONE=*zone*

Explanation

The specified time zone offset is invalid. Allowed values are LOCAL, GMT, +hhmm and -hhmm.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the ZONE specification and retry your request.

FUW0108E Log File Open request failed; ABEND=*xxxxxxxx-xxxxxxxx*

Explanation

The requested IMS log file could not be opened because the OPEN request failed. ABEND Code specifies the reason for the failure. The most common reason is that access was denied due to an authorization failure, which returns ABEND=ABENDS913.

System action

Transaction Analysis Workbench processing stops.

User response

Check the OPEN SVC messages for the failure reason, correct the problem, and retry your request.

FUW0109E Log input file is not DSORG=PS; DS1DSORG=*xxxx*

Explanation

The Log input file does not have a Data Set Organization (DSORG) of PS. DS1DSORG is the unsupported DSORG from the DSCB.

System action

Log file processing stops.

User response

Ensure that the specified Log input file is a valid IMS Log data set with DSORG=PS.

FUW0110E SFUWLINK Load Libraries are mismatched; Starting=V1R1M0-APAR001, Profile=V1R1M0-APAR001. Go to option 0.1 Settings.

Explanation:
Your ISPF dialog session was started by a program that was loaded from an SFUWLINK load library at the Starting level of maintenance. But this library does not match your profile setting. Workbench requires that your Starting and Profile SFUWLINK libraries are the same data set. This ensures that the dialog uses only modules loaded from the same library, to avoid problems caused by incompatibility.

System action:
Log file processing stops.

User response:
Correct the Workbench Load Library setting to specify the most recent Load Library data set name. The Workbench Load Library is specified in your Workbench Settings (Dialog option 0.1).

FUW0111E Unrecoverable logical IO error, Reason=*x*

Explanation

I/O processing for the log files failed due to an internal processing error. Transaction Analysis Workbench keeps an in-storage index of the data it reads. Problems occur when data that is re-read from DASD does not match the index. The most common cause is when the contents of the log file has changed, possibly an OLDS that has been recycled.

System action

Transaction Analysis Workbench processing stops.

User response

Retry your request. If I/O errors persist, contact IBM Software Support.

FUW0112E CSI processing error: R15=*nn* MODID=*nn* RETC=*nn* REAS=*nn* DSN=*dsn*

Explanation

CSI catalog look-up services failed while checking the VSAM attributes of the OMEGAMON for IMS ATF journal.

System action

ATF journal processing stops.

User response

Internal error. Contact IBM Software Support.

FUW0113E Logstream connection|processing error: *reason*

Explanation

Transaction Analysis Workbench uses the System Logger Services to access log streams, but a problem was encountered:

Connection failed
Common reasons include:

080B
Not defined in LOGR policy.

Verify that you have specified the correct log stream name. Use the `D LOGGER,C` MVS system command to verify the log stream is defined and connected to the system.

080D
Authorization for read access denied.

SAF authorization checking has failed, RACF has denied access. AUTH=READ authority to the logstream is required.

0831
Invalid name.

Log stream names use normal data set naming conventions with a maximum length of 26. For example:

```
SYSPLEX.OPERLOG
IFASMF.PRODPLEX.SMF
```

08E2
DASDONLY connected to another system.

DASDONLY log streams can only be connected to one system in the sysplex at a time. Only Coupling Facility log streams can be connected to more than one system.

I/O processing failed
Common reasons include:

0808
Severe log data set I/O error.

0846
Log stream is empty.

For all other errors, check the IXGCONN or IXGBRWSE macro return and reason codes in the *MVS Programming: Assembler Services Reference, Volume 2*.

Blocks in the log stream were not in the expected format

OPERLOG
MDB format

SMF
SMFP format

The unsupported log stream block data is contained in the message.

System action

Log stream processing stops.

User response

- For connection or processing errors, check the reason text or code. For more information about the reason text or code, see the previous explanation. Ensure that you have:
  1. Specified the correct log stream name, and that the log stream is connected to the system
  2. RACF authority to read the log stream
- For OPERLOG or SMF processing errors, check that the log stream is used for that purpose.

FUW0114E VSAM ERROR, DDNAME=*ddname* MACRO=*macro* RETC=*nnnn* REAS=*reas* CALLID=*callid*

Explanation

VSAM processing for an OMEGAMON for IMS ATF journal has failed.

System action

ATF journal processing stops.

User response

Check the VSAM return and reason codes in the *DFSMS Macro Instructions for Data Sets*. For an OPEN request, REAS=0098 indicates authorization checking failed, RACF denied access. Otherwise, contact IBM Software Support.

FUW0115E Time slicing is not allowed for Partitioned Data Sets

Explanation

The request to time slice the selected log file failed because the data set is a PDS. PDS members cannot be time sliced. The I/O sampling technique used to

determine the file time range cannot detect end of the member, and therefore can incorrectly process other members in the PDS.

System action

Transaction Analysis Workbench processing stops.

User response

Process the log file without using time slicing.

FUW0116E Maximum record number exceeded

Explanation:
The number of records in the result set reached the maximum 99,999,999 records before completion.

System action:
Transaction Analysis Workbench processing stops.

User response:
Create a subset of the file using extract facilities or use time-slicing.

FUW0120E IMS Transaction Index record creation has failed; RC=*xx*

Explanation

The IMSINDEX request to create IMS transaction index (CA01) records from the IMS log has failed. This problem can occur when:

1. When the version of the IMS log does not match the specified version in the EXEC PARM=Vvvr.
2. Virtual Storage was exhausted.

System action:
Processing stops.

User response

1. Ensure that you have specified the correct version of IMS; for example specify EXEC PARM=V131 for IMS version 13.
2. Increase you region size; for example REGION=100M. Then retry your request.

FUW0121I IMS Transaction Index records were created; Count=*n*

Explanation:
The IMSINDEX request successfully created the reported number of IMS transaction index (CA01) records.

System action:
Processing continues.

User response:
None.

FUW0200E Command is invalid

Explanation

The specified input command is invalid.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input then resubmit the job.

Related reference

Types of report and extract utility commands
The JCL to run the report and extract utility must specify a SYSIN data set that contains commands to the utility. There are several types of commands: global, action, qualifying, and administrative. You need to know which types of commands apply to the task you want to perform, and specify those commands in the appropriate order.

FUW0202E CODE specification is invalid

Explanation

The CODE command does not specify a valid combination of log type and code. For details, see "CODE command and COND statement" on page 556.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input then resubmit the job.

FUW0203E Parentheses not paired

Explanation

The command input specified parentheses (brackets) that were not paired.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input then resubmit the job.

FUW0204E Quotes not paired

Explanation

The command input specified a character or hexadecimal string in quotes that were not paired.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input then resubmit the job.

FUW0205E Field name is not specified

Explanation

The Field name in the COND statement is not specified. COND requires a Field name, Comparison operator and Field value to be specified. For example:

```
COND TRANCODE EQ 'INQUIRY'
```

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input then resubmit the job.

FUW0206E Comparison Operator is not specified

Explanation

The Comparison Operator in the COND statement is not specified.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input then resubmit the job.

FUW0207E Field value is not specified

Explanation

The Field value in the COND statement is not specified.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input then resubmit the job.

FUW0208E Command keyword is invalid

Explanation

The requested batch command has an invalid keyword.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

EXTRACT command
Writes the selected log records to an extract data set.

REPORT command
Requests a report.

FUW0209E OUTPUT DDname is invalid

Explanation

The REPORT, EXTRACT or CSV command specified an invalid OUTPUT DDname. The DDname must be 1 to 8 characters long. For example: OUTPUT(LOGRPT)

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0210E FILTER or FORM name is invalid

Explanation

The REPORT, EXTRACT, CSV or REXX command specified an invalid Filter name, or the CODE command specified an invalid FORM name. The name must be 1

to 8 characters long. For example: `FILTER(MYRECS)` or `FORM(MY01)`

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0211E FILTER and FORM not allowed; FUWCDS DD missing

Explanation

FILTER or FORM cannot be specified because the IMS system definitions repository (also known as the control data set; ddname FUWCDS) is not specified in the JCL.

System action

Processing stops.

User response

Either remove the FILTER or FORM command or specify the repository in the JCL. To specify this repository in the dialog, select option 0.2 **Repositories**.

FUW0212E No REPORT, EXTRACT, CSV or REXX requests

Explanation

The SYSIN command input did not specify any REPORT, EXTRACT, CSV or REXX requests. At least one REPORT, EXTRACT, CSV or REXX request must be specified.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0213E DDname already used by another request; DDname=*ddname*

Explanation

The specified OUTPUT DDname for the REPORT or EXTRACT request cannot be used because a previous request has already reserved it. Each REPORT and EXTRACT request must specify a unique OUTPUT DDname.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0214E CODE must be preceded by a REPORT, EXTRACT, CSV or REXX request

Explanation

The CODE command must be preceded by a REPORT, EXTRACT, CSV or REXX request.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0215E CODE command has invalid keyword

Explanation

The CODE command has an invalid keyword. For details, see “CODE command and COND statement” on page 556.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0216E FORM is not allowed for CODE(ALL)

Explanation:
A FORM cannot be specified for CODE(ALL). FORM can only be specified for individual log codes, eg. CODE(01) FORM(MY01).

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0217E COND must be preceded by a CODE command

Explanation

The COND command must be preceded by a CODE command. COND identifies a record filtering condition for the previously specified log record code. Multiple COND statements per CODE can be specified.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input then resubmit the job.

FUW0219E FORM cannot be used for this CODE; Form Code=*code*

Explanation

The specified Form cannot be used for this Log code because it was built using another log code.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input then resubmit the job.

FUW0220E START time is after STOP time

Explanation

The START and STOP time stamps specify an invalid reporting interval because the START time stamp is after the STOP time stamp.

System action

Processing stops.

User response

Correct the time stamp specification and retry your request.

FUW0221E TRACK must be preceded by a REPORT, EXTRACT or REXX request

Explanation:
The TRACK command must be preceded by a REPORT, EXTRACT or REXX request. TRACK specifies that Tracking is to be activated for this REPORT, EXTRACT or REXX request.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0222E CONNECT record code must be in the range A0 to FF

Explanation

The CONNECT command must specify either:

1. a hexadecimal record code in the range A0 to FF ,or
2. no record code, in which case all records in the range A0 to FF are treated as Connect records.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0223E FORMAT option is invalid

Explanation

The specified FORMAT option is not supported. Supported formatting options are: STD, FORM, DUMP, HEX0, and HEX1.

System action

Processing stops.

User response

Correct the command input then resubmit the job.

FUW0224E More than one REXX command or a REXX command and REPORT or EXTRACT not allowed

Explanation

REXX requests and EXTRACT and REPORT requests are mutually exclusive. Also only one REXX request is allowed at a time.

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0225E REXX command has invalid keyword

Explanation

The REXX command has an invalid keyword. Allowed keywords are FILTER, EXEC, and ARG.

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0226E EXEC name is invalid

Explanation

The REXX command specified an invalid exec name. The exec must be 1 - 8 characters long. For example, `EXEC(FUWTRANS)` is a valid exec name.

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0227E ARG name is invalid

Explanation

The REXX command specified an invalid argument. If the argument contains blanks it must be enclosed in quotes. For example:

```
ARG('P1 P2 P3')
```

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0228E Level value must be in the range 1 to 255

Explanation

The specified Level value must be in the range 1 to 255.

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0229E ZONE command has invalid keyword

Explanation

The ZONE command has an invalid keyword. See "ZONE command" on page 508.

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0230E Option is not allowed for LSO

Explanation:
The LSO command has no option to specify. The only allowed keyword is LSO.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0231E ELAPSED command has invalid keyword

Explanation

The ELAPSED command has an invalid keyword. The only allowed keywords are MICRO or NANO

System action

Processing stops.

User response

Correct the command input and then resubmit the job.

FUW0232E IMPORT command with another command

Explanation:
The IMPORT command cannot appear with any other command. Only one IMPORT command is allowed.

System action:
Processing stops.

User response:
Correct the command input and then resubmit the job.

FUW0233E IMPORT command has invalid keyword

Explanation

The IMPORT command has an invalid keyword. The only allowed keywords are REPLACE or NOREPLACE.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0234E EXPORT command with another command

Explanation

The EXPORT command cannot appear with any other command. Only one EXPORT command is allowed.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0235E FILTER, as a separate command, must appear with EXPORT command

Explanation

FILTER, as a separate line command, must appear with an EXPORT command.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0236E FORM, as a separate command, must appear with EXPORT command

Explanation

FORM, as a separate line command, must appear with an EXPORT command.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0237E OBJECTLIST, as a separate command, must appear with EXPORT command

Explanation

OBJECTLIST, as a separate line command, must appear with an EXPORT command.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0238E FORM must be provided for CSV command

Explanation

FORM must be completed for a CSV command. The Form describes the format of the CSV output record.

System action

Transaction Analysis Workbench immediately stops processing.

User response

Correct the command input and then resubmit the job.

FUW0239E Invalid SMF report request

Explanation:
The command syntax is REPORT SMF(*type*), where *type* must be one of the types (or specific subtypes) for which Transaction Analysis Workbench provides a tabular report. For a list of these types, see Chapter 83, “Log types and codes,” on page 635.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0240E Unsupported type of Log Stream

Explanation

The LOGSTREAM command did not specify a supported log stream type. Supported log stream types are SMF, OPERLOG, and CQS. The type must prefix the log stream name, for example SMF:IFASMF.FTS1.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input, then resubmit the job.

FUW0241E Only one Log stream can be specified

Explanation

Only one LOGSTREAM command is allowed. Batch processing can only support a single log stream as input into reporting.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the command input, then resubmit the job.

FUW0242E Log stream input requires START date and time to be specified

Explanation

When Log stream input is requested, start date and time must also be specified. Log streams can potentially cover a long period of time (many months) and old data might have been archived to DASD data sets (that are now migrated). A starting point ensures that only required data from the log stream is accessed and reported.

System action

Transaction Analysis Workbench processing stops.

User response

To prevent accidental reporting of the entire log stream, specify the start date and time.

FUW0243E APPLTRAN must specify (DFHAPPL or 1-999)

Explanation:
The **APPLTRAN** specification is invalid. For reporting purposes, **APPLTRAN** causes the actual CICS transaction ID (in DFHTASK C001) to be substituted with the application transaction ID.

System action:
Transaction Analysis Workbench processing stops.

User response:
Correct the command input, then resubmit the job.

Related reference

REPORT CICS-DBCTL command: CICS reports
Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

FUW0244E TRACE operand is not ON/OFF or 1-4

Explanation

The TRACE command expects an operand of either:

1. ON or OFF
2. In the range 1 to 4 for the required DB2 trace level (records type "DTR")

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0245E REQUEST type is not supported

Explanation

The REQUEST type is not one of the following:

1. REQUEST=SUBMIT submits a batch JOB
2. REQUEST=OUTPUT saves job output to a session.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0246E Only one REQUEST is allowed

Explanation:
More than one request has been issued. Only one REQUEST= is allowed and it is also not allowed with other request types such as REPORT, EXTRACT and CSV.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0247E REQUEST=OUTPUT or SCHEDULE not active

Explanation:
A command was specified that is only supported when REQUEST=OUTPUT or REQUEST=SCHEDULE is active. For REQUEST=OUTPUT: JOBNAME, JOBID, SESSION and TASK are required. For REQUEST=SCHEDULE: SESSION and one or more TASKs are required.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0248E JOBNAME= is missing or invalid

Explanation:
REQUEST=OUTPUT requires the job name to be specified, for example JOBNAME=MYREPORT.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0249E JOBID= is missing or invalid

Explanation:
REQUEST=OUTPUT requires the job number be specified, for example JOBID=JOB03256.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0250E SESSION= is missing or invalid

Explanation:
REQUEST=OUTPUT and REQUEST=SCHEDULE require the session number to be specified, for example SESSION=00000047.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0251E TASK= is missing or invalid

Explanation:
REQUEST=OUTPUT requires the session task token be specified, for example TASK=CABD5E37F2B89A10.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0252E TASK= MAXRC specification is invalid

Explanation:
The TASK= specification for a REQUEST=SCHEDULE request supports one sub-operand MAXRC. MAXRC is the maximum allowed return code for the task. Allowed values are from 0 to 9999.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0253E REQUEST=SCHEDULE has invalid option

Explanation
The only allowed options for REQUEST=SCHEDULE are:

1. SKIPIFDONE: The task is skipped if already done
2. REDOIFDONE: The task is always done
3. TIMEOUT: The task scheduler will stop waiting for a task to complete after the specified number of minutes has expired.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0256I IDCAMS called to process the following command: *command text*

Explanation
The DFDSS IDCAMS utility has been called to process a command. The IDCAMS command and messages follow.

Reasons for invoking IDCAMS include: CSV or JSON request renames the output data set when the **NEWNAME** parameter is specified.

Message IDC0531I indicates that the ALTER NEWNAME request was successful.

System action
Processing continues.

In the event of a failure, the IDCAMS return code will be propagated to the step completion code and the completion message FUW0009I.

User response:
None.

FUW0257I CSV delimiter was changed from comma to semicolon because the CTHM option was requested

Explanation:

The **CSV** command of the report and extract utility has been specified to use a comma as the delimiter between fields, either explicitly via the **DELIMITER** parameter, or by default. However, the **CTHM** parameter has also been specified. **CTHM** causes time stamp fields to use a comma as the separator between seconds and milliseconds (*hh*:*mm*:*ss*,*thm*), which would invalidate the CSV.

System action:
The field delimiter is changed to a semicolon. Processing continues.

User response:
If a semicolon delimiter suits your needs, no action is required. Otherwise, specify an alternative delimiter. For example, `CSV DELIMITER(/)`.

FUW0258E **BRX1WRT to STDOUT failed; RC=*return code* ERRNO=*error number* ERRNO2=*error number 2***

Explanation
An extract, CSV, or JSON request could not write to STDOUT.

The most common reason from this problem is:

```
RC=-1 ERRNO=00000071/EBADF
ERRNO2=0571011C/JRFILENOTOPEN
```

indicating that the batch request is not running under the control of a z/OS UNIX process (batch script) that has already opened STDOUT.

System action:
Processing stops.

User response:
If the request is not running under the control of a z/OS UNIX process, then change the JCL to direct the output to a data set or a z/OS UNIX file.

FUW0259I **Output limit (OUTLIM) reached for DDname=*ddname***

Explanation:
The **CSV** or **JSON** command of the report and extract utility specified an **OUTLIM** parameter to limit the number of output records written to the file. The **OUTLIM** limit has been reached for the reported ddname.

System action:
No further records are written to the CSV or JSON file.

User response:
None.

FUW0260E **STREAM parameter is not supported: *parameter***

Explanation:
A **STREAM** command specified a parameter that is not supported.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0261E **STREAM parameter value is invalid: *parameter***

Explanation
A parameter of the **STREAM** command specified a value that is not allowed. For example:

- The value exceeds the maximum allowed length.
- The **PORT** or **TIMEOUT** parameter value is outside the range 0 - 65535.
- The value is invalid for this parameter; for example, the **SECURITY** parameter specifies an unsupported protocol version.

System action:
Processing stops.

User response:
Correct the command input then resubmit the job.

FUW0262E **Required parameter is not specified for STREAM=*stream_name*; Missing PARM=*parameter***

Explanation:
The **STREAM** command with the `NAME(`*`stream_name`*`)` parameter did not specify the required *parameter*.

System action:
Processing stops.

User response:
Insert the missing parameter then resubmit the job.

FUW0263E **Streaming destination is not specified; STREAM=*stream_name***

Explanation:
A **CSV** or **JSON** command specifies the parameter `STREAM(`*`stream_name`*`)`, but there is no corresponding **STREAM** command with the parameter `NAME(`*`stream_name`*`)`.

System action:
Processing stops.

User response:
Insert the missing **HOST** parameter then resubmit the job.

FUW0269E **Stream request has failed; STREAM=*name*; RC=*xx* from *module***

Explanation

The request to stream JSON or CSV data via TCP/IP has failed. The *module* determines the type of failure:

- **CEEPIPI** is the Language Environment preinitialization interface module that calls the Transaction Analysis Workbench streaming module, FUWIPSTR.

  A return code from CEEPIPI indicates a problem either establishing or invoking Language Environment.

- **FUWIPSTR** is the Transaction Analysis Workbench module that handles TCP/IP communications.

  A return code from FUWIPSTR indicates a failure to connect or send to the TCP/IP host.

  An additional error message is issued to explain the reason for the problem. Common reasons:

  – The TCP/IP host is not available

  – The TCP/IP security settings are preventing connection

System action:
Processing stops.

User response

For return codes from CEEPIPI, contact IBM Software Support.

For return codes from FUWIPSTR, see the additional error messages.

FUW0270E SYSUT1 DD is missing

Explanation:
SYSUT1 DD is not specified in the JCL. SYSUT1 contains the skeletal JCL input for REQUEST=SUBMIT.

System action:
Processing stops.

User response:
Specify SYSUT1 DD in the JCL then retry request.

FUW0271E SYSUT2 DD is missing

Explanation:
SYSUT2 DD is not specified in the JCL.
SYSUT2 specifies the internal reader output for REQUEST=SUBMIT.

System action:
Processing stops.

User response:
Specify //SYSUT2 DD SYSOUT=(A,INTRDR) in the JCL then retry request.

FUW0272I Job submitted successfully; JOBNAME=*jobname* JOBID=*jobid*

Explanation:
The SUBMIT request was processed successfully. The job name and id are recorded for your information.

System action:
Processing continues.

User response:
Note the job name and number (id). You may want to check its progress and output in SDSF. If the job is part of a task in a session, then workbench will save the output to the session once the job has completed and on the JES output queue.

FUW0273E OPEN SVC failed for DDNAME=*ddname*

Explanation:
The specified file failed to open.

System action:
Processing stops.

User response:
Verify that the DD statement is correctly specified in the JCL.

FUW0279E %FILE. error; REAS=*CC reason text*

Explanation

The cause of the failure is the %FILE substitution variable in a DD statement, for example //DDN DD DSN=%FILE.SYSname.SYStype.LOGtype.SOURCE The most common causes are:

1. %FILE is specified incorrectly in the workflow task
2. The session has no log files registered
3. No log files match the %FILE qualification criteria.

System action:
Incomplete JCL is generated.

User response

Depending on the type of error:

1. Correct the %FILE specification in the workflow task.
2. Ensure that previous workflow tasks have completed successfully. These tasks may have been responsible for collecting the required data; either via auto-file-selection or an extract-build process.
3. Define the required log files to the session. Then retry the request.

FUW0280E FUWSAVE DD is missing

Explanation:
FUWSAVE DD is not specified in the JCL. For REQUEST=OUTPUT, the job output will be saved to this new data set.

System action:

Processing stops.

User response:
Specify FUWSAVE DD in the JCL then retry request.

FUW0281E FUWPROBR DD is missing

Explanation:
FUWPROBR DD is not specified in the JCL. For REQUEST=OUTPUT, this session repository will be updated with details about the FUWSAVE data set.

System action:
Processing stops.

User response:
Specify FUWPROBR DD in the JCL then retry request.

FUW0282I Job output saved successfully; JOBNAME=*jobname* JOBID=*jobid* DSN=*dsn*

Explanation:
The output for the job name and id combination was successfully saved in the data set.

System action:
Processing continues.

User response:
None required.

FUW0283E Job was not found on JES queue; JOBNAME=*jobname* JOBID=*jobid*

Explanation:
The specified job name and id was not found. A request to JES2/3 to return the status of the job responded with a JOB NOT FOUND condition.

System action:
Processing stops.

User response:
Check that the specified JOBNAME and JOBID is still on the JES output queue, and has not been deleted. If the job output is no longer available then you will need to re-run the job. Then retry this request (with the new JOBNAME and JOBID).

FUW0284E JES processing error: *reason*

Explanation:
A request to the JES subsystem to process a job has failed. The reason describes the system service and error codes associated with the problem.

System action:
Processing stops.

User response

Check the JES service error codes to determine the reason for the problem. Common problems include:

- For Function=SSST, see SYS1.MACLIB(IAZSSST) for errors returned in STATREAS. Common errors are: =08 Invalid job id - not in "JOB01234" format and =2C Invalid job name
- For Function=SSS2, see SYS1.MACLIB(IAZSSS2) for errors returned in SSS2REAS. Common errors are: =F0 Job has no eligible output; may still be executing

FUW0285I JOBNAME=*jobname* JOBID=*jobid* COMP=*comp* STATUS=*status*

Explanation:
The information about the job being processed was returned from the JES subsystem.

System action:
Processing continues.

User response:
Informational only, no action required.

FUW0286E FUWSAVE DD data set is not a PDS

Explanation:
The //FUWSAVE DD DSN= data set is not a PDS. Each SYSOUT file is saved in a PDS member.

System action:
Processing stops.

User response:
Correct the SPACE= allocation to specify directory blocks, for example SPACE=(CYL,(10,10,5),RLSE).

FUW0287E Session is not registered; ID=*id* Repository=*repository*

Explanation:
The requested session is not registered in the session repository.

System action:
Processing stops.

User response:
From the ISPF dialog, select 1 "Sessions" to view the list of registered Sessions. Ensure the SESSION= input command refers to the required session, then retry request.

FUW0288E Task is not known to Session; ID=*id* TASK=*task* Repository=*repository*

Explanation:
The task identified by the 16 byte token (TASK=) is not known to the Session (SESSION=).

System action:
Processing stops.

User response:
From the ISPF dialog, go to the associated session task to determine the correct task token. Correct the token in the JCL then resubmit the job.

FUW0289I **Job output attached to Task; ID=*id* TASK=*task* DSN=*dsn***

Explanation:
The job output save data set has been successfully attached to its associated session task. The job output is now available to view in the session.

System action:
Processing continues.

User response:
Informational only, no action required.

FUW0290E **SUBMIT error: *reason text***

Explanation
The task scheduler failed to submit the task JCL. The reason is provided, and can be the following:

1. A system service failure
2. No job cards were detected in the JCL.

System action:
Processing stops.

User response:
If the error was caused by a system service failure then check for an environmental issue such as security access. If the error was caused by no job cards detected then correct your task JCL to ensure it contains a job card or %JOBCARD statement.

FUW0291I **Task is about to be scheduled; TASK=*task***

Explanation:
The workflow scheduler is about to start processing the specified task.

System action:
Processing continues.

User response:
Informational only, no action required.

FUW0292E **Task scheduling has stopped due to a bad completion code, expected COMP=*CC 1234***

Explanation:
A workflow task has caused scheduling to stop because of a bad completion code.

System action:
Processing stops.

User response:
Review the failed job's output to determine the cause of the problem.

FUW0293E **Task scheduling failed to reach completion**

Explanation:
The scheduling of tasks was terminated due to a problem with the last task that was processed.

System action:
Processing stops.

User response:
Refer to the error message for the task that failed to take corrective action.

FUW0294I **Task scheduling has completed successfully**

Explanation:
The scheduling of all tasks completed successfully.

System action:
Processing continues.

User response:
None required. The output from the tasks may be available from the session workflow.

FUW0295I **Task is skipped because it is already done; TASK=*task***

Explanation:
The workflow scheduler is skipping this task because it has already been done.

System action:
Processing continues with the next task.

User response
If the task has already completed satisfactorily then no action is required. If you want to run this task again then either:

1. Change its status back to NOT DONE
2. Specify the REDOIFDONE option

FUW0296I **Task is skipped because its STATUS is IGNORE; TASK=*task***

Explanation:
The workflow scheduler is skipping this task because its status is set to IGNORE - the task is to be ignored and not run.

System action:
Processing continues with the next task.

User response:
Informational only, no action required.

FUW0297W **Task scheduling has stopped due to a time out waiting for a job to complete**

Explanation:
The previous FUW0285I message identifies the job that did not complete within the required TIMEOUT limit.

System action:

Processing stops, although the task job will continue to process in the background.

User response:
Wait for the task job to complete, then if necessary, resubmit the schedule request to run the subsequent tasks that were not scheduled.

FUW0298I **Task is an instructional note, see printed output in DD *ddname***

Explanation:
The task does not contain JCL and is not submitted. The task contents are printed to the SYSOUT DD name.

System action:
Processing continues.

User response:
Read the note to see if there are any special instructions when executing this workflow.

FUW0299I **SUBMITTED=*123* OK=*123* FAILED=*123* UNKNOWN=*123* SKIPPED=*123* MISSED=*123***

Explanation

For REQUEST=SCHEDULE a recap of task activity:

SUBMITTED
Jobs that were submitted

OK
Jobs that completed successfully

FAILED
Jobs that failed

UNKNOWN
Jobs with an unknown completion status (no longer on the JES queue)

SKIPPED
Tasks that were skipped

MISSED
Remaining tasks that were not processed.

System action:
Processing continues.

User response:
Informational only, no action required.

FUW0300E **Invalid report parameter, Report=MQ, PARM=*value***

Explanation

The PARM parameter of the REPORT command does not contain any valid report names.

System action

Transaction Analysis Workbench processing stops.

User response

Correct the PARM value, or accept the default report by removing the PARM parameter, then resubmit the job.

Related reference
SMF type 116-1: IBM MQ Accounting Class 3 reports
The IBM MQ Accounting Class 3 reports use SMF type 116, subtype 1 and subtype 2 records to show thread-level and queue-level accounting information for IBM MQ transactions.

FUW0400I **GTF processing could not reconstruct some records; CICS=*cicsnumber* DB2=*db2number***

Explanation:
Some records in the GTF data set could not be completely or correctly reconstructed. The number of incomplete CICS and GTF records is recorded.

System action:
The incomplete records were processed, although some may have caused flow on errors to occur.

User response:
If no other record processing or reporting errors occurred then you can ignore this message. Otherwise there may be conditions in this GTF data set that are not supported, contact IBM Software Support.

FUW0500I **Input file selection processing completed, RC=*xx***

Explanation

Transaction Analysis Workbench has completed automated file selection processing, and issues the following return codes:

00
Processing completed successfully

04
Processing completed, warning message issued

08
Processing failed, error message issued

16
Operand specification error

System action

If the return code is 0 or 4, automated file selection completed successfully, and the Transaction Analysis Workbench report JCL has been submitted. If the return code is 8 or 16, automated file selection has failed. Previous error messages explain the reason for the error.

User response

None required. If the return code is 8 or 16, respond to the previous message.

FUW0501E FROM operand not specified

Explanation

The automated file selection utility FROM time operand was not specified, but is required.

System action

Automated file selection processing terminates.

User response

Ensure the FROM operand is specified in the command input, then resubmit the job.

FUW0502E TO operand not specified

Explanation

The automated file selection utility TO time operand was not specified, but is required.

System action

Automated file selection processing terminates.

User response

Ensure the TO operand is specified in the command input, then resubmit the job.

FUW0503E Duplicate *system type* subsystem ID specified, *ID operand=ID value*

Explanation

The automated file selection utility detected an IMSID or DB2ID operand with a duplicate subsystem ID. A subsystem ID can only be specified once in the input.

System action

Automated file selection processing terminates.

User response

Correct the duplicated subsystem ID specification, then resubmit the job.

FUW0504E Operand specified at column *xx* is invalid, *"yyyyyyyy"*

Explanation

The automated file selection utility encountered an invalid operand starting at column *xx. yyyyyyyy* is the first 8 characters of the invalid operand.

System action

Automated file selection processing terminates.

User response

Remove or correct the invalid operand, then resubmit the job.

FUW0505E *ID operand* operand has invalid character at column *xx*

Explanation

The automated file selection utility encountered an invalid character in the DB2ID, HWSID, or IMSID operand specification starting at column *xx.*

System action

Automated file selection processing terminates.

User response

Correct the operand value, then resubmit the job.

FUW0506E *Operand* operand has invalid syntax

Explanation

The automated file selection utility detected that the specified operand had a syntax error.

System action

Automated file selection processing terminates.

User response

Correct the syntax error, then resubmit the job.

FUW0507E Date specified in *operand* operand invalid, RSN=*nnn*

Explanation

The automated file selection utility has found an invalid date specified in the *operand* operand, and issues one of the following reason codes:

001
Date specification is wrong length

002
Year specification is zero or wrong length

003
Julian day specification is zero or wrong length

004
Hour specification is zero or wrong length

005
Minute specification is invalid

006
Second specification is zero or wrong length

007
Fraction of second specification is wrong length

008
Month is not between one (1) and twelve (12)

009
Day specification is zero or wrong length

010
Relative date exceeded 9999 days

System action

Automated file selection processing terminates.

User response

Correct the date specification, then resubmit the job.

FUW0508E IMSID operand not specified

Explanation

The automated file selection utility requires at least one IMSID operand to be specified.

System action

Automated file selection processing terminates.

User response

Specify the IMS subsystem name in the IMSID operand, then resubmit the job.

FUW0509E Start time of first SLDS not found in RECON data set, SSID=*ssid*

Explanation

The automated file selection utility processing could not determine the start time of the first SLDS record. The DBRC API failed to return the start time. The message can be issued for the following reasons:

- There are no SLDS records in the RECON data sets.
- There is a problem with the RECON data sets.
- There is a problem with DBRC.

System action

Automated file selection processing terminates.

User response

Verify that the RECON data sets contain SLDS records for the requested IMS subsystem. Otherwise, contact IBM Software Support.

FUW0510E End time of first SLDS not found in RECON data set, SSID=*ssid*

Explanation

The automated file selection utility processing could not determine the end time of the first SLDS record. The DBRC API failed to return the end time. The message can be issued for the following reasons:

- There are no SLDS records in the RECON data sets.
- There is a problem with the RECON data sets.
- There is a problem with DBRC.

System action

Automated file selection processing terminates.

User response

Verify that the RECON data sets contain SLDS records for the requested IMS subsystem. Otherwise, contact IBM Software Support.

FUW0511E Start time of last SLDS not found in RECON data set, SSID=*ssid*

Explanation

The automated file selection utility processing could not determine the start time of the last SLDS record. The DBRC API failed to return the start time. The message can be issued for the following reasons:

- There are no SLDS records in the RECON data sets.
- There is a problem with the RECON data sets.
- There is a problem with DBRC.

System action

Automated file selection processing terminates.

User response

Verify that the RECON data sets contain SLDS records for the requested IMS subsystem. Otherwise, contact IBM Software Support.

FUW0512E **End time of last SLDS not found in RECON data set, SSID=*ssid***

Explanation

The automated file selection utility processing could not determine the end time of the last SLDS record. The DBRC API failed to return the end time. The message can be issued for the following reasons:

- There are no SLDS records in the RECON data sets.
- There is a problem with the RECON data sets.
- There is a problem with DBRC.

System action

Automated file selection processing terminates.

User response

Verify that the RECON data sets contain SLDS records for the requested IMS subsystem. Otherwise, contact IBM Software Support.

FUW0513W **No Log files for the required time range are available, SSID=*value***

Explanation:
The automated file selection utility detected that there were no SLDS records in the RECON data set within the specified time range for IMS subsystem *value*.

System action:
The automated file selection utility continues attempting to select log data set for other IMS IDs in the request.

User response:
Check the reporting time range, if there is a omission correct the reporting time range, then resubmit the job.

FUW0514W **Log files not available for the complete time range, report period truncated, SSID=*ssid***

Explanation

The automated file selection utility detected that the SLDS records in the RECON data set for the subsystem only partially cover the required date range. The report interval is truncated.

System action

Automated file selection processing continues.

User response

If reporting is required for the entire date/time range, ensure that SLDS records for that range are available in the RECON data set, then resubmit the job.

FUW0515E **DBRC Utility (DSPURX00) failed to return the LOG variable**

Explanation

The DBRC routine did not return the LOG variable in the skeleton.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0516E **DBRC Utility (DSPURX00) ATTACH error, RC=*xx***

Explanation

The automated file selection utility received a bad return code from the ATTACH macro.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0517E **RDJFCB error for DDname *dddddddd*, RC=*xx***

Explanation

The automated file selection utility received a bad return code from the RDJFCB macro for DDname *dddddddd*.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0518E **DDname *ddname* not specified in JCL**

Explanation

The automated file selection utility DDname *ddname* was not specified in the JCL, but is required.

System action

Automated file selection processing terminates.

User response

Ensure the DDname *ddname* is specified in the JCL, then resubmit the job.

FUW0519E DBRC Utility (DSPURX00) Skeleton generation failed, LOG FROM time expected

Explanation

The automated file selection utility has found output from the DBRC routine in the wrong sequence or missing.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0520E DBRC Utility (DSPURX00) Skeleton generation failed, LOG TO time expected

Explanation

The automated file selection utility has found output from DBRC in the wrong sequence or missing.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. IBM Software Support.

FUW0521E DBRC Utility (DSPURX00) has failed, RC=*xx*

Explanation

The DBRC routine has failed to return SLDS information to the automated file selection utility. The SYSPRINT output file contains run information, including DBRC error messages to further explain the problem.

System action

Automated file selection processing terminates.

User response

Check the output from DBRC routine in the SYSPRINT output file. If you cannot resolve the problem, contact IBM Software Support.

FUW0522S DBRC Utility (DSPURX00) has abended, CODE=*code*

Explanation

The automated file selection utility has detected an abend in the DBRC routine. The SYSPRINT output file contains run information, including DBRC error messages to further explain the problem.

System action

Transaction Analysis Workbench processing terminates.

User response

Check the output from DBRC routine in the SYSPRINT output file. If the abend code is S806, check the specified RESLIB and that the DBRC API is available in the RESLIB. If you cannot resolve the problem, contact IBM Software Support.

FUW0523E OPEN failed for DDname *ddname*, RC=*xx*

Explanation

The automated file selection utility received a bad return code from the OPEN SVC when opening the specified DDname.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0524E IMSID operand exceeds maximum length (4)

Explanation

The automated file selection utility has detected that the IMS subsystem ID specification in the IMSID operand is longer than the maximum of four characters.

System action

Automated file selection processing terminates.

User response

Correct the IMSID operand, then resubmit the job.

FUW0525E **TO time is not greater than FROM time**

Explanation

The automated file selection utility has detected that the TO time specified is not greater than the FROM time.

System action

Automated file selection processing terminates.

User response

Correct the FROM and TO times, then resubmit the job.

FUW0526E **IMS release *xxx* is not supported**

Explanation

The automated file selection utility has detected an invalid or unsupported release of IMS specified in the VRM operand. Transaction Analysis Workbench supports IMS releases of V15.4 and later.

System action

Automated file selection processing terminates.

User response

Correct the VRM operand, then resubmit the job.

FUW0527E **Date format is invalid**

Explanation

The automated file selection utility encountered a date that did not adhere to the required format. Date must be a valid calendar date in the format yyyy-mm-dd, or a relative date such as 0, -1, -2 representing today, yesterday, and so on.

System action

Automated file selection processing terminates.

User response

Correct the date format, then resubmit the job.

FUW0528E **Dynamic Allocation failed, DDname *dddddddd,* SSID=*ssssssss,* RC=*xx*/EC=*eeee*/IC=*iiii***

Explanation

The automated file selection utility has failed to allocate the specified DDname. The error and information codes explain the cause of the problem.

System action

Automated file selection processing terminates.

User response

If you cannot correct the problem, then contact IBM Software Support.

FUW0529E **RECON specification error; RECON1 and RECON2 is the minimum specification**

Explanation

The automated file selection utility detected an error in the RECON data set specification. The DBRC routine requires at least two RECON data sets to be specified. At least RECON1 and RECON2 must be specified.

System action

Automated file selection processing terminates.

User response

Specify at least two RECON data sets, then resubmit the job.

FUW0530E **BLDL failed for *xxxx* MDA Members, RC=*xx*/Reas=*yy***

Explanation

The automated file selection utility received a bad return code from the BLDL macro.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0531E **LOAD failed for *xxxx* MDA member *mmmmmmmm*, ABEND=*aaaa-rr***

Explanation

The automated file selection utility failed to load an MDA member. The ABEND code explains the reason for the failure. This abend code could indicate:

- The job region size is too small
- MDA member has an I/O error

System action

Automated file selection processing terminates.

User response

Check the abend code, and if possible, correct the problem. Otherwise, contact IBM Software Support.

FUW0532E ***xxxx* RECON MDA member *mmmmmmmm* has invalid format**

Explanation

The automated file selection utility detected the MDA member for the specified RECON is not in MDA format.

System action

Automated file selection processing terminates.

User response

Verify that RECON MDA member has been generated correctly. Otherwise, contact IBM Software Support.

FUW0533E **Specified *xxxx* RECON data sets do not exactly match RECON MDA members**

Explanation

The automated file selection utility detected that the explicitly specified RECON data sets do not exactly match the RECON MDA members. When the RECON data sets are explicitly specified, the automated file selection utility also checks the RECON MDA members. If at least one RECON MDA member is detected, then the specified RECON data set names must exactly match the RECON MDA members. This ensures that the DBRC RECON data sets are not corrupted by the DBRC API routine DSPAPI00.

System action

Automated file selection processing terminates.

User response

Correct the RECON data sets specification, then resubmit the job.

FUW0534E **Operand *operand* specified more than once**

Explanation

The automated file selection utility detected that the specified operand was duplicated for the subsystem. Each operand can only be specified once per subsystem.

System action

Automated file selection processing terminates.

User response

Correct the duplicated operand specification, then resubmit the job.

FUW0535E **TAPES operand not numeric or greater than 35**

Explanation:
The automated file selection utility requires the TAPES operand to be between 1 and 35 inclusive, or not specified.

System action:
The automated file selection utility terminates.

User response:
Either correct or remove the TAPES operand, then resubmit the job.

FUW0536E **Data set name is longer than 44 characters**

Explanation

The automated file selection utility has detected a data set name longer than 44 characters.

System action

Automated file selection processing terminates.

User response

Correct the data set name specification, then resubmit the job.

FUW0537E **IMS subsystem *ssss* has no VRM operand**

Explanation

The automated file selection utility requires that each IMS subsystem has a VRM operand to specify the release of the subsystem, or that it can be determined from the RESLIB using DFSVC000.

System action

Automated file selection processing terminates.

User response

Ensure each IMS subsystem has a VRM operand specified, then resubmit the job.

FUW0538I **DB2 Log selection completed, RC=*value*, SSID=*value*, FROM=*value*, TO=*value***

Explanation

DB2 BSDS Log selection processing has completed. The following return codes are issued:

00
Processing completed successfully

04
Processing completed, warning message issued

08
Processing failed, error message issued

16
Operand specification error.

System action:
If the return code is 0 or 4, automated input file selection completed successfully. If the return code is 8 or 16, input file selection has failed. Previous error message(s) explain the reason for the error.

User response:
None required.

FUW0539E **Dynamic Deallocation failed, DDname=*dddddddd*, SSID=*ssssssss*, RC=*xx*/EC=*eeee*/IC=*iiii***

Explanation

The automated file selection utility has failed to deallocate the specified DDname. The error and information codes explain the cause of the problem.

System action

Automated file selection processing terminates.

User response

If you cannot correct the problem, then contact IBM Software Support.

FUW0540E **LOCATE failed for *xxxx* Log data set, RC=*xx*/DSN=*dddddddd***

Explanation

The automated file selection utility could not locate the catalog entry for the specified Log data set.

Transaction Analysis Workbench requires the device type (UNIT) information of each Log data set when using shared queue merge processing. Transaction Analysis Workbench uses the catalog when UNIT information is not available in the RECON.

System action

Automated file selection processing terminates.

User response

Either catalog the Log data set or use DBRC utilities to update UNIT and VOLSER information in the RECON.

FUW0541W ***xxxx* Log data set UNIT information is incomplete, DSN=*dddddddd***

Explanation

The automated file selection utility has detected that the specified Log data set has VOLSER information in the RECON, but UNIT information was not available.

System action

Automated file selection processing terminates.

User response

Use DBRC utilities to update the UNIT information in the RECON.

FUW0542E ***xxxx* Log data set has unsupported UNIT type, DSN=*datasetname***

Explanation

The automated file selection utility has detected that the specified Log data set is not eligible as it is not defined as either a TAPE or DASD device on this processor.

System action

Automated file selection processing terminates.

User response

Move the log data set to an eligible device or exclude the log data set.

FUW0543E ***xxxx*** **LOG data set is not cataloged, DSN=*ddddddddd***

Explanation

The automated file selection utility failed to locate the catalog entry for the specified Log data set. Transaction Analysis Workbench requires the UNIT information from the catalog if this information is not available from the RECON.

System action

Automated file selection processing terminates.

User response

Either catalog the LOG data set or use DBRC utilities to update unit information in your RECON data set.

FUW0544E ***xxxx*** **Log data set has more than 255 volumes, DSN=*ddddddddd***

Explanation

The automated file selection utility has detected that the specified Log data set has more than 255 volumes. Transaction Analysis Workbench does not support more than 255 volumes per Log data set.

System action

Automated file selection processing terminates.

User response

The automated file selection utility cannot be used for this Log file.

FUW0545E **No input files were eligible for processing**

Explanation

The automated file selection utility has found no SLDS log data sets, DB2 log files, or IMS Connect Extensions journal data sets to process for one of the following reasons:

- No IMS subsystems were specified.
- No Log data sets are available for the specified time period.
- DBRC failed to return log data set information from the RECON.
- The IMS Connect Extensions definitions repository contained no eligible journals.

A previous message will explain the reason for the error.

System action

Automated file selection processing terminates.

User response

Correct the problem, then resubmit the job.

FUW0546E **FUWSMQW1 Merge Work File DD statement not found in Report JCL skeleton**

Explanation:
The automated file selection utility could not find the FUWSMQW1 DD statement in the Report JCL skeleton. The automated file selection utility requires merge work file(s) for shared queue processing. When not enough tape units are available to concurrently process the log input from all subsystems, the automated file selection utility requires merge work files.

System action:
The automated file selection utility terminates.

User response:
Specify FUWSMQW1 DD statement in the Report JCL skeleton, then resubmit the job.

FUW0547I **DBRC Utility (DSPURX00) completed, RC=*value*, SSID=*value*, FROM=*value*, TO=*value***

Explanation:
The automated file selection utility has successfully invoked the DBRC Utility DSPURX00. When RC=0, DBRC has returned SLDS entries for the specified time range. When RC=4 DBRC could not find any SLDS entries for the specified time range. In this case, the automated file selection utility will re-invoke DBRC with a modified time range to retrieve the first required SLDS entry.

System action:
Automated file selection utility processing continues.

User response:
None required if RC=0 or 4. When RC=12 check that the RECON dataset is at the same IMS level as the library which contains DSPURX00 (usually specified by the RESLIB parameter).

FUW0548I **Report JCL has been submitted, Jobname=*value*, Jobno.=*value***

Explanation:

The automated file selection utility has submitted the Report Set JCL. The job name and job number are specified.

System action:
Automated file selection processing continues.

User response:
None required.

FUW0549E CAT operand must be YES or NO

Explanation

The automated file selection utility detected that the CAT operand was not specified as YES (SLDS data sets are cataloged) or NO (SLDS data sets are not cataloged).

System action

Automated file selection processing terminates.

User response

Correct the CAT operand, then resubmit the job. The dialog generates this operand from **Log Data Sets are Cataloged** on the Log Input - IMS Subsystem panel.

FUW0550E SSID operand exceeds maximum length (8)

Explanation

The SSID operand is longer than the maximum allowed.

System action

Automated file selection processing terminates.

User response

Correct the SSID operand, then resubmit the job.

FUW0551E SSID operand has invalid character at column *cc*

Explanation

The SSID operand contains invalid characters.

System action

Automated file selection processing terminates.

User response

Correct the SSID operand, then resubmit the job.

FUW0552E SLDS operand must be SEC or PRI

Explanation

The automated file selection utility SLDS operand was incorrectly specified.

System action

Automated file selection processing terminates.

User response

Ensure the OLDS operand is specified as SEC or PRI, then resubmit the job.

FUW0553E More than 1035 log files required for SSID=*ssssssss*

Explanation

The FROM and TO date range selected has resulted in the selection of more than the maximum allowed number of log data sets for this subsystem.

System action

Automated file selection processing terminates.

User response

Reduce the FROM, TO time range, then resubmit the job.

FUW0555E *volsername* VOLSER name is too long. DSN=*datasetname*, VOLSER=*volser*

Explanation

The VOLSER specified for the data set is greater than 6 characters.

System action

Automated file selection processing terminates.

User response

Automated file selection utility error. Contact your IBM Software Support.

FUW0556I IMS DBRC API module DSPAPI00 not found

Explanation

The DBRC API module DSPAPI00 was not found in the RESLIB specified. If RESLIB was not specified, then the DSPAPI00 module was not found in the STEPLIB or JOBLIB concatenation.

System action

Automated file selection processing continues.

User response

None required.

FUW0557I The RECON data set has not been upgraded, SSID=*ssssssss*

Explanation

The RECON data sets have not been upgraded to the format required by the IMS version specified by the RESLIB parameter.

System action

Automated file selection processing continues.

User response

None required.

FUW0558I The log data sets will be located using DSPURX00.

Explanation:
This message will always come after message FUW0556E or FUW0557I messages.

System action:
Automated file selection utility processing continues.

User response:
None required.

FUW0559I DBRC API (DSPAPI00) completed, RC=*rc*, SSID=*ssssssss*, FROM=*fromdate*, TO=*todate*

Explanation

Transaction Analysis Workbench has completed DSPAPI log selection processing, and issued the following return codes:

00
Processing completed successfully

04
Processing completed, warning message issued

08
Processing failed, error message issued

16
Operand specification error

System action

If the return code is 0 or 4, automated file selection completed successfully, and the Transaction Analysis Workbench report JCL has been submitted. If the return code is 8 or 16, automated file selection has failed. Previous error messages explain the reason for the error.

User response

None required.

FUW0560E OLDS operand must be YES or NO

Explanation

The automated file selection utility OLDS operand was incorrectly specified.

System action

Automated file selection processing terminates.

User response

Ensure the OLDS operand is specified as YES or NO, then resubmit the job.

FUW0561S DBRC API interface (IPIDBRCA) has abended, CODE=*value*

Explanation:
The automated input file selection API interface has detected an ABEND in the DBRC API interface module IPIDBRCA. The JESMSGLG output file contains the summary ABEND information.

System action:
The automated file selection utility terminates.

User response:
If you cannot resolve the problem, contact IBM Software Support.

FUW0562E No IMS Log files were eligible for processing

Explanation

IMS logs and IMS Connect Extensions journal data sets were requested for this run, however the automated file selection utility did not find any IMS systems with SLDS log data sets to process for one of the following reasons:

- No Log data sets are available for the specified time period.
- DBRC failed to return Log data set information from the RECON.

A previous message will explain the reason for the error.

System action

Automated file selection processing terminates.

User response

Correct the problem, then resubmit the job.

FUW0563E **No CEX journals were eligible for processing**

Explanation

IMS logs and IMS Connect Extensions journals were requested for this run, however the automated journal file selection utility did not find any IMS Connect systems with journal data sets to process for one of the following reasons:

- No journals are available for the specified time period.
- The IMS Connect Extensions definitions repository contained no eligible journals.

A previous message will explain the reason for the error.

System action

Automated file selection processing terminates.

User response

Correct the problem, then resubmit the job.

FUW0564E **No CEX journals were eligible for processing**

Explanation

The automated journal file selection utility did not find any IMS Connect Extensions journal data sets to process for one of the following reasons:

- No journals are available for the specified time period.
- The IMS Connect Extensions definitions repository contained no eligible journals.

A previous message will explain the reason for the error.

System action

Automated file selection processing terminates.

User response

Correct the problem, then resubmit the job.

FUW0565E ***Operand*** **operand is blank**

Explanation

DB2ID, HWSID, IMSID, PROBLEM, or SSID operand has not been specified.

System action

Automated file selection processing terminates.

User response

Correct the operand, then resubmit the job.

FUW0566E **Authorization failed: IMSID=*imsid*, IMSPLEX=*imsplex***

Explanation

The authorization exit ALZUAUTH disallowed the use of the IMSPLEX name for the IMS subsystem.

System action

Automated file selection processing terminates.

User response

Correct the IMSPLEX specification, then resubmit the job. If the IMSPLEX specification is correct and the problem still occurs, then the authorization exit needs updating. See SALZSAMP members ALZU002 and ALZUAUTH for more details.

FUW0567I ***type*** **file selected for system *name*; DSN=*dsname***

Explanation

A file that spans all or part of the required time range has been selected for the specified system. The file *type* can be one of the following:

- SMF
- NTH (OMEGAMON for DB2 near-term history file)

System action:
None.

User response:
None. Information only.

FUW0568E **SMF file selection failed for system *value*; PIPI RC=*value* SFPL RC=*value* REAS1=*value* REAS2=*value* Func=*value***

Explanation:
SMF files for the specified system could not be located. An internal processing error may have occurred.

System action:
Processing immediately stops.

User response

Verify that the system definition repository specified in the JCL, ddname FUWSYSDF, contains all the necessary information about the system:

1. Invoke ISPF dialog option 3 **Systems**
2. Ensure that the system definition repository for **CICS, DB2, more** is the same repository as DD FUWSYSDF.
3. Select the system type, such as CICS or DB2, that is requested in the JCL.
4. Ensure that the requested system is in the list, and then select it.
5. Select **Cyclic SMF Files**, and then ensure that the system has a list of files eligible for selection.

If the previous steps determine that the system definition is valid, then SMF file selection has failed because of an internal error. Contact IBM Software Support.

FUW0569E ***value*** **system** ***value*** **is not defined; FUWSYSDF DSN=*value***

Explanation:
The specified system is not defined in the System Definition repository.

System action:
Processing stops for this system only, and proceeds to the next request (if more than one).

User response:
Follow the steps outlined in the user response for message FUW0568E.

FUW0570I ***value*** **selected for subsystem** ***value***

Explanation:
A file has been selected for a subsystem.

System action:
None.

User response:
None. Information only.

FUW0571E **Repository initialization error, RC=*n***

Explanation

The Transaction Analysis Workbench session repository initialization call failed.

System action

Automated file selection processing stops.

User response

See the job log for the reason for failure. Contact IBM Software Support.

FUW0572E **Repository Problem Edit/Save error, RC=*n***

Explanation

The Transaction Analysis Workbench session repository Problem Edit/Save call failed.

System action

Automated file selection processing stops.

User response

See the job log for the reason for failure. Contact IBM Software Support.

FUW0573E **PROBLEM operand not specified**

Explanation:
The Transaction Analysis Workbench input file selection utility PROBLEM operand was not specified, but is required.

System action:
The automated file selection processing terminates.

User response:
Ensure the PROBLEM operand is specified in the command input, then resubmit the job.

FUW0574E **PROBLEM** ***value*** **is not registered in the session repository**

Explanation:
The Transaction Analysis Workbench input file selection utility PROBLEM operand was not found in the session repository.

System action:
The automated file selection utility terminates.

User response:
Ensure the PROBLEM operand is in the session repository, then resubmit the job.

FUW0575E **DB2ID operand exceeds maximum length (4)**

Explanation:
The automated file selection utility has detected that the DB2 subsystem ID specification in the IMSID operand is longer than the maximum of four characters.

System action:
The automated file selection utility terminates.

User response:
Correct the DB2ID operand, then resubmit the job.

FUW0576E **PROBLEM operand exceeds maximum length (8)**

Explanation:
The file selection utility has detected that the PROBLEM specification is more than 8 characters long.

System action:
Input file selection processing terminates.

User response:
Correct the operand, then resubmit the job.

FUW0577E ***value* system *value* has no SMF files in time range; FUWSYSDF DSN=*value***

Explanation:
The specified system is not defined in the System Definition repository.

System action:
Processing stops for this system only, and proceeds to the next request (if more than one).

User response:
Follow the steps outlined in the user response for message FUW0568E.

FUW0578W **Timezone for HWSID=*hwsid* is zero and local timezone is non-zero**

Explanation

The timezone offset in the IMS Connect Extensions definitions repository is zero and the local system timezone (CVTLDTO) is not.

System action

Automated file selection processing continues.

User response

If this situation is unexpected, check that the corequisite IMS Connect Extensions APAR has been applied.

FUW0579I ***Journals* Journals selected: *status***

Explanation

The IMS Connect journal file selection utility selected *Journals* journals, archive or active.

System action

Automated file selection processing continues.

User response

None required.

FUW0580E **IMS Connect Extensions service failed, RC=*rc***

Explanation

The IMS Connect Journal File Selection utility was unable to initialize its IMS Connect Extensions services environment.

System action

Automated file selection processing terminates.

User response

See the job log for the reason for failure and contact IBM Software Support.

FUW0581E **CEX Definitions Data Set access failed; DDname CEXDEF missing or invalid**

Explanation

The specified IMS Connect Extensions definitions repository data set is invalid or the required DD CEXDEF is missing from the JCL.

System action

Automated file selection processing terminates.

User response

Specify a valid IMS Connect Extensions definitions repository and resubmit the job.

FUW0582W **No journal data sets found for HWSID=*value* in the CEX Definitions Data Set**

Explanation:
Either there are no archive records in the CEX Definitions Data Set for this HWSID, or this HWSID does not exist in the CEX Definitions Data Set.

System action:
The automated file selection utility continues.

User response:
Correct the HWSID and resubmit the job.

FUW0583E **CEX Definitions Data Set access failed**

Explanation

The IMS Connect Journal File Selection utility could not use the IMS Connect Extensions definitions repository.

System action

Automated file selection processing terminates.

User response

Specify a valid IMS Connect Extensions definitions repository and resubmit the job.

FUW0584W **CEX Journals not available for the complete time range, report period truncated, HWSID=*hhhhhhhh***

Explanation

The IMS Connect Journal File Selection utility could not locate journal data sets that cover the entire requested reporting time range.

System action

Reporting continues with only partial coverage of the requested reporting time range.

User response

None required.

FUW0585E **Duplicate CEX subsystem ID specified, HWSID=*hhhhhhhh***

Explanation

The automated file selection utility detected a HWSID operand with a duplicate ID. An ID can only be specified once in the input.

System action

The automated file selection utility terminates.

User response

Correct the duplicated HWSID specification, then resubmit the job.

FUW0586E **HWSID operand exceeds maximum length (8)**

Explanation

The HWSID operand is longer than the maximum allowed.

System action

Automated file selection processing terminates.

User response

Correct the HWSID operand, then resubmit the job.

FUW0587E **DSPAPI Error, FUNC=*function*, RC=*rc*, RSN=X'*reasoncode*', SSID=*ssssssss***

Explanation

An unexpected return code and reason code were returned from the IMS DBRC API call. See Database Recovery Control (DBRC) in the *IMS System Programming API Reference* for a detailed explanation of the reason code.

System action

Automated file selection processing terminates.

User response

If the RECON data sets that you want to use belong to a running IMSplex:

- Ensure that you have correctly specified the name of the IMSplex in the **PARM** parameter of the **EXEC** statement for the automated file selection utility program.
- Ensure that the structured call interface (SCI) address space is running.

Otherwise, contact IBM Software Support.

Related reference

JCL for automated IMS log file selection (using DBRC)

To select IMS log files, the Transaction Analysis Workbench automated file selection utility uses the IMS database recovery (DBRC) API to read RECON data sets.

FUW0588E **DSPAPI Error, *xxxx* data returned does not match the requested ID, Requested=*rrrr*, Returned=*tttt***

Explanation

Unexpected data was returned from the IMS DBRC API call. Possible data corruption.

System action

automated file selection processing terminates.

User response

Automated file selection utility error. Contact IBM Software Support.

FUW0589I **CEX *xxxxxxxx* Journal selection completed, RC=*rc*, HWSID=*hwsid*, FROM=*fromdate*, TO=*todate***

Explanation

IMS Connect Journal File Selection processing completed, and issued one of the following return codes:

00
Processing completed successfully

04
Processing completed, warning message issued

08
Processing failed, error message issued

16
Operand specification error

System action

If the return code is 0 or 4, IMS Connect Journal File Selection completed successfully, and the report JCL has been submitted. If the return code is 8 or 16, IMS Connect Journal File Selection has failed. Previous error messages explain the reason for the error.

User response

None required.

FUW0591E **OMEGAMON for DB2 NTH SEQDATASET *error reason text***

Explanation

There was a problem selecting OMEGAMON for DB2 near-term history data sets for the specified time range.

The reason text explains the failure:

1. The DB2 system definition has an unsupported SEQDATASET specification
2. Catalog look-up error: message FUW0593I provides more information
3. No data sets were found in the catalog, based on the dynamic or static data set name pattern or the GDG base
4. The near-term history data set could not be allocated: message FUW5021E provides more information
5. The near-term history data set is either empty or does not contain DB2 IFCID records

System action:
No near-term history data sets were selected for the specified time range.

User response

The cause of the problem will determine the required response.

For example, you might need to correct the NTH SEQDATASET specification in the DB2 subsystem definition.

FUW0592I **OMEGAMON for DB2 NTH SEQDATASET Start=*yyyy-mm-dd hh:mm:ss.th* DSN=*dsname***

Explanation:
The specified OMEGAMON for DB2 near-term history data set is a candidate for file selection to provide DB2 trace data.

System action:
Message FUW0567I will follow if the data set is selected because it contains data that spans all or part of the requested time range.

User response:
None.

FUW0593I **CSI processing error: *reason text***

Explanation:
The catalog search to locate data sets failed.

System action:
Processing stops.

User response

See the Catalog Search Interface (CSI) return code reasons in the z/OS DFSMS documentation.

Check that the catalog is available and accessible. Otherwise, this message may indicate a problem: contact IBM Software Support.

FUW0700E **ALZEXEC Command is invalid**

Explanation:
The specified REXX command is invalid. The following commands are supported: READ, FETCH, and DISPLAY.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0701E ALZEXEC Parentheses not paired

Explanation:
The REXX command string specified parentheses that were not paired.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0702E ALZEXEC Parameter is invalid; Parameter=*parameter*

Explanation:
The parameter keyword is not valid for the command.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0703E ALZEXEC Required parameter missing; Parameter=*parameter*

Explanation:
A required parameter has not been specified for the command.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0704E ALZEXEC Only one of the parameters RRN, NEXT or PREV may be specified

Explanation:
For the READ command, the parameters RRN, NEXT, and PREV are mutually exclusive.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0705E ALZEXEC Only one of the parameters APPEND, PREPEND, REPLACE, EDIT, VIEW or BROWSE may be specified

Explanation:
For the DISPLAY command, parameters APPEND, PREPEND, REPLACE, EDIT, VIEW and BROWSE are mutually exclusive.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0706E ALZEXEC Required parameter value is missing; Parameter=*parameter*

Explanation:
The parameter requires a value.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0707E ALZEXEC Parameter has an invalid number of values; Parameter=*parameter*

Explanation:
The parameter contains an invalid number of values, for example too many values, or an odd number when an even number is required.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0708E ALZEXEC Parameter value is invalid; Parameter=*parameter*

Explanation:
The parameter value contains invalid characters or is too long.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0709E ALZEXEC Parameter requires no values; Parameter=*parameter*

Explanation:
The parameter is a keyword parameter and accepts no values in parentheses.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0710E ALZEXEC Parameters FILTER and CODE are mutually exclusive

Explanation:
For the READ command you can only specify either FILTER, or CODE, or neither.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec and re-run.

FUW0711E **ALZEXEC Duplicate parameter is invalid; Parameter=*parameter***

Explanation:
A duplicate parameter has been encountered.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec then re-run.

FUW0712E **ALZEXEC Parameter is invalid in batch mode; Parameter=*parameter***

Explanation:
Only sequential reads forwards are allowed in batch mode. The parameters RRN and PREV are not valid for the READ command when run in batch mode.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec then re-run.

FUW0713E **ALZEXEC DISPLAY command is invalid in batch**

Explanation:
The DISPLAY command is used to display data in the ISPF dialog. It cannot be used in batch.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec then re-run.

FUW0714E **ALZEXEC IO error**

Explanation:
An IO error has occurred while reading the logfile. The most likely cause is an attempted read past EOF. For a READ command the REXX exec must check for EOF by testing the RC.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Correct the REXX exec then re-run.

FUW0715E **ALZEXEC Storage not available**

Explanation:
ALZEXEC cannot getmain the storage required for processing.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Increase TSO logon size or batch region size and retry.

FUW0716E **ALZEXEC DISPLAY request but log file not yet open**

Explanation:
The DISPLAY command cannot be used until the log file has been opened.

System action:
ALZEXEC processing stops and RC is set to 20.

User response:
Run the selected REXX exec once the log file has been opened eg after log file browse (ALZPRBRF).

FUW1020E **IMS System Definition library open failed; *abend_code* DSN=*dsname***

Explanation

Transaction Analysis Workbench could not open the IMS system definition repository with the data set name *dsname*, because the specified data set is not a library (partitioned data set). The IMS system definition repository (also known as the IMS Performance Analyzer profile library) must be a PDS. For example, this error occurs if you mistakenly specify the data set name of your control repository (which is a VSAM KSDS, not a PDS).

System action

Processing stops.

User response

Specify the correct data set name for the IMS system definition repository, using either of the following dialog options:

- 3 **Systems** (the **System Definitions Menu**)
- 0.2 **Repositories**

FUW1041E **Reason=*reason* Member=*member* DSN=*dsname***

Explanation

A batch utility failed while attempting to use the specified data set.

System action

Processing stops.

User response

See the *reason* for more detailed diagnosis. Typically, the *reason* contains the corresponding abend and reason codes. In cases where the *dsname* does not

contain members, ignore the *member* name reported in the message.

FUW1042E **A required parameter is missing, invalid, or incompatible in the following definition: *definition***

Explanation

A definition in the input to a batch utility is missing a required parameter, specifies an invalid parameter, or specifies parameters that are incompatible with each other.

System action

Processing stops.

User response

Correct the input then resubmit the job.

FUW3000E **CDS Register VSAM Error, DDname=*dddddddd* Func=*ffff/ii* RC=*RC/RS* Req=ALZCDOLI Key=*kkkkkkkk***

Explanation

A VSAM I/O request to the control repository (also known as the control data set, or CDS) failed. Refer to z/OS *DFSMS Macro Instructions for Data Sets* for an explanation of the VSAM return code (RC) and reason code (RS). The I/O error can be caused by:

- Records in the KSDS not in the format or expected sequence, or
- A physical error with the KSDS data set

Other system messages may provide additional information about the error, and suggestions for corrective action. For example, an OPEN request with RC=08/98 indicates a security violation, and not a problem with the data set.

System action

The control repository request is aborted.

User response

If the message suggests a corruption of the data set then report the problem to IBM Software Support.

FUW3001E **CDS Data Set is corrupted, Reason=*xx***

Explanation

Your control repository (control data set) is corrupted, or an update action cannot be performed against it.

The Reason codes are:

01
CDS is empty on a non-Initialization call. Transaction Analysis Workbench automatically initializes the CDS data set when you first use it. This indicates an internal error.

02
CDS does not contain the CDS Control record. This indicates an internal error.

11
Filter Codes are missing.

21
Filter has too many Expressions.

31
Form Record is missing.

System action

Processing stops.

User response

The required action depends on the reason code:

01
Contact IBM Software Support.

02
Verify that the CDS has only ever been updated by the Transaction Analysis Workbench dialog or the IMS Problem Investigator dialog. Then contact IBM Software Support.

11
Delete the offending Filter. If the problem re-occurs, contact IBM Software Support.

21
Reduce the number of Conditions in the offending log record code for the Filter you are editing. The maximum number of Conditions is 434.

31
Delete the offending Form. If the problem re-occurs, contact IBM Software Support.

FUW3002W **CDS Object in use, try later. Name=*name***

Explanation

Your request to edit a control repository (control data set) object cannot be honored because another user is already editing it. The object can be a Filter, Form or Object List.

System action

Processing stops.

User response

Retry your request when the object becomes available.

FUW3003E ***object*** **not found, Name=*name***

Explanation

The specified object could not be found in the control repository (control data set). The object can be a Filter, Form or Object List.

System action

Processing stops.

User response

Refresh the list of objects by exiting the current panel, then retry your request. If the object still appears in the list but cannot be selected, then contact IBM Software Support.

FUW3004W **CDS Data Set not available, try later**

Explanation

Your request to update the control repository (control data set) could not be honored because another user is already updating it.

System action

Processing stops.

User response

Updates should complete very quickly, so retry your save request.

FUW3005E **ENQ macro failed, RC=*xx***

Explanation

The ENQ macro has failed with an unsupported return code.

System action

Processing stops.

User response

Exit ISPF to free the ENQ and then retry your request. If the problem re-occurs, contact IBM Software Support.

FUW3007W ***object*** **already exists, Name=*name***

Explanation

The specified object already exists in the control repository (control data set). You cannot create a new object with the same name. The object can be a Filter, Form or Object List.

System action

Processing stops.

User response

Select another name for the object and retry your request.

FUW3008W ***object*** **is required, Name=*name***

Explanation

The specified object cannot be deleted from the control repository (control data set) because another object references it. The object can be a Form or Object List.

System action

Processing stops.

User response

None. The object cannot be deleted.

FUW3009C **CDS - *failing component and action***

Explanation

Transaction Analysis Workbench has suffered a catastrophic failure in the specified component.

System action

Processing stops.

User response

If the problem recurs, contact IBM Software Support.

FUW3010E **CDS Data Set not defined, Request=*xx***

Explanation

The control repository for problem analysis is not cataloged.

Transaction Analysis Workbench needed the repository to access a filter, form, object list, or other control.

System action

Processing stops.

User response

Ensure that you have specified the problem analysis control repository under dialog option 0.2 **Repositories**. If the data set is not cataloged, you will be prompted to allocate it when access to a control is required. Or you can predefine the repository using IDCAMS. For details, see JCL member FUWALOCR in the sample library SFUWSAMP.

FUW3011E **Authorization failed ABEND 913, Access=Intent DSN=Data Set**

Explanation:
You do not have sufficient access authority to the specified data set. The access intent can be for Read or Update. When your security server is RACF then message ICH408I may be issued (to the job message log).

System action:
Processing stops.

User response:
Contact your RACF security administrator to grant you access to the data set.

FUW4000E **Unknown error: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation:
The request to stream JSON or CSV data via TCP/IP has failed. The reason for the failure is unknown. The *description*, *function*, and return code are from the point of failure, and can help to identify the reason for the failure.

System action:
Processing stops.

User response:
Contact IBM Software Support.

FUW4001E **System error: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation:
The request to stream JSON or CSV data via TCP/IP has failed while performing a POSIX system function. The return code is from that function. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact IBM Software Support.

FUW4002E **SSL/TLS error: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation:
The request to stream JSON or CSV data via TCP/IP has failed while performing a GSKit SSL/TLS function. The return code is from that function. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact IBM Software Support.

FUW4003E **Operation not permitted: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation

The request to stream JSON or CSV data via TCP/IP has failed for one of the following reasons:

- The operation is not possible, perhaps due to temporary conditions. For example, no spare ports are currently available for network connections.
- The current user does not have permission to perform the operation.

The return code is from the function that attempted to perform the operation. The *description* matches the return code.

System action:
Processing stops.

User response:
Use the *description*, *function* name, and return code to diagnose the reason for the failure.

FUW4004E **Connection could not be established: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation

The request to stream JSON or CSV data via TCP/IP has failed because a connection was refused or the host was unreachable.

The return code is from the function that attempted to establish the connection. The *description* matches the return code.

System action:

Processing stops.

User response:
Use the *description*, *function* name, and return code to diagnose the reason for the failure.

FUW4005E **Operation timed out: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP has failed because an operation timed out.

The return code is from the function that attempted to perform the operation. The *description* matches the return code.

System action:
Processing stops.

User response:
Consider using the **TIMEOUT** parameter of the **STREAM** command to specify a longer timeout. Otherwise, contact your system network support.

FUW4006E **Connection lost: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP has failed because the connection was closed by the peer or dropped.

The return code is from the function that detected the lost connection. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact your system network support.

FUW4007E **Key ring password error: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP with SSL/TLS has failed because the key ring password was missing, wrong, or expired.

The return code is from the function that detected the error. The *description* matches the return code.

System action:
Processing stops.

User response:
Use the **STASH** or **PASSWORD** parameter of the **STREAM** command to specify the correct password.

FUW4008E **Error opening key ring: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP with SSL/TLS has failed because an I/O or formatting error occurred opening the key ring.

The return code is from the function that detected the error. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact your system security administrator.

FUW4009E **Remote host's certificate could not be validated: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP with SSL/TLS has failed because the remote host's certificate could not be validated. Possible reasons include: the certificate could be self-signed, revoked, or have an unknown certificate authority (CA).

The return code is from the function that detected the error. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact your system security administrator.

FUW4010E **Remote host unsupported: *description*: *function*: rc=*decimal_rc*(*hex_rc*)**

Explanation
The request to stream JSON or CSV data via TCP/IP with SSL/TLS has failed because the remote host performed an action that is not supported.

The return code is from the function that detected the error. The *description* matches the return code.

System action:
Processing stops.

User response:
Contact your system security administrator.

FUW4100E **KEYRING is a required parameter when SECURITY has been specified**

Explanation:
A **STREAM** command specified a **SECURITY** parameter, but not a **KEYRING** parameter.

System action:
Processing stops.

User response:
For secure TCP, insert the missing **KEYRING** parameter. Otherwise, for unsecure TCP, remove the **SECURITY** parameter. Resubmit the job.

FUW5000E **Processing error *rsn*. INFO=*info*/*info2***

Explanation:
Generic error message capturing the error module, reason and associated feedback information.

System action:
Processing continues.

User response:
Probable server logic error. Capture the information to assist in problem diagnosis.

FUW5001I **Transaction Analysis Workbench product *action'***

Explanation:
Information messages regarding the state of the FUW product.

System action:
Processing continues.

User response:
None.

FUW5002E **Unable to open file *ddname* - DD statement missing**

Explanation:
DD statement is missing from the Common Services Library server JCL.

System action:
Transaction Analysis Workbench initialization fails and the product is stopped.

User response:
Add the missing DD statement to the server JCL.

FUW5003E **Required load module *mod* not found**

Explanation:
The required load module was not found.

System action:
IBM Transaction Analysis Workbench initialization fails and the product is stopped.

User response:
Add the required library to the server STEPLIB.

FUW5004E **Product initialization error**

Explanation:
FUW product initialization has encountered an error.

System action:
The FUW product will be stopped.

User response:
Probable server logic error. Capture information to assist in problem diagnosis.

FUW5005E **Conversation subtask initialization error. Subtask: *tskid***

Explanation:
Product subtask initialization has encountered an error.

System action:
Product subtask will be stopped and the associated request will be rejected.

User response:
Probable server logic error. Capture information to assist in problem diagnosis.

FUW5006I ***line* FUWCNTL: *card***

Explanation:
Information message echoing the FUWCNTL input control cards.

System action:
Processing continues.

User response:
None.

FUW5007E **No valid repositories defined**

Explanation:
FUWCNTL contains no repository definitions.

System action:
IBM Transaction Analysis Workbench initialization fails and the product is stopped.

User response:
Define at least one repository via FUWCNTL and restart the server.

FUW5008E **Syntax error(s) in FUWCNTL control cards**

Explanation:
FUWCNTL contains invalid control cards.

System action:
Transaction Analysis Workbench initialization fails and the product is stopped.

User response:
See the previous messages for the cause of the syntax errors. Correct the syntax errors, and then restart the server.

FUW5009E **Dynamic allocation failed for repository *rep*, data set *dsn***

Explanation:

Dynamic allocation failed for the named repository. One or more dynamic allocation messages precede this message.

System action:
Workbench initialization continues but the named repository is not available.

User response:
Determine the cause of the dynamic allocate failure. Correct and restart the server.

FUW5010I ***rep*** **repository** ***act*****, data set** ***dsn***

Explanation:
The named repository data set was successfully allocated/freed.

System action:
Processing continues.

User response:
None.

FUW5011I **Repository** ***typ dup*** **is a duplicate**

Explanation:
The repository name or data set name has been duplicated.

System action:
IBM Transaction Analysis Workbench initialization fails and the product is stopped.

User response:
Remove the duplicate specification.

FUW5012E **Command error:** ***reason text***

Explanation:
An FUW product command request sent to the Common Services Library server has invalid syntax. The reason text explains the cause of the problem. The user interface plug-in provides further diagnostics.

System action:
The server request fails.

User response:
If you are using the Transaction Analysis Workbench plug-in then the most likely cause is the FUW product running in the server is back-level. Ensure that maintenance is up to date. Otherwise contact IBM Software Support.

FUW5021E **DYNALLOC error EC=*xxxx* IC=*xxxx* DSN=*xxxx***

Explanation:
The data set could not be allocated. Additional system (IKJ) messages are issued to help explain the cause of the problem. A common error code is EC=1708 when the data set is not cataloged.

System action:
The request fails.

User response:
If the data set has been deleted then your request can no longer be satisfied.

FUW5022E **Data Set not available; RC=*xxxx* DSN=*xxxx***

Explanation:
The data set could not be processed. The RDJFCB return code might help to explain the cause of the problem.

System action:
The request fails.

User response:
Verify that the data set is accessible by browsing it in ISPF Browse.

FUW5023E **Data Set OPEN error; ABENDSsss-cc MEMBER=*member* DSN=*dsn***

Explanation

The member in the PDS data set could not be opened. The system ABEND code might help to explain the cause of the problem. Common abend codes include:

- ABENDS013-18: Member not found
- ABENDS913-38: Insufficient access authority (RACF)

An additional IEC150I message may be issued to further explain the cause of the problem.

System action:
The request fails.

User response:
Verify that the data set is accessible by browsing it in ISPF Browse.

FUW5024E **Member not found; RC=*xxxx* Member=*member* DSN=*dsn***

Explanation:
The member was not found in the data set. The BLDL return code might help to explain the cause of the problem.

System action:
The request fails.

User response:
Verify that the data set contains the required member by browsing it in ISPF Browse.

FUW5100I **Invalid command -** ***type***

Explanation:
GUI client command is invalid for a given reason.

System action:
The command is rejected with a response that includes this error message.

User response:
Probable user error. Correct and redrive the request.

FUW5101I **The command entered has invalid syntax or contains an invalid keyword**

Explanation:
GUI client command specified failed in the command parser.

System action:
The command is rejected with a response that includes this error message.

User response:
Probable user error. Correct and redrive the request.

FUW5102I **DDname *ddname* not allocated**

Explanation:
GUI client command specified a ddname that is not allocated to Common Services Library server.

System action:
The command is rejected with a response that includes this error message.

User response:
Probable user error. Correct and redrive the request.

FUW8000E **ALZEXEC Record not found; RRN=**

Explanation:
The specified relative record number was not found in the current log file.

System action:
ALZEXEC returns and sets RC to 8.

FUW8001E **ALZEXEC Segment not found; Segment=**

Explanation:
The segment specified in the FETCH command was not found in the current record.

System action:
ALZEXEC returns and sets RC to 8.

FUW8002E **ALZEXEC Field not found; Field=**

Explanation:
One or more of the fields specified in the FETCH command was not found in the current record, or segment if POSITION was specified.

System action:
ALZEXEC will set the REXX variables for the fields that are found, and set the REXX variables for the fields not found to null. RC is set to 8.

FUW8003E **ALZEXEC Field values ambiguous**

Explanation:
One or more fields specified in the FETCH command have more than one possible values. This is because the record contains multi-segments.

System action:
ALZEXEC returns and sets RC to 8.

User response:
Use the POSITION parameter to specify the particular segement from which the field value is required.

FUN-prefixed messages

These are the messages from the Common Services Library server.

FUN1003I **Processing *event* at *time***

Explanation:
These are startup and shutdown information messages. *event* specifies "started" or "ended".

System action:
The job continues.

User response:
None. Informational message only.

FUN3001E **Server terminating due to an error condition.**
Feedback: *feedback1: module_id/ rsn_code feedback2 feedback3*

Explanation:
An unsupported error condition has occurred. The server must terminate as its integrity is unknown. The feedback words contain IBM diagnostic and debugging information: the module ID and internal reason code for the module in which the error is generated.

System action:
Processing ends unconditionally and the server terminates.

User response:
Contact IBM Software Support.

FUN3002E **The server experienced an error condition.**
Feedback: *feedback1: module_id/ rsn_code feedback2 feedback3*

Explanation:
An unsupported error condition has occurred in the server. The server can continue processing. The feedback words contain IBM diagnostic and debugging information: the module ID and internal reason code for the module in which the error is generated.

System action:
Processing ends for the affected thread but the server attempts to continue processing.

User response:
Contact IBM Software Support.

FUN3003E **Unable to load module *module*: *description***

Explanation

module
Name of the module that could not be loaded.

description
One of the following:

- Module not found
- BLDL for module failed
- LOAD for module failed
- BPELOAD RC=*BPE return code*

As part of server or product initialization, a LOAD for a required load module failed.

System action

If the routine is a required server module the server will issue a FUN3001E error message and will terminate.

If the routine is a product-based required module, the product will fail initialization and will be stopped.

User response:
If possible, resolve the condition and restart the server or product. Otherwise, contact IBM Software Support.

FUN3004I **DUMPTRACE command ignored as optional DD *ddname* is not present**

Explanation:
A DUMPTRACE modify command was issued but the optional destination FUNDIAG DD is not present in the server job, so the command has been ignored.

System action:
The command is ignored. The server continues.

User response:
Information only; no response needed.

FUN3005I **DUMPSTATS command ignored as optional DD *ddname* is not present**

Explanation:
A DUMPSTATS modify command was issued but the optional destination FUNSTATS DD is not present in the server job, so the command has been ignored.

System action:
The command is ignored. The server continues.

User response:
Information only; no response needed.

FUN3006E **Product task abnormal termination.**
Product: *product code*
Symptom: CODE=*completion code from the trapped abend*
REASON=*reason code from the trapped abend*
Component: *product component name*

Explanation:
The server has recovered from an abend in a product task.

System action:
The product will be stopped. Server processing will continue.

User response:
Contact IBM Software Support.

FUN3007E **Conversation subtask abnormal termination**
Subtask: *product subtask identifier*
Symptom: CODE=*completion code from the trapped abend*
REASON=*reason code from the trapped abend*
Component: *product component name*

Explanation:
The server has recovered from an abend in a product subtask.

System action:
The failing product subtask will be restarted. Product processing will continue.

User response:
Contact IBM Software Support.

FUN3008I **Server in final termination phase, command ignored**

Explanation:
A command was issued after the server had reached the final termination phase. Command processing is suspended for the server when it is in this final phase, so the command has been ignored.

System action:
The command is ignored. Server termination continues.

User response:
Information only; no response needed.

FUN3009E **Server address space MEMLIMIT exceeded. Requested size: *n* MB**

Explanation:
A MEMLIMIT error condition has occurred in the server. The server can continue processing.

System action:

Processing ends for the affected thread but the server attempts to continue processing.

User response:
Raise the MEMLIMIT value for the address space to accommodate its storage needs. The server MEMLIMIT must match, or exceed, the server configuration specification for SDA_MAXLEN.

FUN3010I **F *command verb*,*command parameters (if any)***

Explanation:
A modify command was issued and has been acknowledged by the server.

System action:
Server command processing continues.

User response:
Information only; no response needed.

FUN3011W **Listener socket connection dropped out. New client connections are suspended**

Explanation:
Due to events external to the server, the Listener socket connection has dropped out.

System action:
Without the Listener socket connection the server is unable to accept new client connections. Existing client connections may be be able to continue depending on the event that has caused the Listener socket to be dropped. For example, if the cause was that TCP has ended, then all client connections will have been dropped too.

User response:
Use the RESTARTIP command, or recycle the server in order to reestablish the Listener socket connection.

FUN3012I **Insufficient access authority - UserID=*x***
SAF class: *SAF class* Access intent: *access intent*
Resource: *resource profile*

Explanation:
This message is issued when the server detects an unauthorized request (a violation) made by a user.

System action:
The user request will be rejected.

User response:
Follow the security procedures established for your installation. If no such procedures have been established, report the complete text of this message to the security administrator.

FUN3013E **Maximum initialization time exceeded for product '*product code*'**

Explanation:
The server has attempted to start the given product. However, the product failed to initialize in the maximum time allowed.

System action:
The product will be stopped. Server processing will continue.

User response:
Attempt to identify the cause of the product initialization delay in order to correct the issue. If possible, resolve the condition and restart the server or product. Otherwise, contact IBM Software Support.

FUN3014I **DISPLAY PRODUCT *product code***
Status . . . . . : *status indicator*

Explanation
Result of a console DISPLAY command:

`F server,DISPLAY PRODUCT product`

System action:
None.

User response:
Information only; no response needed.

FUN3015I **In-core user security profiles refreshed**

Explanation
Result of a console SECURITY command:

`F server,SECURITY REFRESH`

System action:
None.

User response:
Information only; no response needed.

FUN3101E **Configuration error: *description***

Explanation
An error in the JCL initialization script prevented the server from initializing. The error can be one of the following:

- FUNCFG keyword missing, check PARM in JCL
- FUNCFG parameter missing, check PARM in JCL
- FUNCFG parameter must specify a PDS member name

System action:
The server will terminate.

User response:
Review the startup JCL and ensure all parameters are valid and rerun the job.

FUN3102E **Error processing PROCLIB member *member***
Description: *description*

Explanation

member
Server configuration member

The server configuration parameter member is in error. The error can be one of the following:

- PROCLIB OPEN failed
- PROCLIB not in fixed format
- Member not found
- Member read failed
- Unsupported record format
- PROCLIB not LRECL=80
- Member too large

System action:
The server will terminate.

User response:
Review the startup JCL and ensure all parameters are valid and rerun the job.

FUN3103E **Error parsing PROCLIB member *member*, BPEPARSE RC=*rc***

Explanation

member
Server configuration member

rc
BPEPARSE return code

The server configuration parameter member is in error. BPE0003E console messages are issued with details of the error identified by the BPE parameter parser.

System action:
The server will terminate.

User response:
Review the server configuration member and ensure that all parameters are valid. Rerun the job.

FUN3104E **TCP_PORT is a required parameter. Specify a value in the range 1 through 65535**

Explanation:
The TCP_PORT server configuration parameter value was not specified, or was specified as zero. A value is required and must be in the range 1 - 65535.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3105E **Invalid maximum number of TCP input-threads: *n*. Valid range is 1 through 64**

Explanation

n
The TCP_THREADS value specified in the server configuration member

The TCP_THREADS server configuration parameter value is invalid. If specified, the value must be in the range 1 - 64. The default is 16.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3106E **Invalid server CCSID: *CCSID* - *description***

Explanation

CCSID
The CCSID value specified in the server configuration member

The CCSID server configuration parameter value is invalid. This represents the CCSID used by the server and is utilized for SDA data translation, when applicable. If specified, the CCSID must represent a single byte character set (SBCS) supported by z/OS Unicode Services. By default, a value of of 37 is used. That is, COM EUROPE EBCDIC.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3107E **Invalid SDA bar-limit: *n*. Valid range is 64 - 4096 KB**

Explanation

n
The SDA_BARLIM value specified in the server configuration member

The SDA_BARLIM server configuration parameter value is invalid. If specified, the value must be in the range 64 - 4096 KB. By default, a limit value of 2048 KB is used.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3108E **Invalid maximum SDA size: *n*. Valid range is 4 - 100 MB**

Explanation

n
The SDA_MAXLEN value specified in the server configuration member

The SDA_MAXLEN server configuration parameter value is invalid. If specified, the value must be in the range 4 - 100 MB. By default, a maximum value of 32 MB is used.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3109E **Invalid SAF class name: *name***

Explanation

name
The SAF_CLASS value specified in the server configuration member

The specified SAF class is not a valid SAF class name.

System action:
The server will terminate.

User response:
Ensure that SAF_CLASS is a valid SAF class name and specifies a defined resource class.

FUN3110E **SAF class not defined: *name***

Explanation

name
The SAF_CLASS value specified in the server configuration member

The SAF class could not be identified. Possible reasons:

- SAF-enabled security (RACF or similar) is not installed.
- The class was not defined.

System action:
The server will terminate.

User response:
Correct the server configuration member if the SAF class is not as expected, or ensure that the SAF class is defined.

FUN3111E **Invalid SERVER_NAME value: *name***

Explanation

name
The SERVER_NAME value specified in the server configuration member

The specified server name is not a valid name. A name must be 1 - 8 alphanumeric characters with no embedded blanks. However, the name cannot start with a numeric character. The characters @, #, and $ are also allowable and are treated as alphabetic.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3112E **Invalid maximum number of TCP sockets: *n*. Valid range is 50 through 2000**

Explanation

n
The TCP_MAXSOC value specified in the server configuration member

The TCP_MAXSOC server configuration parameter value is invalid. If specified, the value must be in the range 50 - 2000. The default is 50.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3113I **Duplicate PRODUCT code '*code*' will be ignored**

Explanation

code
The PRODUCT value specified in the server configuration member

The specified PRODUCT code is a duplicate of an earlier configuration parameter and will be ignored.

System action:
The server will continue.

User response:
Remove the duplicate parameter value to avoid this notification.

FUN3114E **Invalid or unsupported PRODUCT code: '*code*'**

Explanation

code
The PRODUCT value specified in the server configuration member

The specified PRODUCT code is invalid, or at least does not represent a product supported by the server.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3118E **Invalid TCP_RCVTIMEO value: *x*. Valid are 0 (no limit), or 100000 through 1000000 microseconds**

Explanation

n
The TCP_RCVTIMEO value specified in the server configuration member

The TCP_RCVTIMEO server configuration parameter value is invalid. If specified, the value must be 0 (no limit), or in the range 100,000 - 1,000,000. By default a value of 250,000 is used (0.25 seconds).

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3119E **Invalid TCP_SNDTIMEO value: *x*. Valid are 0 (no limit), or 100000 through 1000000 microseconds**

Explanation

n
The TCP_SNDTIMEO value specified in the server configuration member

The TCP_SNDTIMEO server configuration parameter value is invalid. If specified, the value must be 0 (no limit), or in the range 100,000 - 1,000,000. By default a value of 250,000 is used. Ie. 1/4 of a second.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3120E **Invalid PRD_MAXCNVQ# value: *n*. Use a value in the range 1 through 255**

Explanation

n
The PRD_MAXCNVQ# value specified in the server configuration member

The PRD_MAXCNVQ# server configuration parameter value is invalid. If specified, the value must be in the range 1 - 255. By default a value of 64 is used.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3121E **Server instance is already active for SERVER_NAME=*name***

Explanation

name
The SERVER_NAME value specified in the server configuration member

A server instance with the same SERVER_NAME is already active. The server name must be unique across the sysplex.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3122E **SERVER_NAME is a required parameter. Specify a 1 to 8 character name**

Explanation:
The SERVER_NAME server configuration parameter value was not specified, or was specified as blanks. A 1 - 8 alphanumeric character name is required.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3125E **Invalid TCP_NAME value: *name***

Explanation

name
The TCP_NAME value specified in the server configuration member

The specified TCP/IP stack name is not a valid name. A name must be 1 - 8 alphanumeric characters with no embedded blanks. However, the name cannot start with a numeric character. The characters @, #, and $ are also allowed and are treated as alphabetic.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3126E **Invalid TCP_IPV6 value: '*value*'. Use Y/N**

Explanation

value
The TCP_IPV6 value specified in the server configuration member

The specified TCP_IPV6 value is invalid. Specify Y or N.

System action:
The server will terminate.

User response:
Correct the parameter value and rerun the job.

FUN3205I Shutdown command received, server terminating

Explanation:
The server has received a SHUTDOWN command or console stop request and has commenced termination.

System action:
Server termination continues.

User response:
Information only; no response needed.

FUN3206I SHUTDOWN FORCE command received, server terminating

Explanation
The server has received a SHUTDOWN FORCE command and has either commenced server termination with FORCE, or upgraded an earlier shutdown request to use FORCE.

The FORCE option immediately terminates any outstanding client conversations that might be responsible for delaying server termination.

System action:
Server termination continues.

User response:
Information only; no response needed.

FUN3208E TCP address space *name* is not available, server terminating

Explanation

name
The TCP address space name specified in the server configuration member

The given TCP address space name is invalid.

System action:
The server will terminate.

User response:
Make the TCP address space available, or change the TCP_NAME parameter of the configuration member to the name of a TCP address space that is available. Alternatively the TCP_NAME configuration parameter can be removed, which will result in the system's default TCP address space being selected.

FUN3209E TCP/IP port *n* in use

Explanation

n
The TCP/IP port number value specified in the server configuration member

The specified TCP/IP port is currently in use.

System action:
Server continues without TCP/IP support.

User response:
Retry as TCP/IP can take up to 2 minutes to free a port. Change the TCP_PORT parameter of the configuration member.

FUN3210I TCP/IP using port *n*

Explanation

n
The TCP/IP port number value specified in the server configuration member

The given TCP/IP port is being used by the server.

System action:
None. Server continues.

User response:
None. Information only.

FUN3211E Shutdown command rejected, shutdown in progress

Explanation:
The server has received a SHUTDOWN command after it had already commenced server termination.

System action:
Server termination continues.

User response:
Information only; no response needed.

FUN3212I RESTARTIP initiated

Explanation:
The server has initiated the process to perform a RESTARTIP action.

System action:
The asynchronous process to perform the RESTARTIP action continues.

User response:
Information only; no response needed.

FUN3213I SECURITY REFRESH initiated

Explanation:

The server has initiated the process to perform a SECURITY REFRESH action.

System action:
The asynchronous process to perform the SECURITY REFRESH action continues.

User response:
Information only; no response needed.

FUN3214I **Product *code* stop initiated**

Explanation:
The server has initiated the process to perform a product STOP action for the named product.

System action:
The asynchronous process to perform the STOP action continues.

User response:
Information only; no response needed.

FUN3215I **Product *code* stopped**

Explanation:
The named product has been stopped.

System action:
None.

User response:
Information only; no response needed.

FUN3216I **Unable to stop product *code*, status: *state***

Explanation:
The server cannot STOP the named product at this time due to the product's given status.

System action:
The asynchronous process to perform the STOP action terminates.

User response:
Information only; no response needed.

FUN3218I **Unable to perform *action*, shutdown in progress**

Explanation:
The server cannot perform the named action as the server is in shutdown.

System action:
The asynchronous process to perform the name action terminates.

User response:
Information only; no response needed.

FUN3219I **Unable to start product *code*, shutdown in progress**

Explanation:
The server cannot START the named product at this time as the server is shutting down.

System action:
The asynchronous process to perform the START action terminates.

User response:
Information only; no response needed.

FUN3220I **Product *code* start initiated**

Explanation:
The server has initiated the process to perform a product START action for the named product.

System action:
The asynchronous process to perform the START action continues.

User response:
Information only; no response needed.

FUN3221I **Product *code* started**
Description ...: *description*
Version .....: *version.release.number* (*modification*)
Interface Level: *interface module APAR level/modification sublevel*

Explanation:
The named product has been started.

System action:
None.

User response:
Information only; no response needed.

FUN3222I **Unable to start product *code*, status: *state***

Explanation:
The server cannot START the named product at this time due to the product's given status.

System action:
The asynchronous process to perform the START action terminates.

User response:
Information only; no response needed.

FUN3223E ***DISPLAY*/*START*/*STOP* command failed due to an invalid product specification**
Valid products: *products*

Explanation

The command could not be performed because the product specification is invalid, or at least does not identify one of the products configured for the server.

The message text "Valid products: NONE" is possible for START or STOP commands where no products have been configured for the server.

System action:
None.

User response:
Correct the product specification and reissue the command.

FUN3224I **Command ignored, product *code* status: *state***

Explanation:
The command has been ignored as it is not applicable to the current state of the given product.

System action:
None.

User response:
Information only; no response needed.

FUN3225I ***command* command ignored, shutdown in progress**

Explanation:
The command has been ignored as it is not available during server shutdown.

System action:
None.

User response:
Information only; no response needed.

FUN3226I **Server start completed**

Explanation:
The server is now ready to accept client connections.

System action:
None.

User response:
Information only; no response needed.

FUN3227I **Product *code* initialization failed**

Explanation

Initialization has failed for the named product. This could be due to a number of reasons:

- Load failure for required product programs.
- Product CPROG rejected product INIT or failed.
- FUN definition or environment error.

Earlier messages should have been written identifying the cause of the initialization failure.

System action:
The product will be stopped.

User response:
Information only; no response needed.

FUN3228I **Product *code* stopping**

Explanation:
The server has commenced the process of stopping the named product.

System action:
Product STOP processing continues. Note that a product cannot stop while active request threads are outstanding, so the STOP process can be prolonged. No new external client requests will be accepted for the product at this stage.

User response:
Information only; no response needed.

FUN3231E **UNIX System Services callable service *name* not found**

Explanation:
The named USS callable service could not be found. This is a z/OS environmental error.

System action:
Server processing continues.

User response:
Consult your z/OS System Administrator to ensure that UNIX System Services (USS) has been properly installed and configured.

FUN3232E **UNIX System Services callable service *service* (*function*) RETURN_CODE *return code*, REASON_CODE *reason code***

Explanation

function
The TCP/IP function that was attempted

service
The function's USS callable service

return code
The return code as a decimal number

reason code
Further qualifies the RETURN_CODE value, given as a hexadecimal value ccccrrrr. cccc is a halfword reason code qualifier generally used to identify the issuing module and rrrr is the halfword reason code as described in the UNIX System Services Messages and Codes manual.

Common Services Library server received an unexpected return code attempting to perform the given TCP/IP callable service.

System action:
Server processing continues.

User response:
Look up the USS return code in *z/OS UNIX System Services Messages and Codes*.

FUN3233E **Unexpected TCP/IP response. IP operation *function* received ERRNO *error***

Explanation

function
The TCP/IP function that was attempted

error
The TCP/IP error number

Common Services Library server received an unexpected error attempting to perform the named TCP/IP *function*.

System action:
Server processing continues.

User response:
Look up the sockets return codes (ERRNOs) in *z/OS Communications Server IP Sockets Application Programming Interface Guide and Reference*.

FUN3234E **PassTicket generation failed RC=*RC* - Class=PTKTDATA, UserID=*UserID*, ApplName=*ApplName***

Explanation
A PassTicket generation request has failed.

RC
Return code from the RACF routine:

04
Incorrect PassTicket.

08
No PTKTDATA profile found for the application.

12
No task or address space ACEE found.

16
Caller is not authorized.

20
The RACF PTKTDATA class is not active.

24
Error in the session key generator process.

UserID
The user ID associated with the failed request.

ApplName
The application name associated with the failed request.

System action:
The processing thread that requested the PassTicket is terminated.

User response:
Contact IBM Software Support.

FUN3300I **Server default product (FUD) *action***

Explanation:
Information messages regarding the state of the server's default product (FUD).

System action:
Processing continues.

User response:
None.

FUN3301E **Required load module *mod* not found**

Explanation:
The required load module was not found.

System action:
The server's default product (FUD) will be stopped and the server will terminate.

User response:
Add the required library to the server STEPLIB.

FUN3302I **Invalid command - *type***

Explanation:
GUI client command is invalid for a given reason.

System action:
The command is rejected with a response that includes this error message.

User response:
Probable syntax error. Correct and redrive the request.

FUN3303I **The command entered has invalid syntax or contains an invalid keyword**

Explanation:
GUI client command specified failed in the command parser.

System action:
The command is rejected with a response that includes this error message.

User response:
Probable syntax error. Correct and redrive the request.

FUN3304E **Product initialization error**

Explanation:
Server's default product (FUD) initialization has encountered an error.

System action:
The server's default product (FUD) will be stopped and the server will terminate.

User response:
Probable server logic error. Capture information to assist in problem diagnosis.

FUN3305E **Conversation subtask initialization error. Subtask: *tskid***

Explanation:
Product subtask initialization has encountered an error.

System action:
Product subtask will be stopped and the associated request will be rejected.

User response:
Probable server logic error. Capture information to assist in problem diagnosis.

FUN3306W **No records found**

Explanation:
No data was found to match the specified parameters.

System action:
A null response (including this informational message and headers) is returned.

User response:
None.

FUN3307E **Dynamic allocation *type* failed for DSN=*dsn*, UID=*uid***

Explanation:
Dynamic allocation failed for the named object, where the allocation request was driven as part of user request processing. One or more dynamic allocation messages preceed this message.

System action:
Processing continues but the user request associated with the dynamic allocation fails.

User response:
Determine the cause of the dynamic allocation failure. Correct and retry the user request.

FUN3308E **JCLIN data set is not a card-image PDS, DSN=*dsn***

Explanation:
For a server submit command the specified JCLIN data set was found not to be a card-image (LRECL=80) PDS.

System action:
The server submit command fails.

User response:
Specify a card-image PDS and retry the request.

FUN3309E **Access denied to JCLIN data set, DSN=*dsn***

Explanation:
For a server submit command the requesting user is not authorized to read from the JCLIN data set.

System action:
The server submit command fails.

User response:
Correct the user authorization or modify the data set specification and retry the request.

FUN3310E **JCLIN member *mbr* not found, or found to be empty**

Explanation:
For a server submit command the specified JCLIN data set member was not found, or was found but had no records.

System action:
The server submit command fails.

User response:
Modify or respecify the JCLIN member and retry the request.

FUN3311E **JCL submitted to INTRDR but no job resulted**

Explanation:
For a server submit command the specified JCLIN data set member was tailored and submitted. However, no job resulted, which is indicative of invalid JCL with no JOB card.

System action:
The server submit command fails.

User response:
Modify or respecify the JCLIN member and retry the request.

FUN3312I **Job *jobid* submitted**

Explanation:
Job submission has been successful. The job identifier for the submitted job is given. However, if multiple jobs were submitted via a single JCLIN member, then only the last job identifier is returned.

System action:
None.

User response:
None.

FUN3399E **Processing error *rsn*. INFO=*info*/*info2***

Explanation:
Generic error message capturing the error module, reason and associated feedback information.

System action:
Processing continues.

User response:
Probable server logic error. Capture information to assist in problem diagnosis.

How to look up message explanations

You can use several methods to search for messages and codes.

Searching an information center

In the search box that is located in the top left toolbar of any Eclipse help system, such as the IBM Information Management Software for z/OS Solutions Information Center, enter the number of the message that you want to locate. For example, you can enter DFS1065A in the search field.

Use the following tips to improve your message searches:

- You can search for information on codes by entering the code; for example, enter -327.
- Enter the complete or partial message number. You can use the asterisk wildcard character (*) to represent multiple characters, and you can use the question mark wildcard character (?) to represent a single character.

The information center contains the latest message information for all of the Information Management products that are included in the information center.

Searching for messages on the web

You can use any of the popular search engines that are available on the web to search for message explanations. When you type the specific message number or code into the search engine, you are presented with links to the message information in IBM information centers.

Chapter 69. Gathering diagnostic information

Before you report a problem with Transaction Analysis Workbench to IBM Software Support, you need to gather the appropriate diagnostic information.

Provide the following information for all Transaction Analysis Workbench problems:

- A clear description of the problem and the steps that are required to re-create the problem
- All messages that were issued as a result of the problem
- Product release number and the number of the last program temporary fix (PTF) that was installed
- The versions of the relevant subsystems that you are using, such as DB2 or IMS, and the type and version of the operating system that you are using

Provide additional information based on the type of problem that you experienced:

For online abends, provide the following information

- A screen shot of the panel that you were using when the abend occurred
- The job log from the TSO session that encountered the abend
- The job log from the server
- A description of the task that you were doing before the abend occurred

For errors in batch processing, provide the following information

- The complete job log
- Print output
- Contents of the any data sets that were used during the processing

Obtaining dumps

If a Transaction Analysis Workbench batch utility abends, a system dump is written to the SYSUDUMP data set if the DD statement is included in the JCL. If possible, obtain a dump before contacting IBM Software Support. A dump will expedite problem identification and resolution.

Ensure that the JCL that runs the Transaction Analysis Workbench batch utility contains a SYSUDUMP DD statement.

To permanently insert a SYSUDUMP DD statement into JCL generated by the Transaction Analysis Workbench ISPF dialog, modify the JCL skeletons in the SFUWSENU library.

```
//userid  EXEC PGM=FUWBATCH
⋮
//SYSUDUMP DD  SYSOUT=*
```

Figure 115. Specifying SYSUDUMP DD in the JCL

Part 13. Reference

These topics provide reference information for Transaction Analysis Workbench.

Chapter 70. Filters: Log record selection criteria

Filters enable you to select the log records that you want to analyze and exclude others. For example, you can define a filter to select only those records of a particular log type and code that are associated with a particular transaction code or user ID. You can use a filter when browsing logs in the ISPF dialog or when writing JCL for the report and extract utility.

A filter consists of one or more log codes. The log codes identify the log records that you want to filter. More specifically, a filter consists of *combinations* of log type and log code; log codes qualified by their log type, to avoid ambiguity between log records in different log types that have the same log code. For example:

Table 18. Example log type and code combinations

Log type	Log code	Description
DB2	0002	DB2 log record: Page set control
MQ	0002	IBM MQ log extract record: Get
SMF	1E	SMF type 30 (X'1E') records
IMS	ALL	All IMS log records; that is, all log codes of log type IMS
None	ALL	All log codes of all log types

For each (combination of log type and) log code, a filter can optionally specify the following information:

Conditions
A condition is an expression that matches log records based on field values. For example, the following condition matches records whose TRANCODE field value is MENU:

```
TRANCODE EQ 'MENU'
```

A log code can have multiple conditions. Conditions are combined by a logical AND, with specific exceptions.

Exclude status
Whether you want to *exclude* or *select* records that match the log code and any conditions. The default behavior for a filter is to *select* matching records.

Level
A level number, 1 - 255. Log codes at the same level are combined by a logical OR. Different levels are combined by a logical AND. Levels are processed in ascending order: level 1, then level 2, etc. For example, if a log record does not match the log codes and their conditions for level 1, then no further levels are processed: the log record is discarded.

Form
The name of a form that you want to apply when browsing log records of this log code or creating formatted record reports of this log code.

REXX exec
The name of a REXX exec that you want to run when you are browsing logs and you select a log record of this log code, or you zoom on a field in such a log record.

Tracking
Whether or not you want to activate tracking. If tracking is active, the result of the filter includes not just the records that match the filter log codes and conditions, but also any records that belong to the same transactions as those matching records.

Where you can use filters

You can use a filter in the following situations:

Before starting the log browser in the ISPF dialog
To specify a filter before starting the log browser, use the **Filter** field on the **Process** panel for ad hoc log files or on the **Investigate** panel for sessions. You can specify either a filter name or a log code in this field.

To qualify a log code with its log type, specify the log type, a colon, and then the log code. For example, `IMS:08`.

To filter CICS trace entries using this method, you *must* qualify the log code. For example, `CTR:AP`. Otherwise, the **Filter** field interprets the alphanumeric log code of the CICS trace entry as a filter name.

Tip: To show all log codes of a particular log type, specify the log type, a colon, and then `ALL`. For example, to display only IMS log records, specify `IMS:ALL`.

While browsing logs
To specify a filter while browsing logs, select **Filter** in the action bar, or enter **`FILTER`** on the command line.

In the JCL for the report and extract utility
To specify a filter in the SYSIN data set for the report and extract utility, follow a command that processes log records (**`REPORT`**, **`EXTRACT`**, **`CSV`**, **`JSON`**, **`MWP`**, or **`REXX`**) with one or more **`CODE`** commands and related **`COND`** statements. Alternatively, some commands allow you to refer to a saved filter by name.

Related concepts

REXX API for formatting and analyzing logs
Transaction Analysis Workbench provides a REXX application programming interface (API) that you can use to format and analyze logs.

Related tasks

Defining forms
To hide log record fields that are not of interest to you, define and then use a form.

Related reference

CODE command and COND statement
The **`CODE`** command filters log records by including or excluding them based on their log type and code. To refine the filter based on field values in a log record, follow the **`CODE`** command with one or more **`COND`** statements.

Trace levels versus filters

Setting the trace level in Transaction Analysis Workbench offers an easier way to filter CICS and DB2 trace records than explicitly defining filters for the corresponding log codes.

Some subsystems, such as CICS and DB2, can write log records that are more specifically known as *trace* records. Typically, trace records contain event data for debugging or diagnosis rather than for performance statistics or accounting. You can configure each subsystem to write trace records for events at a particular level of detail.

Traces can generate many log records. When you browse logs, trace records can swamp the display. You might want to exclude trace records based on their level of detail. You can use filters to exclude each corresponding log code. However, defining filters to exclude the trace records that correspond to a particular level of detail can be onerous. Log types for traces can contain many log codes. For example, for DB2 trace records, each log code of the log type DTR represents an IFCID.

For CICS and DB2 trace records (log types CTR and DTR), as a more convenient alternative to using filters to explicitly exclude specific log codes, you can set a *trace level*, 0 - 4. Trace levels 0, 1, 2, and 3 display progressively more detailed events; trace level 4 displays all trace records.

To set the trace level while browsing logs in the ISPF dialog, enter TRACE *n* (where *n* is the level, 0 - 4) on the command line, or enter TRACE without a number to open a window where you can set the level.

You can use a filter to override the trace level. If you specify a filter that explicitly selects a log code for trace records not included in the current trace level, then those trace records are selected for processing; they are included in the log browser display.

DB2 IFCIDs by trace level

For a list of the DB2 IFCIDs at each trace level, see the example DB2 **START TRACE** commands for collecting DB2 trace data.

CICS trace point IDs by trace level

The following table lists the CICS trace point IDs at each trace level.

Table 19. CICS trace point IDs at each trace level

Trace level	Domain	Point ID	Description
0	XM	1102	Transaction attach OK: marks the start of the transaction
1	XS	070A	User ID security
1	PG	0901	Initial program link
1	LD	0002	Acquire program
1	AP	1940	Start program
1	AP	00E1	EXEC CICS LINK and RETURN
1	AP	E160	EXEC CICS LINK and RETURN
1	AP	E161	EXEC CICS LINK and RETURN
2	AP	00E1	EXEC CICS
2	AP	E160	EXEC CICS
2	AP	E161	EXEC CICS
2	AP	3250	EXEC SQL entry
2	AP	3251	EXEC SQL exit
2	AP	3181	EXEC SQL results
2	AP	0328	EXEC DLI entry
2	AP	0329	EXEC DLI exit
3	Any	Any	Exceptions
4	Any	Any	All other trace events

Related concepts

CICS trace
You can specify CICS auxiliary trace data sets or z/OS generalized trace facility (GTF) data sets as log files and analyze CICS trace entries in those files.

Collecting DB2 trace data
To collect DB2 trace data, including accounting data, use the DB2 command **START TRACE**.

Collecting CICS trace data

To collect CICS trace data (known in Transaction Analysis Workbench as log type CTR), either use the **CETR** transaction or the corresponding **SPCTRxx** SIT parameters.

Related tasks

Filtering log records while browsing logs
You can apply a filter to display only the log records that you want to analyze, and exclude others. For example, you can apply a filter to display only records of particular log types and codes, or records for a particular transaction code or user ID.

Defining filters

To select only the log records that are of interest to you, define and then use a filter. A filter specifies which log record codes to select or exclude and, optionally, more detailed conditions based on field values.

You can use filters while browsing logs in the ISPF dialog, and when using the report and extract utility.

The following procedure describes how to define a permanent filter that is stored in the control repository and that you can then refer to by name.

Alternatively, while browsing logs, you can define a temporary filter (also known as a "dynamic filter") by entering **FILTER** on the command line. Defining a temporary filter is similar to defining a permanent filter: see steps “4” on page 396 to “6” on page 397 in the following procedure.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 2 **Controls**.

   The **Controls** menu is displayed, showing the data set name of the control repository where filters are stored.
2. Select option 1 **Filters**.

   The **Filters** panel is displayed, showing a list of filters in the control repository.

   Tip: To display the **Filters** panel from anywhere in the dialog, enter **FILTERS** on the command line.
3. To create a new filter, enter **NEW** on the command line. To edit an existing filter, enter S next to the filter name.

   If you enter **NEW**, a window opens, requesting the name of the new filter, and whether you want to base (model) the new filter on an existing filter.

   By default, a new filter contains a line with a blank **Log** (log type) column value and ALL in the **Code** (log code) column. You can use this line to specify filter conditions that apply to all log codes of all log types.
4. To filter specific log types and codes:
   a) On a blank line of the filter, tab to the **Log** column, and then press the Prompt function key (F4) to select a log type.
   b) Tab to the **Code** column, and then press the Prompt function key (F4) to select a log code for that log type.

      Tips:

      - To filter all log codes of a particular log type, specify ALL in the **Code** column.
      - By default, a filter selects (*includes*) records for processing. To *exclude* a log code, enter X in the **/** column. To revert to selecting the log code, enter X again.

   For each line in a filter, you can optionally specify a form (to select only the fields that are of interest to you) and a REXX exec (to perform custom processing).
5. To refine a filter line by specifying conditions based on field values, enter S in the **/** column.

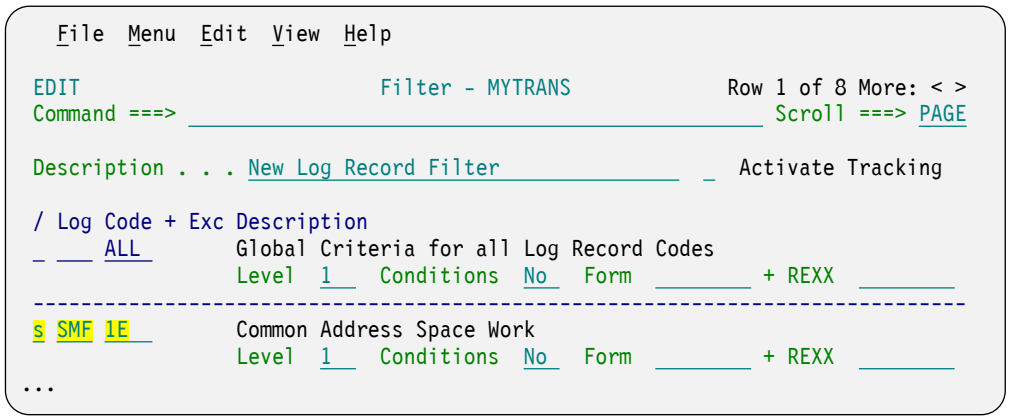


Figure 116. Panel: Defining a filter (for SMF type 30 records; log code 1E)

The **Conditions** panel is displayed. Enter one or more conditions.

For example, the two conditions in the following figure match SMF type 30 records that refer to the ddname EXTRACT and a job name that begins with FUW.

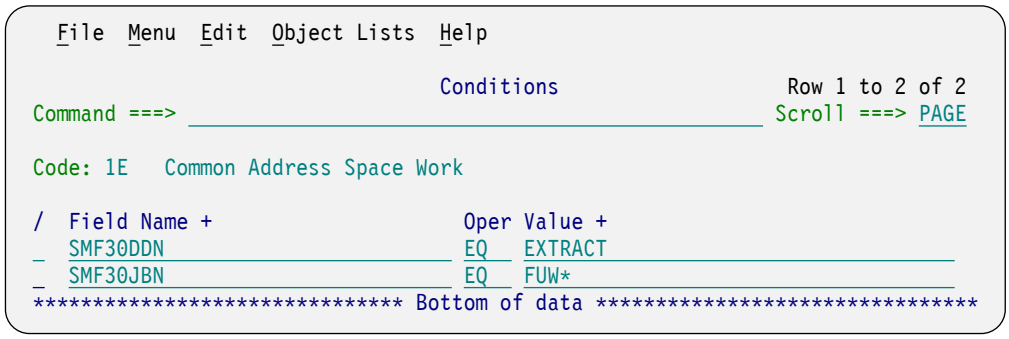


Figure 117. Panel: Defining filter conditions

When you have finished specifying conditions, press the Exit function key (F3) to return to the list of lines in the filter.

6. If you want the filter results to include not just the log records that explicitly match the filter, but also other log records that belong to the same transactions (or units of recovery) as those matching records, select **Activate Tracking**.
7. The first line in a new permanent filter matches all records (all log codes of all log types): it has a blank **Log** column value and ALL in the **Code** column. If you have not specified conditions for this line, delete it: enter D in the **/** column.

   Without conditions, this line matches all records, so any subsequent filter lines would be ignored.
8. Press the Exit function key (F3) to save the filter and return to the **Filters** panel.

   Tip: To exit the panel without saving the filter, press the Cancel function key (F12).

Related concepts

REXX API for formatting and analyzing logs

Transaction Analysis Workbench provides a REXX application programming interface (API) that you can use to format and analyze logs.

Related tasks

Filtering log records while browsing logs

You can apply a filter to display only the log records that you want to analyze, and exclude others. For example, you can apply a filter to display only records of particular log types and codes, or records for a particular transaction code or user ID.

Filter log types and codes

A filter specifies one or more combinations of log type and code, identifying the log records that the filter selects or excludes.

The following table describes the combinations of log type and code that a filter can specify. For some combinations, instead of a specific log code, a filter can specify the special value ALL.

Table 20. Filtering log records: log type and code combinations

To match...	**...specify this log type...**	**...and this log code**	**Example**
Records of a specific log type and code	A specific log type (for example, CMF)	A specific log code of that log type (for example, 6E13)	CMF 6E13 matches CICS monitoring facility (CMF) performance class records
Records of a specific log type, whose log codes begin with a specific two-digit sequence	A specific log type	The first two digits of a log code of that type	IMS 40 matches all IMS checkpoint records (log codes 4001 - 4099)
All records of a specific log type, regardless of log code	A specific log type	ALL	DB2 ALL matches all DB2 log records
All records, regardless of log type or code	None	ALL	

For each log type and code combination in a filter, you can optionally specify one or more conditions to refine the filter based on field values.

Related reference

Filter conditions
A filter condition is an expression that selects log records based on a field value. Conditions refine filters. Without conditions, filters select log records based only on log types and codes.

Log types and codes
Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Filter conditions

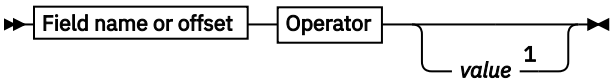
A filter condition is an expression that selects log records based on a field value. Conditions refine filters. Without conditions, filters select log records based only on log types and codes.

Format

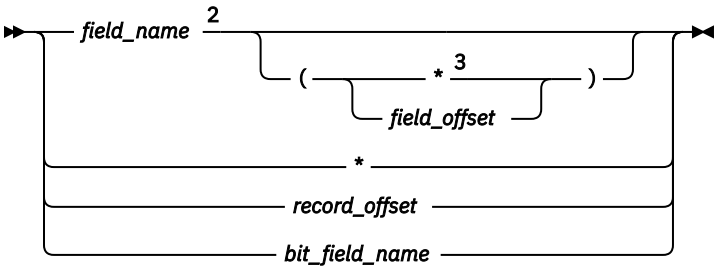
A condition typically consists of the following three parts:

- Field name or, to identify an unnamed sequence of bytes, offset
- Comparison operator
- Value

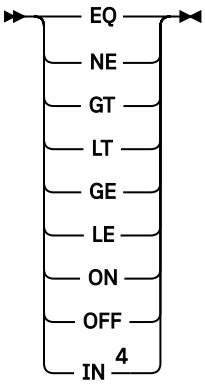
Format of a filter condition



Field name or offset



Operator



Notes:

[1] If the condition specifies the name of a bit field, specify the operator ON or OFF, and omit *value*. The condition compares the predefined value of the bit field with its parent flag field. For all other fields, *value* is required.

[2] *field_name* must name a valid field for the log type and code to which the condition applies. If the log type or code is ALL, *field_name* must name a global field.

[3] An asterisk (*) offset is known as a *floating offset*.

[4] The **IN** operator is available only when using the **COND** statement of the report and extract utility to specify a filter condition for a character field. When **IN** is specified, *value* is a list of string values separated by spaces or commas and enclosed in parentheses.

Example: EQ operator

In the following example, the field name is TRANCODE, the comparison operator is EQ ("equal to"), and the value is MENU:

```
TRANCODE EQ 'MENU'
```

Example: IN operator, only for use with the COND statement

When specifying filter conditions using the **COND** statement of the report and extract utility, rather than, for example, the **Conditions** panel of the ISPF dialog, you can use the IN operator. The IN operator offers a concise way to specify multiple allowed values for a character field. For example, the following single **COND** statement:

```
COND TRANCODE IN ('TRNA','TRNB','TRNC','APP*')
```

is equivalent to the following set of consecutive **COND** statements:

```
COND TRANCODE EQ 'TRNA'
COND TRANCODE EQ 'TRNB'
COND TRANCODE EQ 'TRNC'
COND TRANCODE EQ 'APP*'
```

Related reference

Filter log types and codes
A filter specifies one or more combinations of log type and code, identifying the log records that the filter selects or excludes.

CODE command and COND statement
The **CODE** command filters log records by including or excluding them based on their log type and code. To refine the filter based on field values in a log record, follow the **CODE** command with one or more **COND** statements.

Filter condition field names or offsets

The field name or offset in a filter condition specifies the part of a log record that you want to compare with the filter condition value.

Tip: To list all field names for a log code, create a form: on the primary option menu of the Transaction Analysis Workbench ISPF dialog, select option 2 **Forms**. Enter **NEW** on the command line. Specify the log type and code, and then press Enter. A new form is displayed, containing all fields in that log code. To exit without saving the new form, press the Cancel function key (F12).

The field name or offset must be in one of the following formats.

field_name
Begins the comparison from the start of the field. The comparison length is the length of the field or the length of the condition value, whichever is longer.

If the condition value is shorter than the field, then the condition value is padded before comparison, according to the following rules:

- Character values are left-aligned and padded with blanks (X'40')
- Hexadecimal values are left-aligned and padded with low values (X'00')
- Numeric values are converted to binary, right-aligned and padded with low values (X'00')

Example condition:

```
TRANCODE EQ 'MYTRAN'
```

True if TRANCODE is "MYTRAN". False if TRANCODE is "MYTRANX".

***field_name*(*field_offset*)**
Field name with offset: begins the comparison from an offset relative to the start of the field. The offset is one-based: 1 indicates the start of the field. The comparison length is the length of the condition value. The offset can take the comparison beyond the end of the field.

Example condition:

```
TRANCODE(2) EQ 'YTR'
```

True if TRANCODE contains "YTR" starting at position 2. For example, this condition is true if TRANCODE is "MYTRAN" or "ZYTROPE".

***field_name*(*)**
Field name with floating offset: begins the comparison anywhere in the field. The comparison length is the length of the condition value.

Example condition:

```
TRANCODE(*) EQ 'BANK'
```

True if any part of TRANCODE contains "BANK". For instance, this condition is true if TRANCODE is "MYBANK", "YOURBANK", or "EBANKING".

record_offset
Begins the comparison from an offset relative to the start of the record. The comparison length is the length of the condition value.

You can specify the offset in either decimal as a literal number, such as 16, or hexadecimal in the format X'*hex_digits*', such as X'10').

Decimal record offsets are one-based: 1 indicates the first position in the record. Hexadecimal record offsets are zero-based: X'0' indicates the first position in the record.

Example conditions:

```
5 EQ X'03'
X'04' EQ X'03'
```

These two conditions are equivalent. They are true if the fifth byte of the log record is X'03'.

****
Floating record offset: begins the comparison anywhere in the record. The comparison length is the length of the condition value.

Example condition:

```
* EQ 'TERMINAL99'
```

True if the string "TERMINAL99" occurs anywhere in the record.

bit_field_name
Compares the bits in the bits field name with the bits in the first byte of the parent flag field.

A bit field specifies a bit pattern (a value with one or more bits set on) for its parent flag field.

If you specify a bit field name, specify the operator ON or OFF, and do not specify a condition value: the bit field name implicitly specifies the condition value.

The ON operator only compares the bits that are set on in the bit field with the corresponding bit positions in the parent flag field. The OFF operator only compares the bits that are set off in the bit field with the corresponding bit positions in the parent flag field. The ON and OFF operators do not compare bits in any other positions.

Example conditions:

```
QDF2SMB ON
```

True if the bits set on in the bit field QDF2SMB are also on in the parent flag field, MSGDFLG2.

```
QDF2SMB OFF
```

True if the bits set off in the bit field QDF2SMB are also off in the parent flag field, MSGDFLG2.

Related reference
Global fields

Global fields are generic field names defined by Transaction Analysis Workbench that map to record-type specific field names in log records.

Filter condition operators

A filter condition operator determines how to compare a filter condition field, specified by a field name or offset, with its value.

A filter condition must contain one of the following operators:

EQ
Equal to

NE
Not equal to

GT
Greater than

LT
Less than

GE
Greater than or equal to

LE
Less than or equal to

ON
For checking flag bits only: the bits that are set on in the field are also on in the condition value

OFF
For checking flag bits only: the bits that are set off in the field are also off in the condition value

If the operator is ON or OFF, the following rules apply:

- ON and OFF check only the first byte of the field
- The condition must match either of the following cases:
  - The condition value specifies a bit pattern. For example, the following condition is true if the first and eighth bits of the field MSGDFLG2 are on, regardless of the value of any other bits in MSGDFLG2:

    ```
    MSGDFLG2 ON X'81'
    ```

  - The condition must specify the name of a bit field, and omit the value. The condition compares the predefined value of the bit field with its parent flag field. For example, the following condition is true if the bits set on in the bit field QDF2SMB are also on in the parent flag field MSGDFLG2, regardless of the value of any other bits in MSGDFLG2:

    ```
    QDF2SMB ON
    ```

Filter condition values

A filter condition value is the data that you want to compare with a named field or an offset (unnamed sequence of bytes) in a log record.

If the filter condition specifies a bit field name and the operator **ON** or **OFF**, do not specify a filter condition value.

Otherwise, the filter condition value must be in one of the following formats:

string
Character string, optionally enclosed in single or double quotation marks. Can contain the following masking characters:

%
Percent sign. Masks a single character.

*****
Asterisk. Masks zero or more characters.

X '*hex_digits*'
Hexadecimal data value.

0 - 99999999
1- to 8-digit integer.

&object_list
A leading ampersand (&) indicates a reference to the name of an object list. An object list consists of a set of values for a character field.

An object list enables you to define a set of related values once, and then refer to that set of values in many filter conditions.

For example, you can create an object list for all of the transaction codes that belong to a particular application system. If you create an object list named APPATRNS consisting of the values TRNA, TRNB, and TRNC, then the following filter condition:

```
TRANCODE EQ &APPATRNS
```

is equivalent to the following sequence of filter conditions:

```
TRANCODE EQ 'TRNA'
TRANCODE EQ 'TRNB'
TRANCODE EQ 'TRNC'
```

which, according to the rules for combining filter conditions, means "TRANCODE equal to TRNA, *or* equal to TRNB, *or* equal to TRNC".

Similarly, the following filter condition:

```
TRANCODE NE &APPATRNS
```

is equivalent to the following sequence of filter conditions:

```
TRANCODE NE 'TRNA'
TRANCODE NE 'TRNB'
TRANCODE NE 'TRNC'
```

("TRANCODE not equal to TRNA, *and* not equal to TRNB, *and* not equal to TRNC")

Object lists are stored in the control repository. To select an existing object list when editing a filter condition in the dialog, tab to the **Value** field on the **Conditions** panel, and press the Prompt function key (F4).

To create or edit object lists, select **Object Lists** in the action bar on the **Conditions** panel.

Rules for combining filter conditions

Multiple conditions for a log code are combined by a logical AND, with the following exception: consecutive conditions that have the same field name (or offset) and operator, where the operator is something other than NE (not equal to), are combined by a logical OR.

General case: AND

The following example demonstrates the general case, where conditions are combined by a logical AND. In this example, the two consecutive conditions select log records if their transaction code is not MENU *and* not LOGO (neither MENU nor LOGO).

```
TRANCODE NE 'MENU'
TRANCODE NE 'LOGO'
```

Exception: OR

The following example demonstrates the exception, where conditions are combined by a logical OR. In this example, the two consecutive conditions select log records if their transaction code is MENU *or* LOGO.

```
TRANCODE EQ 'MENU'
TRANCODE EQ 'LOGO'
```

Example filters

These examples illustrate some typical filtering requirements.

The examples are presented in a condensed layout that is similar to, but does not exactly match, the layout of the corresponding dialog panels.

Filter with multiple levels (shown as a logical expression)

This example shows a filter with multiple levels, and then presents the same filter as its equivalent logical expression, to help clarify how the filter is processed.

Filter

```
Log Code  Description       Level  Conditions

    ALL   Global Criteria   1      LSN GE 0000000000001023
                                   LSN LE 0000000000007829

IMS 01    IMS Message       2      TRANCODE EQ T1
                                   TRANCODE EQ T2
                                   USERID EQ JOHN
IMS 07    Condensed Command 2      TRANCODE EQ T3
                                   TRANCODE EQ T4
                                   USERID EQ BOB
```

Equivalent logical expression

```
(
  LSN GE 0000000000001023
  AND
  LSN LE 0000000000007829
)
AND
(
  (
     CODE EQ 01
     AND
     (
        TRANCODE EQ T1
        OR
        TRANCODE EQ T2
     )
     AND
     USERID EQ JOHN
  )
  OR
  (
    CODE EQ 07
    AND
    (
       TRANCODE EQ T3
       OR
       TRANCODE EQ T4
    )
    AND
    USERID EQ BOB
  )
)
```

Filter with multiple levels (shown as pseudocode)

This example shows a filter with multiple levels, and then presents the same filter as pseudocode, to help clarify how the filter is processed.

Filter

```
Log Code  Description       Level  Conditions

    ALL   Global Criteria   1      LSN GE 0000000000001000
                                   LSN LE 0000000000008000

IMS 01    IMS Message       2      MSGXSTXT(*) EQ 'UNAUTHORIZED'

IMS 03    IMS Message       2      MSGXSTXT(*) EQ 'UNAUTHORIZED'

IMS 50    Database Update   2      DATABASE EQ 'ACCOUNTS'
```

Equivalent pseudocode

```
For all log codes:
If (LSN >= 0000000000001000) AND (LSN <= 0000000000008000) then
  Continue to level 2
else
  Bypass record

For log code 01:
If message text contains the string 'UNAUTHORIZED' then
  Process record

For log code 03:
If message text contains the string 'UNAUTHORIZED' then
  Process record

For log code 50:
If database name is 'ACCOUNTS' then
  Process record

Otherwise:
  Bypass record
```

Chapter 71. Forms: Fields of interest in a log record

A form is a list of the fields that are of interest to you from one type of log record.

You can use forms when browsing a log record in the ISPF dialog, and when using the report and extract utility to create comma-separated values (CSV), JSON, or a formatted record report. Using forms makes it easier and quicker to interpret log records and reduces the volume of output.

Tip: When browsing log records or creating formatted record reports, instead of referring to a single form by name (specific to one log record type), you can use a filter that refers to multiple forms (for different log record types).

Forms are stored in the control repository.

Defining forms

To hide log record fields that are not of interest to you, define and then use a form.

1. On the Transaction Analysis Workbench Primary Option Menu, select option 2 **Controls**.

   The **Controls** menu is displayed, showing the data set name of the control repository where forms are stored.
2. Select option 2 **Forms**.

   The **Forms** panel is displayed, showing a list of existing forms in the control repository.

   Tip: To display the **Forms** panel from anywhere in the dialog, enter **FORMS** on the command line.
3. To create a new form, enter **NEW** on the command line. To edit an existing form, enter S next to the form.

   If you enter **NEW**, a window opens, requesting details of the new form, including the log record type. If you specify an IMS log record type, you must also specify an IMS version. You can choose to create the default form that includes all fields in the record, or base (model) the new form on an existing form.

   The list of fields in the form is displayed. Horizontal lines in the list, and ID column values, identify segments. Segments are record sections that contain a logical grouping of fields.
4. Remove the fields that are not of interest to you. To remove a single field, enter D next to the field. To remove a block of fields, enter DD next to the first and the last fields in the block.

   Tip: To undo the previous deletion, enter **UNDO** on the command line. To reinstate all deleted fields, either enter **UNDO** repeatedly or create a new form.
5. Press the Exit function key (F3) to save the form and return to the **Forms** panel.

   Tip: To exit the form edit panel without saving the form, press the Cancel function key (F12).

Related concepts

Filters: Log record selection criteria
Filters enable you to select the log records that you want to analyze and exclude others. For example, you can define a filter to select only those records of a particular log type and code that are associated with a particular transaction code or user ID. You can use a filter when browsing logs in the ISPF dialog or when writing JCL for the report and extract utility.

Repositories
Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Related tasks

Filtering log records while browsing logs

You can apply a filter to display only the log records that you want to analyze, and exclude others. For example, you can apply a filter to display only records of particular log types and codes, or records for a particular transaction code or user ID.

Creating CSV files from log files in your ad hoc list
You can use Transaction Analysis Workbench to save data from any supported log record type to a comma-separated values (CSV) file. You can use the CSV file with other applications, such as PC-based spreadsheet applications. A CSV file created by Transaction Analysis Workbench contains selected fields from a single log record type and code.

Using forms to exclude fields when browsing log records

You can use forms to exclude fields from display when browsing log records in the ISPF dialog.

In the log browser, when you are browsing an individual record:

1. Enter FORM in the **Format** field (near the top-right corner of the panel).
2. Choose a form:
   - To explicitly specify a form, tab to the **Form** field, and either enter the name of a form, or press the Prompt function key (F4) to select from a list of forms for records of the displayed log type and code.
   - To use the form that the active filter specifies for records of the displayed log type and code, enter / (slash) next to **Use Form in Filter**. If the active filter does not specify a form for these records, the **Use Form in Filter** option has no effect.

To stop using the form, either:

- Clear the value of the **Form** field or the **Use Form in Filter** option, and then press Enter.
- Enter STD in the **Format** field.

Related tasks

Formatting log records
You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

Using forms to exclude fields when creating formatted record reports

When using the report and extract utility to create a formatted record report, you can optionally use forms to exclude fields. Formatted record reports reproduce the formatting that the ISPF dialog displays when you browse an individual record in the log browser.

When using the report and extract utility to create a formatted record report, you can, optionally, apply forms in either or both of the following ways:

- Specify the **FILTER** parameter of the **REPORT** command:

  ```
  REPORT FILTER(filter)
  ```

  Specify the name of a filter that refers to the forms you want to use for your formatted record report.

  Tip: To use the ISPF dialog to generate JCL for a formatted record report that uses the **FILTER** parameter:

  1. On the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.
  2. Enter SUB next to the log file that you want to process.
  3. Select a filter that refers to the forms you want to use for your formatted record report.
  4. Select the **Report** option.
  5. Select the **Form** option.

- Specify one or more **CODE** commands (one for each type of log record that you want to report) following the **REPORT** command:

```
CODE … FORM(form)
```

Related tasks

Formatting log records

You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

Related reference

REPORT command

Requests a report.

Using forms to specify the columns in CSV files

When using the ISPF dialog to create a comma-separated values (CSV) file, you must specify a form name. The form identifies the type of log record and which fields to write to the CSV file.

Related tasks

Creating CSV files from log files in your ad hoc list

You can use Transaction Analysis Workbench to save data from any supported log record type to a comma-separated values (CSV) file. You can use the CSV file with other applications, such as PC-based spreadsheet applications. A CSV file created by Transaction Analysis Workbench contains selected fields from a single log record type and code.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Chapter 72. CICS-DBCTL reports

CICS-DBCTL reports process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.

The description of each report shows how the report columns correspond to, or are calculated from, fields in the original records.

For more information about the contents of these fields, see the CICS and IMS documentation.

The report descriptions use the following notation to refer to fields in CMF records:

group, *field ID*, *field name*

For example:

DFHTASK, C001, TRAN

Related concepts

Reporting
Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

CICS-DBCTL transactions
You can analyze response time problems in CICS-DBCTL transactions using CICS monitoring facility (CMF) performance class records (SMF type 110 records), IMS log records, or a combined view of both.

Related tasks

Creating CICS-DBCTL reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

CICS-DBCTL report JCL

To create CICS-DBCTL reports, use the **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands of the report and extract utility. The multi-step JCL presented here generates all CICS-DBCTL reports.

Example

```
//UIDFUW   JOB NOTIFY=&SYSUID
//*
//* 1: Report and Extract exception CICS-DBCTL transactions (in CMF)
//*
//S1       EXEC PGM=FUWBATCH,COND=(0,NE)
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//CICSIMS  DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=HLQ.SMF.DAILY
//EXTRACT1 DD  DISP=SHR,DSN=&SYSUID..FUW.CMF.EXTRACT
//SORTCTIM DD  DSN=&&SC,DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,1)
//SYSIN    DD  *
PAGELIM(999999)
LINECNT(0)
ZONE(LOCAL)
REPORT CICS-DBCTL
PARM(LIST=S1L,SUMM=S1S,EXTRACT=EXTRACT1,DBCTLONLY,ABEND,RESPONSE>2.0)
/*
//*
//* 2: Sort the exception CICS transactions into START time sequence
//*
//S2       EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=&SYSUID..FUW.CMF.EXTRACT
//SORTOUT  DD  DISP=SHR,DSN=&SYSUID..FUW.CICS.TRANINDX
//SORTWK01 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK02 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK03 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK04 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
```

```
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DSN=&&SC,DISP=(OLD,DELETE)
//*
//* 3: Report and Extract exception IMS DBCTL threads (in IMS log)
//*
//S3       EXEC PGM=FUWBATCH,PARM=V111
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//IMSDBCTL DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=HLQ.IMS.SLDS
//EXTRACT1 DD  DISP=SHR,DSN=&SYSUID..FUW.IMS.EXTRACT
//SORTITIM DD  DSN=&&SI,DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,1)
//SYSIN    DD  *
PAGELIM(999999)
LINECNT(0)
ZONE(LOCAL)
REPORT IMS-DBCTL
PARM(LIST=S3L,SUMM=S3S,EXTRACT=EXTRACT1,FAIL,PROCESS>1.0,ROK)
/*
//*
//* 4: Sort the exception IMS DBCTL threads into START time sequence
//*
//S4       EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=&SYSUID..FUW.IMS.EXTRACT
//SORTOUT  DD  DISP=SHR,DSN=&SYSUID..FUW.IMS.TRANINDX
//SORTWK01 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK02 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK03 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK04 DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  DSN=&&SI,DISP=(OLD,DELETE)
//*
//* 5: Combined CICS and IMS DBCTL exception analysis
//*
//S5       EXEC PGM=FUWBATCH,PARM=V111
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DBCTLEXC DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=&SYSUID..FUW.CICS.TRANINDX
//LOGIN    DD  DISP=SHR,DSN=&SYSUID..FUW.IMS.TRANINDX
//SYSIN    DD  *
PAGELIM(999999)
LINECNT(0)
ZONE(LOCAL)
REPORT CICS-DBCTL
PARM(COMBINED=S5C)
/*
```

Description

Step S1

Uses the CMF records (in the input SMF file, ddname SMFIN) that are for CICS-DBCTL transactions, and that either abended or had a response time of greater than two seconds, to create the following outputs:

- CICS-DBCTL list report (in output data set S1L)
- CICS-DBCTL summary report (S1S)
- CMF extract, containing the selected CMF records

Also creates a **SORT** statement in the temporary data set &&SC (used in the next step).

Step S2

Uses the **SORT** statement in the temporary data set &&SC (created by the previous step) to sort the CMF extract by CICS transaction start time. The sorted output data set is known as a CICS transaction index (used in step S5).

Step S3

Uses the IMS log records (in the input IMS log file, ddname LOGIN) that are for DBCTL threads, and that either abended, generated a Fast Path failure (type 5938 log record), or had a process time of greater than one second, to create the following outputs:

- IMS-DBCTL list report (in output data set S2L)

- IMS-DBCTL summary report (S2S)
- IMS log extract, containing IMS transaction index records (type CA01 user log record)

Also creates a **SORT** statement in the temporary data set &&SI (used in the next step).

Note: The one-second IMS process time threshold specified in this step is deliberately less than the two-second CICS response time threshold specified in the corresponding earlier step (S1) for CMF records. There is no drawback to creating a larger IMS transaction index than might otherwise be necessary. Combined CICS and IMS reporting in step S5 will still match successfully; unmatched IMS threads will not be used in the combined report.

Step S4

Uses the **SORT** statement in the temporary data set &&SI (created by the previous step) to sort the IMS log extract by IMS transaction start time. The sorted output data set is known as a IMS transaction index (used in step S5).

Step S5

Uses the CICS transaction index (ddname SMFIN) and IMS transaction index (ddname LOGIN) created by earlier steps to create a CICS-DBCTL combined summary report.

In this example, we have deliberately not specified the parameter **MATCHONLY** in the **PARM** command. Omitting **MATCHONLY** means that the CICS-DBCTL combined summary report might contain data from CMF records (in the CICS transaction index) for which there is no matching IMS data (in the IMS transaction index).

Related reference

REPORT CICS-DBCTL command: CICS reports
Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

Report and extract utility
The report and extract utility processes logs to create reports and extracts; convert logs to comma-separated values (CSV) or or JavaScript Object Notation (JSON); runs REXX execs that process log files; and exports or imports controls (filters, forms, and object lists) between control repositories.

REPORT IMS-DBCTL command: IMS reporting
Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

REPORT CICS-DBCTL command: combined CICS and IMS reporting
Requests a report that combines output from earlier **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands to show CICS DBCTL transaction response times across CICS and IMS.

CICS-DBCTL list report

The CICS-DBCTL list report uses data in CMF performance class (SMF type 110) records to show the performance characteristics of individual instances of CICS transactions, with a focus on DBCTL.

Example

```
V1R3M0          2011-04-06 Wednesday          CICS-DBCTL List                          Page   1

Tran  Task Num  Start Time       Response  CPU Time  IMS Reqs  IMS Wait  ABEND  APPLID  RMUOWID
----  --------  ---------------  --------  --------  --------  --------  -----  ------  ----------------
ORDR       113  08.39.14.241959  1.038349  0.005241        37  0.018037         CICSP1  C79458194C25C642
BANK       128  08.41.00.041884  1.755926  0.005499        37  1.456115         CICSP1  C794587E32358820
ORDR       170  08.47.22.063694  14.06754  0.026473        29  8.777141  ADCD   CICSP1  C79459EA853EFB03
X
BANK       168  08.47.14.739730  2.517171  0.006078        37  2.115206         CICSP1  C79459E3892DA202
X
```

Description

Column heading	Description	Field or formula
Tran	CICS transaction ID	DFHTASK, C001, TRAN
Task Num	CICS task number	DFHTASK, P031, TRANNUM
Start Time	Transaction start time	DFHCICS, T005, START
Response	Response time	(DFHCICS, T006, STOP) - (DFHCICS, T005, STOP)
CPU time	CPU time	DFHTASK, S008, USRCPUT
IMS Reqs	Total count of EXEC DLI requests	DFHDATA, A179, IMSREQCT
IMS Wait	Total time taken to process IMS requests	**When DBCTL is not threadsafe** DFHDATA, S186, IMSWAIT **When DBCTL is threadsafe** When DBCTL is threadsafe, the IMSWAIT clock field is not used because it is not recorded by CICS; the field is always zero. Instead, IMS Wait is calculated from the sum of the following two clock fields: DFHRMI, S005, RMIEXDLI DFHRMI S004, RMIDBCTL The RMI fields are optional in the CMF record. To collect these fields, you must specify RMI=YES in your MCT.
ABEND	Abend code	DFHPROG, C114, ABCODEC
APPLID	CICS generic APPLID	SMFMNPRN
RMUOWID	Recovery Manager unit-of-work ID	DFHTASK, C132, RMUOWID

The combination of APPLID and RMUOWID form the IMS recovery token that is passed to and used by IMS to uniquely identify the thread. You can use this token to match a CICS transaction to its associated IMS thread.

An X in the last column indicates that the transaction is an exception; it meets the exception criteria specified by the **ABEND** and **RESPONSE>** parameters of the **REPORT** command.

Related reference

REPORT CICS-DBCTL command: CICS reports

Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

CICS-DBCTL summary report

The CICS-DBCTL summary report uses data in CMF performance class (SMF type 110) records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

Example

```
V1R3M0          2011-04-06 Wednesday          CICS-DBCTL Summary                          Page   4

Tran  APPLID     CMF Count  Response  CPU Time  IMS Reqs  IMS Wait     ABEND  Rate/Sec  Exceptions
----  --------  ----------  --------  --------  --------  --------  --------  --------  ----------
DBEU  CICSP1            60  11.12982  0.008967        35  4.256977        10         0          60
ORDR  CICSP1            58  10.79668  0.008468        34  4.200507        10         0          58
```

Description

Column heading	Description	Field or formula
Tran	CICS transaction ID.	DFHTASK, C001, TRAN
APPLID	CICS generic APPLID.	SMFMNPRN
CMF Count	The number of CMF performance class records processed, typically equal to the number of transactions that ran in the CICS system.	
Response	Average response time.	DFHCICS, T006, STOP-START
CPU time	Average CPU time.	DFHTASK, S008, USRCPUT
IMS Reqs	Average count of EXEC DLI requests.	DFHDATA, A179, IMSREQCT
IMS Wait	Average elapsed time taken to process the IMS requests.	**When DBCTL is not threadsafe** DFHDATA, S186, IMSWAIT **When DBCTL is threadsafe** When DBCTL is threadsafe, the IMSWAIT clock field is not used because it is not recorded by CICS; the field is always zero. Instead, IMS Wait is calculated from the sum of the following two clock fields: DFHRMI, S005, RMIEXDLI DFHRMI S004, RMIDBCTL The RMI fields are optional in the CMF record. To collect these fields, you must specify RMI=YES in your MCT.
ABEND	The number of times the transaction abended.	DFHPROG, C114, ABCODEC
Rate/Sec	The number of transactions processed per second, averaged over the reporting period.	

Column heading	Description	Field or formula
Exceptions	The number of transactions that were identified as exceptions; that met the exception criteria specified by the **ABEND** and **RESPONSE>** parameters of the **REPORT CICS-DBCTL** command. If no exception criteria were specified then this column is omitted.	

Related reference

REPORT CICS-DBCTL command: CICS reports
Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

IMS-DBCTL list report

The IMS-DBCTL list report uses data in IMS log records to show the performance characteristics of individual instances of DBCTL threads started by CICS transactions.

Example

```
V1R3M0          2011-04-06 Wednesday           IMS-DBCTL List                          Page   1
      Task
Tran   Num  PSB name  Start Time        Elapsed  CPU Time  FF Get  FF Upd  FP Get  FP Upd  ABEND  APPLID   RMUOWID
----  ----  --------  ---------------  --------  --------  ------  ------  ------  ------  -----  --------
--------------
ORDR   113  DFHTWM04  08.47.00.027978  7.346773  0.004921      14      21       6       9         CICSP1
C79459D5810D6A23
DBEU   128  DFHTWM04  08.47.07.385608  7.337655  0.004453      14      21       8      12         CICSP1
C79459DC85773221
ORDR   170  DFHTWM04  08.47.22.064699  13.98990  0.003550      11      16       4       6  U0777  CICSP1
C79459EA853EFB03 X
DBEU   168  DFHTWM04  08.47.14.741096  22.51435  0.004661      14      21       8      12         CICSP1
C79459E3892DA202 X
```

Description

Column name	Description
Tran	CICS transaction ID 1
Task Num	CICS task number 1
PSB name	Program specification block (PSB) name
Start Time	IMS thread start time (end of scheduling)
Elapsed	The elapsed time that the PSB is scheduled by the CICS transaction; from EXEC DLI SCHED to the end of syncpoint processing
CPU Time	The CPU time consumed by the IMS thread to process the DLI calls
FF Get	The count of full-function database get DLI calls issued by the transaction
FF Upd	The count of full-function database update DLI calls issued by the transaction
FP Get	The count of Fast Path database get DLI calls issued by the transaction
FP Upd	The count of Fast Path database DLI calls issued by the transaction
ABEND	Abend code or Fast Path failure reason code (SF=*nn*)
APPLID	CICS generic APPLID
RMUOWID	Recovery Manager unit-of-work ID

Note:

1
The CICS transaction ID and task number will only be reported for IMS version 11 or higher.

The combination of APPLID and RMUOWID form the IMS recovery token that is passed to and used by IMS to uniquely identify the thread. You can use this token to match a CICS transaction to its associated IMS thread.

An X in the last column indicates that the transaction is an exception; it meets the exception criteria specified by the **ABEND** and **RESPONSE>** parameters of the **REPORT** command.

Related reference

REPORT IMS-DBCTL command: IMS reporting
Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

IMS-DBCTL summary report

The IMS-DBCTL summary report uses data in IMS log records to show the performance characteristics of DBCTL threads started by, summarized by CICS transaction ID, program, and APPLID.

Example

```
V1R3M0          2011-04-06 Wednesday           IMS-DBCTL Summary                        Page   3

Tran Program  APPLID     08 Count   Elapsed  CPU Time  StaDelay  Schedule   IC Wait   PS Wait   ABEND  Rate/Sec
Exceptions
---- -------- --------   --------  --------  --------  --------  --------   -------   -------   -----  --------
----------
DBEU DFHTWM04 CICSP1           42  10.94999  0.004092  0.001133  0.000183         0         0       7         0
42

                          07 Count   DB call   DB Gets   DB Upds  IO Count   IO Time  LockWait
                          --------  --------  --------  --------  --------  --------  --------
                                41        33        13        19         4  0.003438  3.980170

                          FP Count   FP call   FP Gets   FP Upds   FP Wait   FP Fail
                          --------  --------  --------  --------  --------  --------
                                41        19         7        11         0         7

                                    Synctime   Phase 1   Phase 2    FP PH2   OTHREAD
                                    --------  --------  --------  --------  --------
                                    0.011938  0.006555  0.005383  0.002232  0.017659
```

Description

Column heading	Description
Tran	CICS transaction ID.
Program	IMS PSB name.
APPLID	CICS generic APPLID.
08 Count	The number of IMS threads that were scheduled. An IMS DBCTL thread starts in the IMS log with a type 08 log record.
Elapsed	The elapsed time that the PSB is scheduled by the CICS transaction; from EXEC DLI SCHED to the end of syncpoint processing.
CPU time	The CPU time consumed by the IMS thread to process the DLI calls.

Column heading	Description
StaDelay	An estimate only, “Start Delay” is the elapsed time it takes for the CICS transaction to start processing and successfully schedule the IMS PSB. This elapsed time is derived by subtracting two STCK values: 1. The last 8 bytes of the 16 byte IMS recovery token. This STCK value is the RMUOWID that is assigned by CICS at the start of the transaction. It allows CICS to coordinate with external resource managers such as IMS. 2. The STCK time stamp in the suffix of the type 08 log record. This time represents the end of the successful scheduling of the PSB. STADELAY = 2 – 1
Schedule	The elapsed time for IMS to complete scheduling of the PSB.
IC Wait	The elapsed time delay in PSB scheduling due to Intent Conflict.
PS Wait	The elapsed time delay in PSB scheduling due to Pool Space shortage.
ABEND	The number of times the transaction abended.
Rate/Sec	The number of transactions processed per second, averaged over the reporting period.
Exceptions	The number of transactions that were identified as exceptions.
07 Count	The number of IMS threads that completed processing. An IMS DBCTL thread terminates in the IMS log with a type 07 log record. **Note:** CPU time and full-function database call statistics are not available unless the thread terminates and the type 07 record is available.
FF Call	The count of full-function database DLI calls issued by the transaction (Get + Update).
FF Get	The count of full-function database get DLI calls issued by the transaction.
FF Upd	The count of full-function database update DLI calls issued by the transaction.
IO Count	The count of full-function database I/O calls issued on behalf of the transaction.
IO Time	The elapsed time of full-function database I/O calls issued on behalf of the transaction.
LockWait	The elapsed time delay in transaction processing due to waits for full-function database locks to be obtained due to contention.
FP Count	The number of IMS threads that used Fast Path databases. Fast Path statistics are recorded in the type 5937/38 log record at syncpoint time.
FP Call	The count of Fast Path database DLI calls issued by the transaction (Get + Update).
FP Get	The count of Fast Path database get DLI calls issued by the transaction.
FP Upd	The count of Fast Path database update DLI calls issued by the transaction.
FP Wait	The count of Fast Path waits that delayed transaction processing; including waits for DEDB buffers, as well as CI and UOW locks.
FP Fail	The number of IMS threads that failed Fast Path processing; represented in the IMS log by the type 5938 log record (rather than the type 5937 for a successful completion).
SyncTime	Syncpoint processing elapsed time.

Column heading	Description
Phase 1	Phase 1 syncpoint processing elapsed time.
Phase 2	Phase 2 syncpoint processing elapsed time.
FP PH2	The part of phase 2 syncpoint processing elapsed time that is attributable to Fast Path.
OTHREAD	The elapsed time for Fast Path OTHREAD processing to complete. Phase 2 syncpoint processing will queue updated DEDB buffers to OTHREAD for IO processing. This is an asynchronous process that may complete before or after the transaction ends. OTHREAD elapsed time provides an indication as to whether OTHREAD is a system bottleneck.

Related reference

REPORT IMS-DBCTL command: IMS reporting
Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

CICS-DBCTL combined summary report

The CICS-DBCTL combined summary report combines data from CMF performance class (SMF type 110) records with data from IMS log records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

Example

```
V1R3M0     2012-03-28 Wednesday     CICS-DBCTL Summary                      Page   1

Tran  APPLID   CMF Count  Response  CPU Time  IMS Reqs  IMS Wait     ABEND  Rate/Sec  CICS
----  -------- ---------  --------  --------  --------  --------  --------  --------
BANK  CICSP1          60  11.12982  0.008967        35  4.256977        10         0

                08 Count   Elapsed  CPU Time  StaDelay  Schedule   IC Wait   PS Wait  IMS
                ---------  --------  --------  --------  --------  --------  --------
                       42  10.94999  0.004092  0.011668  0.000183         0         0

                07 Count   DB call   DB Gets   DB Upds  IO Count   IO Time  LockWait
                ---------  --------  --------  --------  --------  --------  --------
                       41        33        13        19         4  0.003438  3.980170

                FP Count   FP call   FP Gets   FP Upds   FP Wait   FP Fail
                ---------  --------  --------  --------  --------  --------
                       41        19         7        11         0         7

                           Synctime   Phase 1   Phase 2    FP PH2   OTHREAD
                           --------  --------  --------  --------  --------
                           0.011938  0.006555  0.005383  0.002232  0.017659
```

Description

This report combines the layouts of two other reports: the CICS-DBCTL summary report (that uses data from CMF records) and the IMS-DBCTL summary report (that uses data from IMS log records).

The first section of this report (marked CICS in the example) matches the layout of the CICS-DBCTL summary report.

The remaining four sections (starting from the section marked IMS in the example) match the layout of the IMS-DBCTL summary report.

Interpretation

- The combination of these two reporting perspectives helps you to analyze the behavior of CICS-DBCTL transactions across CICS and IMS. For example:
  - The `IMS Wait` column in the CICS section of the report shows how long CICS waited for IMS to respond. Various columns in the IMS sections of the report, such as `LockWait`, provide details of the components of `IMS Wait`, helping you to more accurately diagnose long `IMS Wait` times.
  - The `IMS Reqs` column in the CICS section of the report shows the number of IMS requests. Various columns in the IMS sections of the report, such as `DBGets`, provide details of the types of IMS request, helping you to understand the type of activity performed by the CICS-DBCTL transaction.
- If this report is created using the parameter **MATCHONLY**, then `CMF Count` will always match `08 Count`.
- If this report is created without **MATCHONLY**, then matching `CMF Count` and `08 Count` values might indicate a problem in IMS rather than CICS. Understanding why requires an understanding of how the report is created:
  - Transaction Analysis Workbench creates the CICS-DBCTL combined summary report from two input files: a CICS transaction index (created from CMF records for CICS-DBCTL transactions) and an IMS transaction index (created from IMS log records for DBCTL threads). `CMF Count` is the number of records in the CICS transaction index. `08 Count` corresponds to the number of records in the IMS transaction index that match the records in the CICS transaction index.
  - When creating a CICS-DBCTL combined summary report, Transaction Analysis Workbench begins with the CICS transaction index. For each record in the CICS transaction index, Transaction Analysis Workbench attempts to find a matching record in the IMS transaction index (that is, for the corresponding DBCTL thread).
  - Suppose that the CICS transaction index and IMS transaction index contain *exceptions* only, not transactions that completed normally. That is, the CICS transaction index is based on CMF records that match specified exception criteria (such as an abend), and the IMS transaction index is based on IMS log records that match similar exception criteria.
  - If `CMF Count` matches `08 Count`, it means that, for every record in the CICS transaction index, Transaction Analysis Workbench found a matching record in the IMS transaction index. For every CICS-DBCTL transaction that had a problem, there was a problem in IMS.
  - If a DBCTL thread for a CICS-DBCTL transaction had completely normally, without problems, it would not have qualified as an exception; it would not be represented in the IMS transaction index, and the `08 Count` would be lower. If, instead of matching, `CMF Count` had been greater than `08 Count`, we could have concluded that a problem was occurring in CICS, but not (always) in IMS. (Note that, because Transaction Analysis Workbench begins with the CICS transaction index, and then looks for matching records in the IMS transaction index, `CMF Count` will never be less than `08 Count`.
  - The **REPORT CICS-DBCTL** command that created the CMF extract selected only and IMS extracts that, when sorted (becoming the CICS transaction index and IMS transaction index),

Related reference

REPORT CICS-DBCTL command: combined CICS and IMS reporting
Requests a report that combines output from earlier **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands to show CICS DBCTL transaction response times across CICS and IMS.

REPORT CICS-DBCTL command: CICS reports
Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

REPORT IMS-DBCTL command: IMS reporting
Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

Chapter 73. DB2 accounting exception reports and extracts

DB2 accounting exception reports and extracts are generated from DB2 accounting (SMF type 101) records. DB2 accounting records from the specified input files are checked against optional *filtering criteria* for the DB2 subsystem ID (SSID), plan, and connection type. Records that match the filtering criteria are checked against *exception criteria* that specify thresholds for various performance-related values, such as CPU time.

Only records that meet or exceed a threshold are included in the reports and extracts.

Each record can trigger multiple exceptions. The report lists every exception.

The SYSPRINT output data set contains FUW0061I messages that recap the type and number of exceptions.

The extract is also known as a *DB2 accounting index*.

Example report

```
V1R3M0          2013-05-31 Friday                 DB2 Exception List                          Page
1


SSID Correlation  Connect  Plan     Auth id  Time             Exception               Threshold
LUWID
---- ------------ -------- -------- -------- ---------------  ----------------------- ---------
----------------------------
DBA6 POOLFB660002 FUWTCIC  FBODCP06 TWM      16:26:21.950580  Response       2.230828      0.01  FTS3/DBA6LU/
CB70FA74C5AE/0002
DBA6 POOLFB660002 FUWTCIC  FBODCP06 TWM      16:26:21.950580  Class 1 CPU   15.503411      0.02  FTS3/DBA6LU/
CB70FA74C5AE/0002
DBA6 POOLFB660002 FUWTCIC  FBODCP06 TWM      16:26:21.950580  Class 2 CPU   16.695062      0.03  FTS3/DBA6LU/
CB70FA74C5AE/0002
DBA6 POOLFB660002 FUWTCIC  FBODCP06 TWM      16:26:21.950580  Log records           8         0  FTS3/DBA6LU/
CB70FA74C5AE/0002
```

Description

Column heading	Description	Source field or input
SSID	DB2 subsystem ID.	QWHSSSID
Correlation	Correlation ID.	QWHCCV
Connect	Connection name. If the connection name is not available, this column reports the connecting system type.	Connection name: QWHCCN Connecting system type: QWHCATYP
Plan	Plan name.	QWHCPLAN
Auth id	Authorization ID.	QWHCAID
Time	End of accounting interval time stamp. This is not the thread start time. This time matches the record time stamp displayed in the ISPF dialog.	QWHSSTCK
Exception	The exception event that was triggered: its description and value.	Various

Column heading	Description	Source field or input
Threshold	The exception threshold that was exceeded.	**PARM** command following the **REPORT DB2X** command
LUWID	Logical unit of work ID that uniquely identifies the DB2 accounting record.	QWHSLWID

Related concepts

Reporting
Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

Related tasks

Creating DB2 accounting exception reports and extracts
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports and extracts of DB2 accounting records that match the exception criteria that you specify.

Related reference

REPORT DB2X command: DB2 accounting exception report and extract
Requests a report or an extract, or both, of DB2 accounting (SMF type 101) records that match the specified exception criteria.

SMF type 101: DB2 Thread Accounting Summary report
The DB2 Thread Accounting Summary report uses SMF type 101 records to show accounting information for DB2 connections.

DB2 accounting exception report and extract JCL

To create DB2 accounting exception reports and extracts, use the **REPORT DB2X** command of the report and extract utility.

Example

The following JCL generates a report and an extract.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//EXTRACT  DD DSN=MY.FUW.EXTRACT,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0-09.00
STOP  0-16.00
REPORT DB2X
CODE(DTR:003)
  COND SM101SSI IN (DB2*)
  COND QWHCPLAN IN (ORDER* STOCK BANKING)
PARM(
  LIST
  EXTRACT
  CICS
  RESPONSE>0.5
  CPU1>0.1
)
/*
```

In this example, the report and extract batch utility program, FUWBATCH, reads input log records from the SMF log stream that is identified by the **LOGSTREAM** command in the SYSIN data set.

The **START** and **STOP** commands specify a relative date value of 0, meaning today, with a start time of 9 a.m. and a stop time of 4 p.m.

The commands that follow the **REPORT DB2X** command qualify the behavior of the **REPORT DB2X** command.

The **CODE** command specifies filtering criteria. **CODE(DTR:003)** restricts the input log records to DB2 accounting records (log type DTR, log code 003).

The **COND** statements add conditions to the filtering criteria specified by the preceding **CODE** command:

- The first **COND** statement restricts the DB2 accounting field that contains the DB2 subsystem ID, SM101SSI, to values that start with "DB2".
- The second **COND** statement restricts the DB2 accounting field that contains the plan name, QWHCPLAN, to any of the following values: values that start with "ORDER", the value "STOCK", or the value "BANKING".

The **PARM** command adds a third condition to the filtering criteria: the **CICS** parameter restricts the connection type specified in the input log records to CICS connections.

The three filter conditions are combined by a logical AND. Only the DB2 accounting records that meet all of the conditions are selected for further processing.

The selected records are checked against the exception criteria specified by the **PARM** command. In this example, records are exceptions if they meet any of the following conditions:

- Response time is equal to or greater than one second
- CPU class 1 time is equal to or greater than 0.1 seconds

The **LIST** parameter of the **PARM** command requests a report. The report is written to the ddname DB2X, which is dynamically allocated to SYSOUT if not explicitly specified.

The **EXTRACT** parameter of the **PARM** command requests a report. The extract is written to the ddname EXTRACT.

Related reference

REPORT DB2X command: DB2 accounting exception report and extract
Requests a report or an extract, or both, of DB2 accounting (SMF type 101) records that match the specified exception criteria.

Report and extract utility
The report and extract utility processes logs to create reports and extracts; convert logs to comma-separated values (CSV) or or JavaScript Object Notation (JSON); runs REXX execs that process log files; and exports or imports controls (filters, forms, and object lists) between control repositories.

DB2 accounting exception criteria

Exception criteria specify thresholds for various performance-related accounting values, such as CPU time. Only records that trigger an exception (match the exception criteria) are included in the DB2 accounting exception extracts and reports.

A record triggers an exception if the value in the record is *greater than or equal to* the corresponding threshold value. There are two special cases:

- If the threshold value is zero (0), any value *greater than zero* triggers an exception.
- For exception criteria based on conditions: if the threshold value is one (1), an exception is triggered if the condition is true.

Threshold values for CPU and elapsed time are specified in seconds, from 1 microsecond to 99 seconds. For example:

Threshold value	A record triggers an exception if its value is...
0.000005	5 microseconds or greater
2	2 seconds or greater
2.0	2 seconds or greater (same as 2)
0	Greater than zero

Threshold values for counts are specified as integers, 0 - 999999. For example:

Threshold value	A record triggers an exception if its value is...
1000	1000 or greater
0	Greater than zero

DB2 accounting records can contain accumulated activity for multiple threads. Such records are known as *rollup* accounting records. The **Apportion rollup** option in the ISPF dialog, and the equivalent **APPORTIONROLLUP** parameter of the **REPORT DB2X** batch command, control how exception criteria are evaluated for rollup accounting records:

- If the **Apportion rollup** option is selected, accounting values are divided by the rollup count before comparison with threshold values; the values are apportioned equally among all of the threads.

  Response (thread elapsed) time is treated differently because it is recorded for each thread being rolled up; the value for each thread is checked individually.
- If the **Apportion rollup** option is not selected, accounting values are compared directly with threshold values; the rollup is treated as a single transaction.

Exception criteria are divided into the following categories.

Response and CPU time

These exception criteria are based on a time value. Allowable threshold values:

- 0.000001 - 99 seconds
- 0 (any value greater than zero triggers an exception)

Table 21. DB2 accounting exception criteria for response and CPU time

Exception criteria	Keyword in PARM command following REPORT DB2X command	Threshold field or formula
Response time	**RESPONSE**	If rollup is not apportioned: QWACESC - QWACBSC If rollup is apportioned: QWARESC - QWARBSC
CPU class 1	**CPU1**	The sum of the following components: • Central processor (CP) times: Non-nested: QWACEJST - QWACBJST Stored procedure: QWACSPCP + QWACSPNF_CP UDF: QWACUDCP + QWACUDFNF_CP Trigger: QWACTRTT + QWACTRTE • Specialty engine (SE) times: Non-nested: QWACCLS1_zIIP Stored procedure: QWACSPNF_zIIP + QWACSP_CLS1se UDF: QWACUDFNF_zIIP + QWACUDF_CLS1se Trigger: QWACTRTT_zIIP + QWACTRTE_se

Table 21. DB2 accounting exception criteria for response and CPU time (continued)		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
In-DB2 elapsed	**INDB2**	The sum of the following components: Non-nested: QWACASC Stored procedure: QWACSPEB + QWACSPNF_ELAP UDF: QWACUDEB + QWACUDFNF_ELAP Trigger: QWACTRET+ QWACTREE
CPU class 2	**CPU2**	The sum of the following components: • Central processor (CP) times: Non-nested: QWACAJST Stored procedure: QWACSPTT + QWACSPNF_CP UDF: QWACUDTT + QWACUDFNF_CP Trigger: QWACTRTT + QWACTRTE • Specialty engine (SE) times: Non-nested: QWACCLS2_zIIP Stored procedure: QWACSPNF_zIIP + QWACSP_CLS2se UDF: QWACUDFNF_zIIP + QWACUDF_CLS2se Trigger: QWACTRTT_zIIP + QWACTRTE_se
Database I/O	**DBIO**	QWACAWTI
Lock suspend	**LOCKSUSP**	QWACAWTL + QWACAWTJ

Stored procedure

These exception criteria are based on a time value. Allowable threshold values:

- 0.000001 - 99 seconds
- 0 (any value greater than zero triggers an exception)

Table 22. DB2 accounting exception criteria for stored procedures		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Elapsed	**SPELAP**	QWACSPNF_ELAP
CPU	**SPCPU**	QWACSPNF_CP

Row activity

These exception criteria are based on a count value. Allowable threshold values: 0 - 999999.

Table 23. DB2 accounting exception criteria for row activity		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Fetched	**FETCH**	QXRWSFETCHD
Inserted	**INSERT**	QXRWSINSRTD
Updated	**UPDATE**	QXRWSUPDTD
Deleted	**DELETE**	QXRWSDELETD

Locking

These exception criteria are based on a count value. Allowable threshold values: 0 - 999999.

Table 24. DB2 accounting exception criteria for locking		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Deadlocks	**DEADLOCK**	QTXADEA
Suspends	**SUSPEND**	QTXASLOC
Timeouts	**TIMEOUT**	QTXATIM
Lock requests	**LOCKREQ**	QTXALOCK

Buffering

These exception criteria are based on a count value. Allowable threshold values: 0 - 999999.

Table 25. DB2 accounting exception criteria for locking		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Get pages	**GETPAGE**	QBACGET
Update pages	**UPDPAGE**	QBACSWS

Logging

These exception criteria are based on a count value. Allowable threshold values: 0 - 999999.

Table 26. DB2 accounting exception criteria for locking		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Log records	**LOGRECS**	QWACLRN

Abnormal conditions

These exception criteria are based on a condition. Allowable threshold value: 1 (a record is an exception if the condition is true).

Table 27. DB2 accounting exception criteria for locking		
Exception criteria	**Keyword in PARM command following REPORT DB2X command**	**Threshold field or formula**
Abort	**ABORT**	QWACABRT
Check pending	**CHKP**	QXPRRESI

Related concepts

Exception analysis

Transaction Analysis Workbench exception analysis identifies log records that meet your exception criteria, such as long response time or abnormal conditions, and creates a transaction index of those log records for further analysis.

Related reference

REPORT DB2X command: DB2 accounting exception report and extract

Requests a report or an extract, or both, of DB2 accounting (SMF type 101) records that match the specified exception criteria.

Chapter 74. SMF reports

Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

The description of each report shows how the report columns correspond to, or are calculated from, fields in the SMF records. For more information about the contents of these fields, see the documentation for each SMF record type provided with z/OS or the relevant subsystem, such as DB2.

To use the Transaction Analysis Workbench dialog to generate the JCL for these reports, see "Creating SMF reports" on page 126.

Related concepts

Reporting
Transaction Analysis Workbench combines its own reporting with other products to help investigate multiple facets of a problem.

Report and extract utility
The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Related tasks

Creating SMF reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of SMF records.

Related reference

REPORT command for SMF reports
Requests either an SMF Recap report that gives an overview of the contents of an SMF file or, for some SMF record types only, a report that processes a particular type of SMF record; these reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

SMF codes
The SMF log type consists of log codes that identify MVS System Management Facilities records.

SMF Recap report
The SMF Recap report provides a high-level overview of the contents of an SMF file, such as which systems and subsystems wrote records to that file, and which record types the file contains.

SMF report JCL

To create an SMF report, use the **SMF** parameter of the **REPORT** command of the report and extract utility.

The following figure shows the format of the JCL to create an SMF report from a dumped SMF data set.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//SMFRPT    EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN001 DD  DSN=<dumped SMF data set>,
//             DISP=SHR
//SYSUDUMP DD  SYSOUT=*
//<rpt dd> DD  SYSOUT=*
//SYSIN    DD  *
REPORT SMF(<log code>) OUTPUT(<rpt dd>)
⋮
/*
```

Figure 118. JCL to create an SMF report from a dumped SMF data set

The following figure shows the format of the JCL to create an SMF report from an SMF log stream.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//SMFRPT    EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//<rpt dd> DD  SYSOUT=*
//SYSIN    DD  *
LOGSTREAM SMF:<log stream>
START <start date and time>
REPORT SMF(<log code>) OUTPUT(<rpt dd>)
⋮
/*
```

Figure 119. JCL to create an SMF report from an SMF log stream

<FUW HLQ>
The high-level qualifier of the Transaction Analysis Workbench libraries installed on your system. SFUWLINK is the default low-level qualifier of the executable load module library that contains the report and extract utility program, FUWBATCH.

<dumped SMF data set>
To process a dumped SMF data set instead of a log stream: specify an SMFIN001 (or just SMFIN) DD statement that refers to the dumped SMF data set. Do not specify a **LOGSTREAM** command in the SYSIN data set.

<log stream>
To process a log stream instead of a dumped SMF data set: in the SYSIN data set, before any **REPORT** commands, specify a **LOGSTREAM** command that refers to the log stream. SMF log stream names have the high-level qualifier IFASMF. For example:

```
LOGSTREAM SMF:IFASMF.PROD1
```

If you specify a **LOGSTREAM** command, the SMFIN001 (or SMFIN) DD statement is ignored.

<start date and time>
You can specify **START** and **STOP** commands to limit the report to a particular time interval.

START is required for reporting from log streams.

<rpt dd>
The ddname of the output data set where you want the report to be written.

If you specify a ddname in an **OUTPUT** parameter without a matching DD statement, the report goes to a sysout data set with that ddname, as if you had specified a DD statement with SYSOUT=*.

<log code>
The log code that identifies the SMF record type and, if applicable, subtype for the report. For example, `REPORT SMF(33-2)` requests the report for SMF record type 33, subtype 2.

Tip: The topic title for each report in this documentation contains the log code that you must specify to request that report. For example, to request the report described in the topic "SMF type 79-15: IRLM Long Lock Detection report" on page 449, you specify `REPORT SMF(79-15)`. These are the only log codes that are supported by the **SMF** parameter of the **REPORT** command.

To request multiple reports in a single batch job, add more **REPORT** commands to the SYSIN data set. Ensure that the **OUTPUT** parameter of each **REPORT** command refers to a unique ddname. For example:

```
⋮
//JOBSTATS DD  SYSOUT=*
//LOGGER   DD  SYSOUT=*
//SYSIN    DD  *
REPORT SMF(30)   OUTPUT(JOBSTATS)
REPORT SMF(88-1) OUTPUT(LOGGER)
⋮
```

Some SMF reports support an additional **REPORT** command parameter, **PARM**, to select one or more reports for the log code specified by the **SMF** parameter.

You can follow the **REPORT** command with **CODE** commands, and subordinate **COND** statements, that filter the log records for the report.

The following example creates a job statistics report, for yesterday, for jobs whose names begin with the characters CICS:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW      EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//JOBSTATS DD  SYSOUT=*
//SYSIN    DD  *
LOGSTREAM SMF:IFASMF.MVS1
START -1
STOP 0
REPORT SMF(30) OUTPUT(JOBSTATS)
  CODE(SMF:30-)
    COND SMF30JBN EQ 'CICS*'
/*
```

Figure 120. JCL to create an SMF report from an SMF log stream, with record filtering

Tip: Note the dash after the number 30 in `CODE(SMF:30-)`. The trailing dash identifies 30 as a decimal number; if you omit the dash, the **CODE** statement interprets its value as a hexadecimal number (log code). `CODE(SMF:1E)` and `CODE(SMF:30-)` are equivalent.

Related reference

Report and extract utility
The report and extract utility processes logs to create reports and extracts; convert logs to comma-separated values (CSV) or or JavaScript Object Notation (JSON); runs REXX execs that process log files; and exports or imports controls (filters, forms, and object lists) between control repositories.

SMF Recap report

The SMF Recap report provides a high-level overview of the contents of an SMF file, such as which systems and subsystems wrote records to that file, and which record types the file contains.

To request an SMF Recap report, you must manually edit JCL for the Transaction Analysis Workbench report and extract utility, and specify the following command:

```
REPORT SMF(RECAP) OUTPUT(SMFRECAP)
```

where SMFRECAP is the ddname of the output data set that will contain the report.

The SMF Recap report consists of up to five sections:

Systems
Lists the SMF system IDs (SIDs) in the file, including record count, time range, and whether records are in time sequence.

SMF record counts
Counts the number of each type of SMF record in the file.

CICS
Lists the CICS systems that have monitoring, CICS TS statistics, or CICS server statistics (type 110) records in the file.

DB2
Lists the DB2 subsystems that have statistics (type 100), accounting (101), or performance trace (102) records in the file.

IBM MQ
Lists the IBM MQ subsystems that have statistics (type 115) or accounting (116) records in the file.

The report contains CICS, DB2, and IBM MQ sections only if the SMF file contains the corresponding record types. For example, the report contains a CICS section only if the SMF file contains type 110 records.

Systems

```
V1R3M0          Transaction Analysis Workbench – Systems

SID        Count  Start Date Time           End Date Time         Time sequence
---- -----------  ---------- -----------  ---------- -----------  -------------
SYSA      625653  2011-06-24 15:55:00.01  2011-06-24 16:14:59.99  Yes
SYSB        2068  2011-06-24 15:55:00.95  2011-06-24 16:14:53.33  Yes
SYSC      515880  2011-06-24 15:55:00.05  2011-06-24 16:14:59.96  Yes
SYSD       16413  2011-06-24 15:55:09.35  2011-06-24 16:14:50.87  Yes
SYSE      511711  2011-06-24 15:55:00.00  2011-06-24 16:14:59.99  Yes
SYSF      522955  2011-06-24 15:55:00.06  2011-06-24 16:15:00.00  Yes
*TOT     2176881  2011-06-24 15:55:00.00  2011-06-24 16:15:00.00  Yes
```

Figure 121. SMF Recap report section: Systems

Column heading	Description
SID	SMF system ID
Count	The number of SMF records
Start	The time stamp of the oldest record
End	The time stamp of the youngest record
Time sequence	Whether the SMF records are in time sequence (Yes or No)

If the SMF file contains records from multiple systems, this report section contains a final *TOT line as a total for all of the systems.

Time sequence indicates how you can use Transaction Analysis Workbench to process this SMF file. To locate records by time and merge with other data sources, the Transaction Analysis Workbench ISPF dialog requires the records in an SMF file to be in time sequence. If the records in an SMF file are not in time sequence, you might wish to sort the file before using it with the ISPF dialog. Transaction Analysis Workbench batch reporting to process SMF files that are not in time sequence; however, if you want to perform batch reporting using an SMF file that is not in time sequence, and date/time range checking is required, you must use the **UNSORTED** batch command.

Sometimes, when the SMF file contains records from multiple systems, individual systems are in time sequence (time sequence is "Yes" for one or more systems) but the total file is not in time sequence; the time sequence for the final *TOT line is "No". This indicates that the original individual system files have been copied to this single master file one at a time, rather than merged in time sequence.

SMF record counts

```
V1R3M0     Transaction Analysis Workbench - SMF record counts

Type       Count  Description
---  -----------  --------------------------------
 30          392  Common Address Space Work
 64           54  VSAM Status
 70            6  RMF Processor Activity
 71            3  RMF Paging Activity
 72          117  RMF Workload Activity and Storage Data
 73            3  RMF Channel Path Activity
 74           36  RMF Activity of Several Resources
 75           21  RMF Page Data Set Activity
 78            6  RMF Virtual Storage and I/O Queuing Activity
 80            6  Security Product Processing
 88           60  System Logger Data
 92         1852  File System Activity
100          139  DB2 Statistics
101          711  DB2 Accounting
102       135660  DB2 Performance
110          761  CICS Transaction Server
115            6  MQ Statistics
116           99  MQ Accounting
          139932  *** Grand Total ***
```

Figure 122. SMF Recap report section: SMF record counts

Column heading	Description
Type	SMF record type (decimal)
Count	The number of records of this type
Description	Description of the SMF record type (for IBM-defined SMF record types only; blank for user-defined or third-party SMF record types)

CICS

```
V1R3M0                          Transaction Analysis Workbench - CICS

                   ------------------- Monitoring -------------------  - CICS TS -  -- CICS Server
Statistics ---
Job Name APPLID   Performance    Exception     Resource     Identity   Statistics    TSQueue
CFDT        NC
-------- -------- -----------  -----------  -----------  -----------  -----------  ---------  --------
--------
CICPAOR1 CICPAOR1        6012            0            0            0           25          0
0         0
CICPAOR2 CICPAOR2        5940            0            0            0           26          0
0         0
CICPAOR3 CICPAOR3        5904            0            0            0           22          0
0         0
CICPFOR1 CICPFOR1       31212            0            0            0            0          0
0         0
CICPQOR1 CICPQOR1       40764            0            0            0            0          0
0         0
CICPTOR1 CICPTOR1       15432            0            0            0           15          0
0         0
CICPTOR2 CICPTOR2        2340            0            0            0           13          0
0         0
```

Figure 123. SMF Recap report section: CICS

Column heading	Description
Job Name	CICS Transaction Server address space job name
APPLID	CICS generic APPLID

Column heading	Description
Performance	The number of CICS tasks (transactions) with CMF performance class data
Exception	The number of exception events
Resource	The number of CICS tasks (transactions) with CMF transaction resource usage class data
Identity	The number of CICS tasks (transactions) with CMF identity class data
CICS TS statistics	The number of CICS Transaction Server statistics records
TSQueue	The number of CICS Server statistics records for Temporary Storage Queue
CFDT	The number of CICS Server statistics records for Coupling Facility Data Table
NC	The number of CICS Server statistics records for Named Counter Sequence Number

DB2

```
V1R3M0   Transaction Analysis Workbench - DB2

     ------ Accounting ------
SSID    Class 1+     Class 7+  Performance   Statistics
---- -----------  -----------  -----------  -----------
DBA3         206          204       135634           45
DBP3           3            0           26           10
DB3A         149          149            0           84
```

Figure 124. SMF Recap report section: DB2

Column heading	Description
SSID	DB2 subsystem ID
Accounting class 1+	The number of DB2 accounting class 1, 2, 3 records (IFCID 003). Activated by the following DB2 command: `START TRACE(ACCTG) DEST(SMF) CLASS(1,2,3)` Generates SMF records of type 101-0.
Accounting class 7+	The number of DB2 accounting class 7, 8, 10 records (IFCID 239). Activated by: `START TRACE(ACCTG) DEST(SMF) CLASS(7,8,10)` Generates SMF records of type 101-0.
Performance	The number of DB2 performance trace records. Activated by: `START TRACE(PERFM) DEST(SMF) CLASS(1,…)` Generates SMF records of type 102.

Column heading	Description
Statistics	The number of DB2 statistics records. Activated by: `START TRACE(STAT) DEST(SMF) CLASS(1,…)` Generates SMF records of type 100.

IBM MQ

```
V1R3M0   Transaction Analysis Workbench - IBM MQ

     ------ Accounting ------
SSID     Class 1      Class 3   Statistics
---- -----------  -----------  -----------
CSQ6          49           50            6
```

Figure 125. SMF Recap report section: IBM MQ

Column heading	Description
SSID	IBM MQ subsystem ID
Accounting Class 1	The number of MQ accounting class 1 records. Activated by the following MQ command: `START TRACE(ACCTG) DEST(SMF) CLASS(1)` Generates SMF records of type 116-0.
Accounting Class 3	The number of MQ accounting class 3 records. Activated by: `START TRACE(ACCTG) DEST(SMF) CLASS(3)` Generates SMF records of type 116-1.
Statistics	The number of MQ statistics records. Activated by: `START TRACE(STAT) DEST(SMF) CLASS(1)` Generates SMF records of type 115.

Related reference

SMF reports

Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

SMF type 30: Address Space Activity report

The Address Space Activity report uses data from SMF type 30 records to show activity in each address space per interval, where the interval depends on the record subtype.

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                      SMF Type=30 Address Space Activity

------------Interval-------------   System                          ----------- CPU -----------  EXCPs  ----Storage-----
Start Date/Time      Duration Typ    Name   Jobname  Stepname Comp        TCB        SRB  %CPU  /Sec   <16M  >16M 64bit

2011-02-25 09:45:00  00:14:59 INT  FTS1     DB2PMSTR MASTER   0000   1.188172   0.284325   0.2      1    1M   25M    0M
```

```
2011-02-25 09:45:00  00:14:59 INT  FTS1     DB2PDBM1 DBM      0000   0.059287   0.697935   0.1      5    2M  150M    0M

2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPMPP1 REGION   0000   0.110287   0.003570   0.0      0    0M   11M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPCTL  CONTROL  0000   0.300900   0.068595   0.0      0    2M   21M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPDBRC DBRC     0000          0   0.000255   0.0      0    0M   15M    0M
2011-02-25 11:30:01  00:14:58 INT  FTS1     IMSPDLIS DLISAS   0000   0.028560   0.001147   0.0      0    1M   15M    0M

2011-02-25 11:30:01  00:14:58 INT  FTS1     CICSPAOR CICS     0000  97.986300   1.432845  11.1     21    4M 1369M    0M
⋮
```

Description

Column heading	Description	Field or formula
Start	The start of the interval represented by this record. The start of the interval depends on the subtype. For example, for record subtype INT (SMF30TYP=2), the Start column value is the start of the SMF global recording interval.	SMF30ISS
Interval	Interval duration (end time minus start time).	SMF30IET - SMF30ISS
Typ(e)	Record subtype: **STA** Job start or start of other work unit. (SMF30TYP=1) **INT** Activity since previous interval ended. Produced only when interval recording is active. (SMF30TYP=2) **STE** Activity for the last interval before step termination. Produced only when interval recording is active. (SMF30TYP=3) **STT** Step total. (SMF30TYP=4) **JOB** Job termination or termination of other work unit. (SMF30TYP=5) **SAS** System address space, which did not go through full function start. Written only at the expiration of an interval; the values are cumulative and indicate data collected since the initialization of the address space. (SMF30TYP=6)	SMF30TYP
System Name	System name.	SMF30SYN
Jobname	Job or session name.	SMF30JBN
Stepname	Step name (taken from name on EXEC card).	SMF30STM
Comp	Step or job completion code. For details, see the SMF documentation for field SMF30SCC.	SMF30SCC
CPU: TCB	Step CPU time under the task control block (TCB), in microseconds.	(((SMF30CSU_L * 10) / SMF30CPC) * SMF30SUS) / 16
CPU: SRB	Step CPU time under the service request block (SRB), in microseconds.	(((SMF30SRB_L * 10) / SMF30SRC) * SMF30SUS) / 16

Column heading	Description	Field or formula
CPU: %CPU	An indicator of the CPU utilization that this step used during this interval. **Note:** Values greater than 100% will be reported if the job/started task is capable of multitasking on a multiprocessor machine with more than one CPU available to that system/logical partition. To verify this situation, run the "SMF type 70-1: RMF Processor Activity report" on page 443.	(TCB + SRB) / Interval * 100
EXCPs:/Sec	Execute channel program (EXCP) counts per second.	SMF30TEX / Interval
Storage: <16M	The sum of the following two values: • Maximum virtual storage in bytes allocated from the local system queue area (LSQA) and the SWA subpools (less than 16 megabytes). • Maximum virtual storage in bytes allocated from the user subpools (less than 16 megabytes).	SMF30ARB + SMF30URB
Storage: >16M	The sum of the following two values: • Maximum virtual storage in bytes allocated from the local system queue area (LSQA) and the SWA subpools (greater than 16 megabytes). • Maximum virtual storage in bytes allocated from the user subpools (greater than 16 megabytes).	SMF30EAR + SMF30EUR
Storage: 64bit	Amount of 64-bit private storage in bytes that |is obtained by this step or job. This includes guarded virtual storage.	SMF30HVO
Paging/Sec: In	Number of pages that were paged in from auxiliary storage per second.	SMF30PGI / Interval
Paging/Sec: Out	Number of pages that were paged out to auxiliary storage per second.	SMF30PGO / Interval
Paging/Sec: Swap	Number of address space swap sequences per second. (A swap sequence consists of an address space swap-out and swap-in. Logical swap-out and swap-in are not included.)	SMF30NSW / Interval

Related tasks

Tutorial: Submitting a batch report
This tutorial shows you how to create a batch report for a session.

SMF type 33-2: APPC/MVS Conversation List report

The APPC/MVS Conversation List report uses SMF type 33, subtype 2 records to shows details of each inbound and outbound APPC conversation.

Example

```
V1R3M0                              Transaction Analysis Workbench                              Page   1
                                SMF Type=33-2 APPC/MVS Conversation List Report

                Local               Partner                        ----------- Time -----------  ------ Bytes ------
```

```
Start Time       LU Name   Direction  UserId      Job Name  SyncLvl    InputQ   Process     Total  Received       Sent
18:16:47.624543  MVSLU02   Outbound              TWM#RBAT  Syncpt                .324737   .324737        68         83
                 ** Partner ** TPname=IADGEXP_PROFILE

18:16:47.796620  IADGAPPC  Inbound               IADGMPPA  Syncpt    .166232     .154551   .320783        83         68
                 *** Local *** TPname=IADGEXP_PROFILE

18:18:15.136719  MVSLU02   Outbound              TWM#RBAT  Syncpt                .149274   .149274        68         83
                 ** Partner ** TPname=IADGEXP_PROFILE

18:18:15.154361  IADGAPPC  Inbound               IADGMPPA  Syncpt    .014874     .139197   .154071        83         68
                 *** Local *** TPname=IADGEXP_PROFILE

18:18:20.772310  MVSLU02   Outbound              TWM#RBAT  Syncpt                .168397   .168397        68         83
                 ** Partner ** TPname=IADGEXP_PROFILE

18:18:20.790290  IADGAPPC  Inbound               IADGMPPA  Syncpt    .014933     .153256   .168189        83         68
                 *** Local *** TPname=IADGEXP_PROFILE
⋮
```

Description

Column heading	Description	Field or formula
Start Time	Inbound requests: Date/Time when the conversation was associated with a new address space for processing. If an APPC/MVS server processed this conversation, this field shows the time when the server received the conversation through the Receive_Allocate service. Outbound requests: Date/Time when the local program called the Allocate service.	SMF33CST
Local LU Name	Conversation local LU name (not fully qualified).	SMF33CLL
Direction	Inbound or outbound conversation.	SMF33CIO
Partner UserId	Partner user ID.	SMF33CPU
Job Name		SMF33JID
SyncLvl	Sync level of the conversation.	SMF33CSL
Time: InputQ	Elapsed time conversation spent in allocate queue before being processed.	SMF33CST - SMF33CRT (start - receive)
Time: Process	Elapsed conversation processing time.	SMF33CET - SMF33CST (end - start)
Time: Total	Elapsed time between the allocate request for the conversation being received and the conversation being deallocated.	SMF33CET - SMF33CRT (end - receive)
Bytes: Received	Number of bytes received during the conversation.	SMF33CDR
Bytes: Sent	Number of bytes sent during the conversation.	SMF33CDS

SMF type 42-6: DASD Data Set I/O report

The IBM MQ Accounting Class 1 reports use SMF type 42, subtype 6 records to analyze the performance of data set I/O, including read and write frequency, DASD response time, and cache usage.

Exception criteria and selection criteria

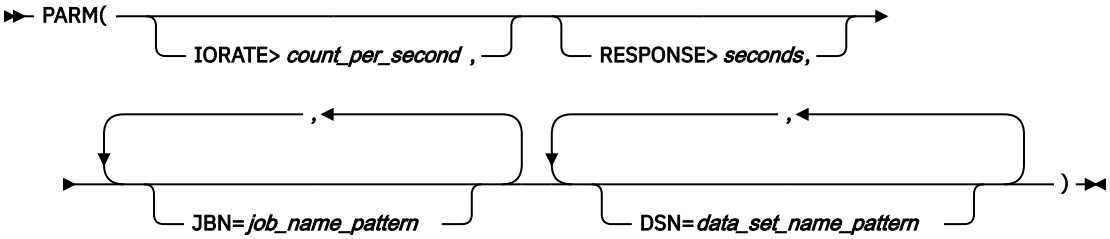
You can optionally specify two types of criteria for this report: exception criteria and selection criteria.

Exception criteria limit the report to *exceptions*: data sets with an I/O rate or response time greater than values that you specify. Exception criteria enable you to create reports that are limited to data sets that, based on these criteria, might require further investigation.

Selection criteria limit the report to job names and data set names that match the patterns you specify. Selection criteria enable you to create reports for specific systems, or subsystems, and their data sets. For example, to analyze the performance of DB2 data set I/O, you could specify a job name pattern of DB2* and a data set name pattern of DSNDB*.

Syntax

To specify exception and selection criteria, follow the **REPORT** command with one or more **PARM** commands.



Exception criteria:

IORATE>*count_per_second*
The data set is an exception when the I/O count per second exceeds the specified threshold value. Allowed values are 1 - 999999.

By itself, a high I/O rate might not indicate a problem; however, if you also specify exception criteria for response time, then the report include only those data sets with a combination of high I/O activity and long response time.

RESPONSE>*seconds*
The data set is an exception when the average I/O operation response time exceeds the specified threshold value. Allowed values are 0.000001 - 999999.

For example, RESPONSE>0.5 specifies that a data set is an exception when the average I/O operation response time exceeds 0.5 seconds.

If you specify both **IORATE** and **RESPONSE**, then a data set is reported only if it matches both exception criteria.

Selection criteria:

JBN=*job_name_pattern*
Selects data sets that belong to a job that matches the specified job name pattern.

DSN=*data_set_name_pattern*
Selects data sets that match the specified data set name pattern.

For VSAM clusters, you cannot specify only the cluster name; you must specify either the complete data set name of the data or index component, or use a wildcard to select both components. For example, for a VSAM cluster named CLUSTER.NAME, specify one of the following:

- To select only the data component, specify DSN=CLUSTER.NAME.DATA
- To select only the index component, specify DSN=CLUSTER.NAME.INDEX
- To select both components, specify DSN=CLUSTER.NAME.*

The job name and data set name patterns can include asterisks (*) as wildcard characters to represent zero or more characters. Wildcards can occur at any position in the pattern.

You can specify multiple job name and data set name patterns. If you specify only job name patterns, then a data set is selected if it matches any of the job name patterns. If you specify only data set name

patterns, then a data set is selected if it matches any of the data set name patterns. If you specify both job name patterns and data set name patterns, then a data set is selected if it matches any combination of job name pattern and data set name pattern.

Some example combinations of job name and data set name selection criteria for different types of system:

CICS KSDS
JBN=CICS*,DSN=CICS.KSDS.*

DB2 database data sets
JBN=DB2*,DSN=DB2.DSNDB.*

IMS database data sets
JBN=IMS*,DSN=IMS.DB.*

A **PARM** command cannot span multiple lines of JCL. If your criteria do not fit on a single line, specify multiple consecutive **PARM** commands. For example:

```
REPORT SMF(42-6)
PARM(RESPONSE>2.5,IORATE>100)
PARM(JBN=DB2*,DSN=DB2.DSNDB.*)
```

In this example, where there is only one job name pattern and one data set name pattern specified, a data set is reported only if it matches all of the criteria.

Example

```
V1R2M0                                  Transaction Analysis Workbench                                  Page   1

                                        SMF Type=42.6 DASD Data Set I/O

----------Interval-----------  System
Start Date/Time      Duration  Name   Jobname
2014-01-13 00:17:54  00:00:01  FTS1   DBP4DBM1

     DSN: DB2P.DSNDBD.CPAP320.XREFRIND.I0001.A001

          -- I/O per sec --  ---------- DASD response time breakdown (average) ------------  ---- Maximum ----
             Reads   Writes  Response  Queuing  Pending  Connect  Disc Rd Disc Wrt      DAO  Response  Service
                 5      145  0.001920 0.000000 0.000000 0.001664 0.000000 0.000000 0.000000  0.012800 0.012672

          ---- Cache candidate rate per second ------  --- Cache I/O per sec ----
             Total Hits      Read Hits     Write Hits       Seq      RLC      ILC
                70   0%         5   0%        65   0%        80        0        0
⋮
```

Description

Column heading	Description	Field or formula
Interval: Start Date/Time	The start of the interval represented by this record.	SMF42PTS
Interval: Duration	Interval duration (end time minus start time).	SMF42PTE - SMF42PTS = Interval
System Name	System name.	SMF42SID
Jobname	Job name.	S42JDJNM
DSN	Data set name.	S42DSN
I/O per sec: Reads	Read I/O requests per second.	S42DSRDT / Interval
I/O per sec: Writes	Write I/O requests per second.	(S42DSION - S42DSRDT) / Interval

Column heading	Description	Field or formula
DASD response time breakdown (average): Response	Response time.	S42DSIOR
... : Queuing	Control unit queue time.	S42DSIOQ
... : Pending	I/O pending time	S42DSIOP
... : Connect	I/O connect time	S42DSIOC
... : Disc Rd	Read I/O disconnect time	S42DSRDD
... : Disc Wrt	Write I/O disconnect time Disconnect time is a measure of the time spent resolving cache misses for read I/O operations. Disconnect time also includes the time it takes to perform synchronous replication for write I/O operations using, for example, IBM Metro Mirror technology on the DS8000. Solid state drives are ideally suited to benefit workloads that are incurring high numbers of cache misses (for example, random reads) but are not expected to substantially reduce the elapsed times for use of synchronous replication technologies.	S42DSIOD - S42DSRDD
... : DAO	Device active only time. DAO is the portion of time in an I/O between which a channel end is received and a device end is received, where they do not occur together.	S42DSDAO
Maximum: Response	Maximum data set I/O response time.	S42DSMXR
Maximum: Service	Maximum data set service time.	S42DSMXS
Cache candidate rate per second: Total	Number of cache candidates per second.	S42DSCND / Interval
... : Hits	Hit ratio for total cache candidates.	S42DSHTS / S42DSCND * 100
... : Read	Number of read cache candidates per second.	(S42DSCND - S42DSWCN) / Interval
... : Hits	Hit ratio for read cache candidates.	(S42DSHTS - S42DSWCI) / (S42DSCND - S42DSWCN) * 100
... : Write	Number of write cache candidates per second.	S42DSWCN / INTERVAL
... : Hits	Hit ratio for read cache candidates.	S42DSWCI / S42DSWCN * 100
Cache I/O per sec: Seq	Number of sequential I/O operations per second.	S42DSSEQ / Interval
... : RLC	Number of RLC I/O operations per second.	S42DSRLC / Interval
... : ILC	Number of ILC I/O operations per second.	S42DSICL / Interval

SMF type 64: VSAM Data Set report

The VSAM Data Set report uses SMF type 64 records to show CLOSE and end-of-volume (EOV) statistics for VSAM data sets.

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                      SMF Type=64 VSAM Data Set Status

                                                                    --Splits--              -------Calls-------    -RLS
Activity-
Close Date Time     Data set name                                     CA   CI   Ext   EXCPs   Get  Upd  Del  Ins     LSR
CF  DASD

2011-02-02 16:00:01 FUNDIP.OME.FTS1MVS.RKM2EDS3.DATA                  5   17    1    3322    13    1    0 1314
0    0    0
2011-02-02 16:00:01 FUNDIP.OME.FTS1MVS.RKM2EDS3.INDEX                 0    5    1    1796     0  259    0    0
0    0    0
2011-02-02 16:00:01 FUNDIP.OME.FTS1MVS.RKM2EDS3.DATA                  5   18    1    3378    13    1    0 1340
0    0    0
2011-02-02 16:00:01 FUNDIP.OME.FTS1MVS.RKM2EDS3.INDEX                 0    5    1    1850     0  261    0    0
0    0    0
2011-02-02 16:00:32 FUNDIP.OME.FTS1MVS.RKM2EDS3.DATA                  5   19    1    3436    13    1    0 1353
0    0    0
2011-02-02 16:00:32 FUNDIP.OME.FTS1MVS.RKM2EDS3.INDEX                 0    5    1    1902     0  275    0    0
0    0    0
2011-02-02 16:00:55 FUNDIP.ANF.QUEUE.DATA                             0    0    1 3754685   23K 8658 4353 1602
0    0    0
2011-02-02 16:00:55 FUNDIP.ANF.QUEUE.INDEX                            0    0    1 3739616    13    0    0    0
0    0    0
2011-02-02 16:01:12 SYS1.DDIR.D                                      15  324    1   24034    1M    0  24K  32K
0    0    0
2011-02-02 16:01:12 SYS1.DDIR.I                                       0   15    1    5669     0  617    0    0
0    0    0
⋮
```

Description

Column heading	Description	Field or formula
Close Date Time	Date and time data set was closed or reached end-of-volume (EOV).	SMF64DTE, SMF64TME
Data set name		SMF64DNM
Splits: CA	Number of control areas that were split in the component.	If VSAM RLS is enabled: SMF64DAS Otherwise: SMF64NAS + SMF64DAS
Splits: CI	Number of control intervals that were split in the component.	If VSAM RLS is enabled: SMF64DCS Otherwise: SMF64NCS + SMF64DCS
Extents	Number of extents.	SMF64NEX + SMF64DEX
EXCPs	Number of execute channel programs (EXCPs) for the data set.	If VSAM RLS is enabled: SMF64DEP Otherwise: SMF64NEP + SMF64DEP

Column heading	Description	Field or formula
Calls: Get	Number of records that were retrieved from the component.	If VSAM RLS is enabled: SMF64DRE Otherwise: SMF64NRE + SMF64DRE
Calls: Upd	Number of records that were updated in the component.	If VSAM RLS is enabled: SMF64DUP Otherwise: SMF64NUP + SMF64DUP
Calls: Del	Number of records that were deleted from the component.	If VSAM RLS is enabled: SMF64DDE Otherwise: SMF64NDE + SMF64DDE
Calls: Ins	Number of records that were inserted into the component.	If VSAM RLS is enabled: SMF64DIN Otherwise: SMF64NIN + SMF64DIN
RLS Activity: LSR	Number of requests where the data was obtained from the local shared buffer pool.	SMF64BMH
RLS Activity: CF	Number of requests where the data was obtained from the DFSMS coupling facility cache structure.	SMF64CFH
RLS Activity: DASD	Number of requests where the data was obtained from DASD.	SMF64RIO

SMF type 70-1: RMF Processor Activity report

The RMF Processor Activity report uses SMF type 70, subtype 1 records to show processor activity for each logical partition, per RMF interval.

Before you run this report

If the SMF input file is likely to contain records from more than one MVS system, it is recommended that you sort the file before running this report.

This initial sort step groups the input records in date, time, and system name order. The example report shown here was produced by pre-sorting the SMF input file using the following DFSORT program control statements:

```
SORT FIELDS=(99,4,CH,A,95,4,CH,A,181,8,CH,A)
INCLUDE COND=((6,1,CH,EQ,X'46'))
```

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page    1
                                    SMF Type=70-1 RMF Processor Activity

                                                          ------------------------ Number of Address Spaces
----------------------
```

```
- Interval Start --  System     - %CPU Busy -        IO   ---- In ---  -In Ready-  -Out Ready-  -Out Wait-  -Log
Ready-  -Log Wait-
  Date       Time    Name        LPAR     MVS       Rate    Avg   Max   Avg   Max    Avg   Max   Avg   Max    Avg
Max   Avg   Max

2010-08-17 23:45:00  FTS1       68.75   87.42     2282.4    151   156     7    86      0     1     0     0      0
0   121   124
                     FTS2        4.07    4.50        9.4     77    80     1    15      0     0     0     0      0
0    48    48
                     FTS3        4.03    4.39       12.6     69    72     1     9      0     0     0     0      0
0    46    46

2010-08-18 00:00:00  FTS1       61.15   72.16     1934.8    150   155     5    76      0     0     0     0      0
0   122   124
                     FTS2        4.15    4.72        8.4     77    79     1     8      0     0     0     0      0
0    47    48
                     FTS3        3.88    4.41       11.7     69    73     1    13      0     0     0     0      0
0    45    46
⋮
```

Description

Column heading	Description	Field or formula
Interval Start	RMF interval start time. If you have synchronized RMF and SMF recording intervals (the default RMF behavior), then this is the same as the SMF global recording interval start time.	SMF70IST
System Name	System name for current system as defined in parmlib member IEASYS*xx* SYSNAME parameter.	SMF70SNM
% CPU Busy: LPAR	The percentage of online time that the logical partition was busy.	(Sum(SMF70PDT) / (SMF70INT * SMF70BNP)) * 100
% CPU Busy: MVS	The percentage of the online time the processor was busy.	((Sum(SMF70INT) - Sum(SMF70WAT)) / (SMF70INT * SMF70BNP)) * 100
IO Rate	The total rate per second that this processor handled I/O interrupts. The rate reflects the processing for the entire interval. This might include periods of time when the SRM enabled or disabled this processor for I/O interrupts. The rate includes interrupts handled by the second level interrupt handler (SLIH), as well as those handled by the Test Pending Interrupt (TPI) instruction.	(Sum(SLIH) + Sum(TPI)) / SMF70INT where: **SLIH** Interruptions that the second level interrupt handler handled **TPI** Interruptions that the Test Pending Interrupt instruction handled
Number of Address Spaces: In: Avg	Average number of address spaces that are in central storage (corresponds to SRM in queue). The In count includes the In Ready count.	SMF70ITT / SMF70SAM
...In: Max	Maximum number of address spaces that are in central storage.	SMF70IMM
...In Ready: Avg	Average number of address spaces that are in central storage and ready to execute or currently in execution.	SMF70RTT / SMF70SAM
...In Ready: Max	Maximum number of address spaces that are in central storage and ready to execute or currently in execution.	SMF70RMM

Column heading	Description	Field or formula
...Out Ready: Avg	Average number of address spaces on the SRM out queue that are physically swapped out of central storage and ready to execute. **Note:** Some address spaces on the SRM out queue might represent those TSO/E users that the SRM intentionally delayed to meet an installation's response time objective. Because these address spaces do not represent a potential performance problem, they are not included in the value reported for OUT READY.	SMF70OTT / SMF70SAM
...Out Ready: Max	Maximum number of address spaces on the SRM out queue that are physically swapped out of central storage and ready to execute.	SMF70OMM
...Out Wait: Avg	Average number of address spaces on the SRM wait queue that are physically swapped out of central storage and not ready to execute.	SMF70WTT / SMF70SAM
...Out Wait: Max	Maximum number of address spaces on the SRM wait queue that are physically swapped out of central storage and not ready to execute.	SMF70WMM
...Log(ical) Ready: Avg	Average number of address spaces on the SRM out queue that are physically in central storage but logically swapped out of central storage and ready to execute.	SMF70LTT / SMF70SAM
...Log(ical) Ready: Max	Maximum number of address spaces on the SRM out queue that are physically in central storage but logically swapped out of central storage and ready to execute.	SMF70LMM
...Log(ical) Wait: Avg	Average number of address spaces on the SRM wait queue that are physically in central storage but logically swapped out of central storage and not ready to execute.	SMF70ATT / SMF70SAM
...Log(ical) Wait: Max	Maximum number of address spaces on the SRM wait queue that are physically in central storage but logically swapped out of central storage and not ready to execute.	SMF70AMM

SMF type 75: RMF Page Data Set Activity report

The RMF Page Data Set Activity report uses SMF type 75 records to show page data set activity information gathered by the Resource Measurement Facility (RMF).

Before you run this report

If the SMF input file is likely to contain records from more than one MVS system, it is recommended that you sort the file before running this report.

This initial sort step groups the input records in date, time, and system name order. You can use the following DFSORT program control statements:

```
SORT FIELDS=(59,4,CH,A,55,4,CH,A,141,8,CH,A)
INCLUDE COND=((6,1,CH,EQ,X'4B'))
```

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                  SMF Type=75 RMF Page Data Set Activity

Date: 2010-08-17  Time: 23:45:00  SID: FTS1

Page    -------- Slots Used -------     %    Bad     In    Trans   Number   Pages
Type     Alloc    Min    Max    Avg  Full  Slots    Use     Time  I/O Req   Xferd VIO Data Set Name

PLPA     44999  20078  20078  20078   45%      0      0        0        0       0     FUNDI1.FTS1.PAGE.PLPA
Common   89999   3129   3129   3129    3%      0      0        0        0       0     FUNDI1.FTS1.PAGE.COMMON
Local    1080K   101K   101K   101K    9%      0      0        0       10      10  Y  FUNDI1.FTS1.PAGE.LOCAL1
Local    1080K   102K   102K   102K    9%      0      0        0       10      10  Y  FUNDI1.FTS1.PAGE.LOCAL2
Local    1080K   103K   103K   103K   10%      0      0        0        6       6  Y  FUNDI1.FTS1.PAGE.LOCAL3
Local    1080K   109K   109K   109K   10%      0      0        0       13      13  Y  FUNDI1.FTS1.PAGE.LOCAL4
Local    1080K   105K   105K   105K   10%      0      0        0        6       6  Y  FUNDI1.FTS1.PAGE.LOCAL5

Date: 2010-08-17  Time: 23:45:00  SID: FTS2

Page    -------- Slots Used -------     %    Bad     In    Trans   Number   Pages
Type     Alloc    Min    Max    Avg  Full  Slots    Use     Time  I/O Req   Xferd VIO Data Set Name

PLPA     44999  20083  20083  20083   45%      0      0        0        0       0     FUNDI1.FTS2.PAGE.PLPA
Common   10799   2317   2317   2317   21%      0      0        0        0       0     FUNDI1.FTS2.PAGE.COMMON
Local      157K   6596   6596   6596    4%      0      0        0        6       6  Y  FUNDI1.FTS2.PAGE.LOCAL1
Local      157K   6707   6707   6707    4%      0      0        0        0       0  Y  FUNDI1.FTS2.PAGE.LOCAL2
Local      157K   9191   9191   9191    6%      0      0        0        0       0  Y  FUNDI1.FTS2.PAGE.LOCAL3
Local      157K   7686   7686   7686    5%      0      0        0        0       0  Y  FUNDI1.FTS2.PAGE.LOCAL4
Local      157K   8628   8628   8628    5%      0      0        0        0       0  Y  FUNDI1.FTS2.PAGE.LOCAL5
⋮
```

Description

Column heading	Description	Field or formula
Page Type	Page space type.	SMF75PST
Slots Used: Alloc	Total number of slots contained within the page swap data set.	SMF75SLA
Slots Used: Min	Minimum number of slots used.	SMF75MNU
Slots Used: Max	Maximum number of slots used.	SMF75MXU
Slots Used: Avg	Average number of slots used.	SMF75AVU
% Full	Average percentage of allocated slots actually used.	(SMF75AVU / SMF75SLA) * 100
Bad Slots	Number of unusable slots.	SMF75BDS
In Use	Number of samples indicating data set was being used by the auxiliary storage manager (ASM).	SMF75USE
Trans Time	Page transfer time.	((SMF75CYC * 1000) * SMF75USE) / SMF75PGX
Number I/O Req	Number of I/O requests for the data set.	SMF75SIO
Pages Xferd	Number of pages transferred to or from page data set.	SMF75PGX
VIO	"Y" if data set accepts virtual input/output (VIO) pages.	SMF75FL2 (bit 0)
Data Set Name	Page data set name.	SMF75DSN

SMF type 78-2: RMF Virtual Storage Activity report

The RMF Virtual Storage Activity report uses SMF type 78, subtype 2 records to show minimum and maximum virtual storage usage data per RMF interval.

Before you run this report

If the SMF input file is likely to contain records from more than one MVS system, it is recommended that you sort the file before running this report.

This initial sort step groups the input records in date, time, and system name order. You can use the following DFSORT program control statements:

```
SORT FIELDS=(75,4,CH,A,71,4,CH,A,157,8,CH,A)
INCLUDE COND=((6,1,CH,EQ,X'4E'),AND,(23,2,CH,EQ,X'0002'))
```

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                     SMF Type=78-2 RMF Virtual Storage Activity

- Interval Start --  System                    ---------------------- Usage ----------------------
Date       Time      Name      Type     Size     Min Time             Max Time             Avg   Pct

2010-06-14 00:00:00  FTS1      CSA     3364K    612K 23:59:59.28    612K 23:59:59.28    612K  18.2
                               ECSA     384M    131M 23:59:59.28    131M 00:14:49.52    131M  34.1
                               SQA     1744K    444K 23:59:59.28    444K 23:59:59.28    444K  25.5
                               ESQA   47772K  22156K 00:07:19.68  22456K 00:13:19.35  22199K  46.5

                     FTS2      CSA     3364K    376K 23:59:59.28    376K 23:59:59.28    376K  11.2
                               ECSA     384M  39796K 00:09:49.63  39888K 00:02:09.31  39851K  10.1
                               SQA     1744K    404K 23:59:59.28    404K 23:59:59.28    404K  23.2
                               ESQA   47772K  19300K 00:07:39.61  19368K 23:59:59.28  19326K  40.5

                     FTS3      CSA     3364K    540K 23:59:59.28    540K 23:59:59.28    540K  16.1
                               ECSA     384M    119M 00:02:39.71    119M 00:12:19.58    119M  31.2
                               SQA     1744K    400K 23:59:59.28    400K 23:59:59.28    400K  22.9
                               ESQA   47772K  19128K 00:02:19.79  19256K 23:59:59.28  19171K  40.1

           00:15:00  FTS1      CSA     3364K    612K 00:14:58.96    612K 00:14:58.96    612K  18.2
                               ECSA     384M    131M 00:24:59.79    131M 00:27:29.74    131M  34.1
                               SQA     1744K    444K 00:14:58.96    444K 00:14:58.96    444K  25.5
                               ESQA   47772K  22116K 00:17:19.47  22388K 00:15:18.88  22166K  46.4

                     FTS2      CSA     3364K    376K 00:14:58.96    376K 00:14:58.96    376K  11.2
                               ECSA     384M  39796K 00:16:09.22  39868K 00:15:18.88  39836K  10.1
                               SQA     1744K    404K 00:14:58.96    404K 00:14:58.96    404K  23.2
                               ESQA   47772K  19296K 00:27:39.18  19344K 00:14:58.96  19328K  40.5

                     FTS3      CSA     3364K    540K 00:15:00.01    540K 00:15:00.01    540K  16.1
                               ECSA     384M    119M 00:23:19.13    119M 00:15:00.01    119M  31.2
                               SQA     1744K    400K 00:15:00.01    400K 00:15:00.01    400K  22.9
                               ESQA   47772K  19128K 00:22:19.36  19188K 00:15:00.01  19176K  40.1
⋮
```

Description

Column heading	Description	Field or formula
Interval Start: Date	RMF interval start date.	SMF78DAT
Interval Start: Time	RMF interval start time.	SMF78IST
System Name	System name.	SMF78SNM

Column heading	Description	Field or formula
Type	Virtual storage type: **CSA** Common service area (<16M) **ECSA** Extended common service area (>16M) **SQA** System queue area (<16M) **ESQA** Extended system queue area (>16M)	
Size	Virtual storage size.	For each virtual storage type: **CSA** R782CS **ECSA** R782ECS **SQA** R782SS **ESQA** R782ESS
Usage: Min	Minimum usage for the RMF intervals covered by the time period of this report.	For each virtual storage type: **CSA** R782CSAU, sub-field VSDBMIN **ECSA** R782CSAU, sub-field VSDAMIN **SQA** R782SQAU, sub-field VSDBMIN **ESQA** R782SQAU, sub-field VSDAMIN
Usage: Time (Min)	Time of the minimum usage.	For each virtual storage type: **CSA** R782CSAU sub-field VSDBNTME **ECSA** R782CSAU sub-field VSDANTME **SQA** R782SQAU sub-field VSDBNTME **ESQA** R782SQAU sub-field VSDANTME

Column heading	Description	Field or formula
Usage: Max	Maximum usage for the RMF intervals covered by the time period of this report.	For each virtual storage type: **CSA** R782CSAU sub-field VSDBMAX **ECSA** R782CSAU sub-field VSDAMAX **SQA** R782SQAU sub-field VSDBMAX **ESQA** R782SQAU sub-field VSDAMAX
Usage: Time (Max)	Time of the maximum value.	For each virtual storage type: **CSA** R782CSAU sub-field VSDBXTME **ECSA** R782CSAU sub-field VSDAXTME **SQA** R782SQAU sub-field VSDAXTME **ESQA** R782SQAU sub-field VSDAXTME
Avg	Average usage for the RMF intervals covered by the time period of this report.	For each virtual storage type: **CSA** (R782CSAU sub-field VSDBTOTL) / SMF78SAM **ECSA** (R782CSAU sub-field VSDATOTL) / SMF78SAM **SQA** (R782SQAU sub-field VSDBTOTL) / SMF78SAM **ESQA** (R782SQAU sub-field VSDATOTL) / SMF78SAM

SMF type 79-15: IRLM Long Lock Detection report

The IRLM Long Lock Detection report uses SMF type 79, subtype 15 records to show locks that occur when sharing data among several IMS instances in a sysplex.

Example

```
V1R3M0                                          Transaction Analysis Workbench                                  Page
1
                                          SMF Type=79.15 IRLM Long Lock Detection Report
             Cycle    Entry                                    Reg
Max                                        CICS
Time         Number   Type  IMS ID    Trancode  PSBname   PST  Typ  Duration  Locks  Recovery Token
Resource  Task

08:51:47.440 25853771 Wait  ISA2      CI1CSAC3  PCM0F0     49      11.534336      0  CI1CSAC3/C5BF632F08B62783
HNMTRM01  00088603
```

```
08:51:47.440 25853771 Block ISA3      CI1ESAE1  PCM0F0    127     111.149056     44  CI1ESAE1/
C5BF62D0456F8085            00036462
08:54:36.250 25854107 Wait  ISA3      CI1ESAE5  PCM0F0    102      11.534336      0  CI1ESAE5/C5BF63D077B36503
HNMTRM01  00088040
08:54:36.250 25854107 Block ISA4      CI1FSAF3  PCM0F0     40      98.566144     44  CI1FSAF3/
C5BF637DEF1A2001            00032398
15:25:31.580 25900783 Wait  ISA1      CI1ASAA2  PRE0F0     90      11.534336     26  CI1ASAA2/C5BFBB316C472003
SHSECN08  00013029
15:25:31.580 25900783 Block ISA1      CI1ASAA1  PSA0F0     60      11.534336      2  CI1ASAA1/
C5BFBB3166E1F584            00048273
⋮
```

Description

Column heading	Description	Field
Time	Time when the record was moved into the SMF buffer.	SMF79TME
Cycle Number	Dead lock cycle number.	R79FDLKC
Entry Type	"Block" (R79FETYP="B") or "Wait" (R79FETYP="W").	R79FETYP
IMS ID	IMS subsystem ID.	R79FIMSI
Trancode	Transaction name or job name.	R79FTRNM
PSBname	PSB name.	R79FPSBN
PST	PST number.	R79FPSTN
Reg Typ	Region type.	R79FRGTY
Duration	Scheduled elapsed time.	R79FLHTI
Max Locks	Max lock held.	R79FLHCN
Recovery Token	Recovery token.	R79FRCVT
Resource	Resource (DB/Area) name.	R79FRSNA
CICS Task	CICS task ID (if CICS).	R79FCTID

SMF type 88-1: System Logger Log Stream Summary report

The System Logger Log Stream Summary report uses SMF type 88, subtype 1 records to show system logger data, including log stream activity and performance degradation events.

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                  SMF Type=88-1 System Logger - Logstream Summary

Logstream name             MVSID   Structure name    Group       First interval start    Last interval stop      Total
In
STC@CICS.CCVQ42D2.DFHLOG   FTS1    *DASDONLY*                    09:45:00.00  2/25/2011  11:45:00.00  2/25/2011
0002

           ---------- IXGWRITES ----------               ---------------- DELETIONS -----------------
                                              Bytes       Count       Count       Bytes       Bytes
                                           Writn to        With     Without       After    Int Stor
                          Total     Average  Interim        DASD       DASD     Offload    w/o DASD
              Count       Bytes       Bytes  Storage       Write      Write     w. DASD       Write
           --------    --------    --------  --------    --------    --------    --------    --------
Total          22831      16311K        714     96432K       19579        2860       82883K      12009K
Rate(/Sec)         3        2265              13393           2           0          11511        1668
Minimum            0           0                  0           0           0              0           0
Maximum         6810     5228496              29237K        6265         821       27136K     3366912

           ------------------------------------------------ EVENTS ------------------------------------------------
                              Demand                                                                  Demand     Staging
                   Staging       DASD       Block     Staging       Entry      Struct     Init'd     DS Async
           Offloads    Threshld     Shifts      Length       Full        Full        Full    Offloads    Buf Full
           --------    --------    --------  --------    --------    --------    --------    --------    --------
Total            28          76           1                    0           0           0           0           0
Rate(/Sec)        0           0           0                    0           0           0           0           0
Minimum           0           0           0         120        0           0           0           0           0
Maximum           8           0           1       33767        0           0           0           0           0
```

```
          ----------------------- EVENTS -------------------------    --------------- DASD Writes ----------------
                                                 Struct     Struct
                                               Rebuilds   Rebuilds                  Total
             Type1      Type2      Type3       Init'd     Complt'd      Count       Bytes     Average       Waits
          --------   --------   --------    --------    --------    --------    --------    --------    --------
Total        22755         76          0           0           0          60      14878K           0          31
Rate(/Sec)       3          0          0           0           0           0        2066                       0
Minimum          0          0          0           0           0           0           0                       0
Maximum       6794         21          0           0           0          19     5295992                      11
⋮
```

Description

Row headings:

Total
Total for this field across all intervals

Rate(/Sec)
Activity Rate per second for this field

Minimum
Minimum value seen for this field in any interval

Maximum
Maximum value seen for this field in any interval

IXGWRITES information:

Column heading	Description	Field
Count	The number of IXGWRITE requests.	SMF88LWI
Total Bytes	Bytes written by IXGWRITE requests.	SMF88LWB
Average Bytes	The average number of bytes written by IXGWRITE requests.	Sum(SMF88LWB) / SMF88LWI
Bytes Writn to Interim Storage	The number of bytes written to interim storage.	SMF88SWB

DELETIONS information:

Column heading	Description	Field or formula
Count With DASD Write	The number of deletes from interim storage written to DASD.	SMF88SAI
Count Without DASD Write	Number of deletes from interim storage without having been written to the log data set.	SMF88SII
Bytes After Offload w. DASD	Bytes deleted after data was offloaded to DASD log data sets. If SMF88SIB is high and the SMF88SAB is low, CICS is successfully using interim storage to avoid the I/O incurred by offloading to DASD log data sets.	SMF88SAB
Bytes Int Stor w/o DASD Write	Count of bytes deleted instead of being written to DASD. Due to CICS tail trimming; that is, deletion of records which are no longer required for recovery. This value shows how successfully CICS avoids offloads for data that it intends to delete from interim storage.	SMF88SIB

EVENTS information:

Column heading	Description	Field or formula
Offloads	Number of times the log stream was offloaded.	SMF88EO

Column heading	Description	Field or formula
Staging Threshold	Number of times system logger detected a Staging Data Set Threshold Hit condition (HIGHOFFLOAD reached) for the staging data set.	SMF88ETT
Demand DASD Shifts	Number of log stream DASD shifts (additional log data set allocates) initiated by this system. For DFHLOG and DFHSHUNT this value should be small, otherwise too much data is being offloaded. (The LS_SIZE parameter for the IXCMIAPU logstream definition utility should be checked.)	SMF88EDS
Block Length	Block length.	Minimum: SMF88LIB Maximum: SMF88LAB
Staging Full	Number of times staging data set was full. The cause of any non-zero condition should be investigated.	SMF88ETF
Entry Full	Number of times all log streams connected to the structure are offloaded by IXLOGR due to 90% of the structure's list entries being full.	SMF88EFS
Struct Full	Number of times a structure full condition was reached. The cause of any non-zero condition should be investigated.	SMF88ESF
Demand Init'd Offloads	Number of demand initiated offloads.	SMF88EDO
Staging DS Async Buf Full	Number of times the system logger detected a Staging Data Set Async Buffer Full condition for this log stream on this system for this SMF interval.	SMF88EAF
Type1	Type 1 CF event. Normal write. Indicates that, after the write completed, the percentage of resource in use by the structure was less than the high offload threshold, meaning that system logger is using the coupling facility successfully. This number should be high.	SMF88SC1
Type2	Type 2 CF event. Indicates that, after the write completed, the percentage of the log stream in use was greater than or equal to the high off load threshold. This can happen at the point where the offload value is reached or the offload is already in progress.	SMF88SC2

Column heading	Description	Field or formula
Type3	Type 3 CF event. Indicates that a given log stream is close to consuming 90% of the coupling facility resource allocated to it. A type-3 completion can occur if there is a failure which prevents system logger from promptly moving data from the coupling facility structure to DASD log data sets or if the system logger configuration is tuned incorrectly. For example, system logger's access to its DASD log data sets would be slowed if those data sets reside on the same device as some other heavily-used data sets. A type-3 can also occur if many log streams are defined to share the same structure, because each newly defined log stream causes system logger to dynamically repartition storage among the existing log streams. If a log stream has a large proportion of type-3 completions, system logger is getting dangerously close to the STRUCTURE FULL condition.	SMF88SC3
Struct Rebuilds Init'd	Number of structure rebuild events initiated for this log stream, as seen by this system. Excessive structure rebuilds should be investigated. Structures are rebuilt in the event of logstream connectivity failure in accordance with the REBUILDPERCENT parameter of the IXCMIAPU utility.	SMF88ERI
Struct Rebuilds Complt'd	Number of structure rebuild events completed for this log stream, as seen by this system. Excessive structure rebuilds should be investigated. Structures are rebuilt in the event of logstream connectivity failure in accordance with the REBUILDPERCENT parameter of the IXCMIAPU utility.	SMF88ERC

DASD Writes information:

Column heading	Description	Field or formula
Count	Number of DASD write requests.	SMF88LIO
Total Bytes	Total bytes written to DASD (offload data sets).	SMF88LDB
Average	Average number of bytes written to DASD (offload data sets).	Sum(SMF88LDB) / SMF88LIO
Waits	Number of times System Logger had to suspend processing before writing to DASD because a previous DASD write request had not completed.	SMF88LIS

SMF type 101: DB2 Thread Accounting Summary report

The DB2 Thread Accounting Summary report uses SMF type 101 records to show accounting information for DB2 connections.

Example

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                      SMF Type=101 DB2 Accounting Summary


DB2    Plan       ---------- Connection -----------   Thread
SSID   Name       Name     Type                        Count
```

```
DB3A   CEXTPGM    IADG     IMS MPP                       68                                    Start: 2010-06-24
15:27:39
                                                                                               End:   2010-06-24
16:44:00
                           Class1: Thread Time      Avg: Elapsed=70.43305  CPU= .011006          Interval:
01:16:20
                                                    Max: Elapsed=2045.732  CPU= .013724          Rate/
sec:               < 1
                           Class2: In-DB2 Time      Avg: Elapsed= .015108  CPU= .006035
                                                    Max: Elapsed= .033537  CPU= .008234
                           Class3: Suspend Time     Avg: Total  = .008709  I/O= .000000  Lock/Latch= .002404  Other=
.006305
                                                    Max: Total  = .017377  I/O= .000000  Lock/Latch= .007199  Other=
.010178
                           Buffer Manager Summary   Avg: GtPgRq=     7.0  SyPgUp=     3.0
                                                    Max: GtPgRq=       7  SyPgUp=       3
                           Locking Summary          Avg: Suspnd=      .0  DeadLk=      .0  TmeOut=      .0  MxPgLk=
1.0
                                                    Max: Suspnd=       0  DeadLk=       0  TmeOut=       0
MxPgLk=       1
                           SQL DML Query/Update     Avg: Sel=      .0  Ins=     1.0  Upd=     1.0  Del=     1.0
                                                    Max: Sel=       0  Ins=       1  Upd=       1  Del=       1
                           SQL DML 'Other'          Avg: Des=      .0  Pre=      .0  Ope=     1.0  Fet=     9.0
Clo=     1.0
                                                    Max: Des=       0  Pre=       0  Ope=       1  Fet=       9
Clo=       1


DB3A   MQATPGM    IADG     IMS MPP                       18                                    Start: 2010-06-24
15:26:35
⋮
```

Description

The report contains the following information for each connection:

Class1: Thread Time

Elapsed
Elapsed time covered by the DB2 Accounting record; derived from end time minus begin time. It gives the time from when the DB2 thread is obtained (at the first SQL call) to the time it is terminated or reused by another sign-on (which might be well after the task completes if it is a protected thread).

CPU
TCB CPU time used by the thread; derived from QWACEJST minus QWACBJST.

Class2: In-DB2 Time

Elapsed
Accumulated elapsed time used in DB2 (field: QWACASC).

CPU
Accumulated TCB CPU time used in DB2 (field: QWACAJST).

Class3: Suspend Time
This data is only available when DB2 class 3 accounting trace data is present.

Total
Total class 3 suspend time.

I/O
Accumulated elapsed I/O wait time (field: QWACAWTI).

Lock/Latch
Accumulated lock and latch time (field: QWACAWTL).

Other
Total of the other nine class 3 suspend clocks:

1. Log write I/O (field: QWACAWLG)
2. Page latch contention (field: QWACAWTP)
3. Send message to other DB2 members in the data sharing group (field: QWACAWTG)
4. Global contention for parent L-Locks (field: QWACAWTJ)

5. Stored procedure waiting for available TCB (field: QWACCAST)
6. User-defined function waiting for available TCB (field: QWACUDST)
7. Read I/O done under another thread (field: QWACAWTR)
8. Write I/O done under another thread (field: QWACAWTW)
9. Synchronous execution unit switch for DB2 Commit, Abort, or Deallocation processing (field: QWACAWTE)

Buffer Manager Summary

These fields give the totals for all buffer pools.

GtPgRq
Number of get page requests issued (field: QBACGET).

SyPgUp
Number of system page (buffer) updates (field: QBACSWS).

Locking Summary

Suspnd
Number of suspends due to lock conflict (field: QTASLOC).

DeadLk
Number of deadlocks (field: QTXADEA).

TmeOut
Number of timeouts (field: QTXATIM).

MxPgLk
Maximum number of page locks held (field: QTXANPL).

SQL DML Query/Update

Sel
Number of SELECT statements processed (field: QXSELECT).

Ins
Number of INSERT statements processed (field: QXINSRT).

Upd
Number of UPDATE statements processed (field: QXUPDTE).

Del
Number of DELETE statements processed (field: QXDELET).

SQL DML 'Other'

Des
Number of DESCRIBE, DESCRIBE CURSOR, DESCRIBE INPUT, and DESCRIBE PROCEDURE statements processed (field: QXDESC).

Pre
Number of SQL PREPARE statements processed (field: QXPREP).

Ope
Number of OPEN statements processed (field: QXOPEN).

Fet
Number of FETCH statements processed (field: QXFETCH).

Clo
Number of CLOSE statements processed (field: QXCLOSE).

Related reference

DB2 accounting exception reports and extracts

DB2 accounting exception reports and extracts are generated from DB2 accounting (SMF type 101) records. DB2 accounting records from the specified input files are checked against optional *filtering criteria* for the DB2 subsystem ID (SSID), plan, and connection type. Records that match the filtering criteria are checked against *exception criteria* that specify thresholds for various performance-related values, such as CPU time.

SMF type 116-0: IBM MQ Accounting Class 1 reports

The IBM MQ Accounting Class 1 reports use SMF type 116, subtype 0 records to show accounting information for IBM MQ transactions.

To produce SMF type 116, subtype 0 records (also known as message manager data records), you need to start IBM MQ accounting trace class 1.

There are two IBM MQ Accounting Class 1 reports:

List report
Shows accounting information for each IBM MQ thread. For details, see “Description” on page 457.

Summary report
Shows average and maximum values for the information in the List report, summarized by connection and, if applicable, transaction identifier. A connection is defined by a connection type (such as CICS) and a connection name (such as the CICS region applid); for details, see “Description” on page 457.

Both reports are generated when you specify the **REPORT** command parameter **SMF(116-0)**.

If you need more IBM MQ accounting information than is provided by these class 1 reports, see the more detailed “SMF type 116-1: IBM MQ Accounting Class 3 reports” on page 458.

Example List report

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                                SMF Type=116-0 IBM MQ Accounting Class 1 List


                             -- Connection --  Tran/                                          --------- GET/PUTx
Counts ---------
Date       Time        SSID Type    Name        Task    Userid    ---- Time -----  - Counts -         <=99   <=999
<=9999 >=10000

2010-03-03 09:00:40.28 SYSU CICS    CICSSYSN  TRAL      CICSSYSN  Elapse 45.62196  COMM     1  Get       0
0       0       0
                                               16067              CPU    0.000332  BKO      1  Put       1
0       0       0
2010-03-03 09:00:45.40 SYSB CICS    CICSS01N  TRAL      CICSS01N  Elapse 58.33937  COMM     1  Get       0
0       0       0
                                               17029              CPU    0.000281  BKO      1  Put       0
1       0       0
2010-03-03 09:00:47.39 SYSB CICS    CICSS01N  TRCA      CICSS01N  Elapse 1.992634  COMM     0  Get       0
0       0       0
                                               17030              CPU    0.008195  BKO      1  Put       0
0       0       0
2010-03-03 09:00:54.74 SYSU CICS    CICSSYSN  TRCA      CICSSYSN  Elapse 14.46009  COMM     0  Get       0
0       0       0
                                               16068              CPU    0.008270  BKO      1  Put       0
0       0       0
2010-03-03 09:01:40.62 SYSU CICS    CICSSYSN  TRAL      CICSSYSN  Elapse 45.87968  COMM     1  Get       0
0       0       0
                                               16070              CPU    0.000419  BKO      1  Put       1
0       0       0
2010-03-03 09:01:45.54 SYSB CICS    CICSS01N  TRAL      CICSS01N  Elapse 419.1730  COMM     1  Get       0
0       0       0
                                               17034              CPU    0.000370  BKO      1  Put       0
1       0       0
⋮
```

Example Summary report

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                              SMF Type=116-0 IBM MQ Accounting Class 1 Summary


MQ     -- Connection --              ---- QMGR CPU ----   -- Elapsed Time --
SSID   Type    Name     Tran          Average   Maximum    Average   Maximum
SYSB   CICS    CICSSYSP BBTR         0.000896  0.001344   505.4793  2122.330

               Thread  --------- Average GET Counts ----------   --------- Average PUT Counts ----------
                Count       <=99     <=999    <=9999   >=10000        <=99     <=999    <=9999   >=10000
                    6        0.0       0.0       1.3       0.0         0.0       0.0       2.7       0.0


MQ     -- Connection --              ---- QMGR CPU ----   -- Elapsed Time --
SSID   Type    Name     Tran          Average   Maximum    Average   Maximum
SYSB   CICS    CICSS01N TRAL         0.000381  0.000573   121.2418  2161.082

               Thread  --------- Average GET Counts ----------   --------- Average PUT Counts ----------
```

```
                Count       <=99     <=999    <=9999   >=10000        <=99     <=999    <=9999   >=10000
                   60        0.0       0.0       0.0       0.0         0.0       1.0       0.0       0.0
⋮
```

Description

The IBM MQ Accounting Class 1 reports contain the following information:

Task identification

- IBM MQ subsystem name (SSID), extracted from the instrumentation standard header (described by assembler macro CSQDQWHS in the IBM MQ library SCSQMACS)
- Connection details (such as type, name, and transaction ID), extracted from the instrumentation correlation data (macro CSQDQWHC)

Summary statistics

For each task:

- Elapsed time, number of commits and backouts, extracted from the accounting data (macro CSQDQWAC)
- CPU time, number of gets and puts, extracted from the message manager accounting data (macro CSQDQMAC)

The following table describes each column in the reports:

Column heading	Description	Field
Date	Date portion of the SMF record time stamp.	SM116DTE
Time	Time portion of the SMF record time stamp.	SM116TME
SSID	IBM MQ subsystem name.	QWHSSSID
Connection: Type	**QMGR** Queue manager **CICS** CICS **BAT/TSO** Batch or TSO **IMSCTL** IMS control region **IMSDEP** IMS MPP or BMP **CMDSERV** Command server **CHIN** Channel initiator **RRSBAT** RRS batch **UNKNOWN** None of the preceding connection types	QWHCATYP
Connection: Name	Depends on the connection type: **CICS** CICS region applid **IMSDEP or IMSCTL** IMS subsystem ID Blank for other connection types.	QWHCCN

Column heading	Description	Field
Tran	Depends on the connection type: **CICS** CICS transaction code **IMSDEP** PSB name Blank for other connection types.	QWHCCV
Task	CICS connection type only: the CICS task number.	QWHCCV
Userid	User ID associated with the transaction.	QWHCOPID
Time: Elapse	IBM MQ thread end time minus start time.	QWACESC - QWACBSC
Time: CPU	CPU time used.	QMACCPUT
Counts: COMM	Number of commit phase 2 requests.	QWACCOMM
Counts: BKO	Number of backout requests.	QWACBACK
GET Counts: <=99	Number of GET calls for 0 - 99 bytes.	QMACGETA
GET Counts: <=999	Number of GET calls for 100 - 999 bytes.	QMACGETB
GET Counts: <=9999	Number of GET calls for 1000 - 9999 bytes.	QMACGETC
GET Counts: >=10000	Number of GET calls for 10000 or more bytes.	QMACGETD
PUTx Counts: <=99	Number of PUT and PUT1 calls for 0 - 99 bytes.	QMACPUTA
PUTx Counts: <=999	Number of PUT and PUT1 calls for 100 - 999 bytes.	QMACPUTB
PUTx Counts: <=9999	Number of PUT and PUT1 calls for 1000 - 9999 bytes.	QMACPUTC
PUTx Counts: >=10000	Number of PUT and PUT1 calls for 10000 or more bytes.	QMACPUTD

SMF type 116-1: IBM MQ Accounting Class 3 reports

The IBM MQ Accounting Class 3 reports use SMF type 116, subtype 1 and subtype 2 records to show thread-level and queue-level accounting information for IBM MQ transactions.

To produce SMF type 116, subtype 1 and subtype 2 records, you need to start IBM MQ accounting trace class 3.

There are two types of IBM MQ Accounting Class 3 report:

List report
Shows detailed accounting information for each thread, including activity on each queue.

Summary report
Shows accounting information summarized by one of the following groupings:

- Transaction (with no queue-specific information)
- Transaction and, for each transaction, activity on each queue
- Queue (with no transaction-specific information)

- Queue and, for each queue, the transactions that used that queue

Example List report

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                               SMF Type=116-1 IBM MQ Accounting Class 3 List

MQACCT1  Printed at 10:50:15  2/03/2011  Data from 21:59:54  3/03/2010


SSID: SYSU Type: CICS    Name: CICSSYSN Tran: TRAL     User: CICSSYSN Task:  16067 NetName: N/A              UOWID: N/A
Channel:                     Channel Connection:                                                Start:  3/03/2010
21:59:54.66

  COMMIT   Count                1  Avg Elapsed   0.000892   Avg CPU      0.000025
  Other    Total Calls          1  Avg Elapsed   0.000558   Avg CPU      0.000024
           #Old Pages           4  #New Pages           1

  Queue: TRR.APPLICATION_ALERT                            (SYSA                                          )
  QType: REMOTE   IType: NONE       GDisp: Q_MGR    Date:  3/03/2010  Time: 09:00:40  P/Set No:        1  BufferPool
No:       1
  First Opened:  3/03/2010 22:00:40.28   Last Closed:  3/03/2010 22:00:40.28   CF Structure Name:

               Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
            ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    PUT1             1   0.000117   0.000117          0           0        0.0          0

    PUT1     Total Bytes          59  #PUT w/Data             1  Min Msg Size        59  Max Msg Siz         59

  Queue: TRR.APPLICATION_ARCHIVE.OK
  QType: LOCAL    IType: NONE       GDisp: Q_MGR    Date:  3/03/2010  Time: 09:00:40  P/Set No:        0  BufferPool
No:       0
  First Opened:  3/03/2010 22:00:40.28   Last Closed:  3/03/2010 22:00:40.28   CF Structure Name:

               Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
            ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    OPEN             1   0.000008   0.000008
    CLOSE            1   0.000002   0.000001
    INQ              1   0.000004   0.000004

  Queue: TRR.APPLICATION_ARCHIVE.FAIL
  QType: LOCAL    IType: NONE       GDisp: Q_MGR    Date:  3/03/2010  Time: 09:00:40  P/Set No:        0  BufferPool
No:       0
  First Opened:  3/03/2010 22:00:40.28   Last Closed:  3/03/2010 22:00:40.28   CF Structure Name:

               Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
            ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    OPEN             1   0.000008   0.000008
    CLOSE            1   0.000002   0.000002
    INQ              1   0.000004   0.000004

  Queue: BRIDGE_FAIL
  QType: LOCAL    IType: NONE       GDisp: Q_MGR    Date:  3/03/2010  Time: 09:00:40  P/Set No:        0  BufferPool
No:       0
  First Opened:  3/03/2010 22:00:40.28   Last Closed:  3/03/2010 22:00:40.28   CF Structure Name:

               Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
            ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    OPEN             1   0.000058   0.000057
    CLOSE            1   0.000007   0.000007
    INQ              1   0.000018   0.000018

  In-MQ Time (Total)      Elapsed:  0.001684   CPU:   0.000282

SSID: SYSB Type: CICS    Name: CICSS01N Tran: TRAL     User: CICSS01N Task:  17029 NetName: N/A              UOWID: N/A
⋮
```

Example Summary (by transaction) report

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                     SMF Type=116-1 IBM MQ Accounting Class 3 Summary (By TRANSACTION)

MQACCT2  Printed at 10:57:15  2/28/2011  Data from 09:00:40 03/03/2010 to 09:59:52 03/03/2010

SSID: SYSB  Type: CICS     Name: CICSSYSP  Tran: TRTI      Threads:         2
  Other     Avg Count              6.0   Avg Elapsed    0.000116  Avg CPU      0.000112

  In-MQ Time (Total)      Elapsed:  0.000233   CPU:   0.000224
  In-MQ Time (Average)    Elapsed:  0.000116   CPU:   0.000112
```

```
SSID: SYSB  Type: CICS     Name: CICSSYSP  Tran: TRTL      Threads:         4

  In-MQ Time (Total)      Elapsed:         0   CPU:          0
  In-MQ Time (Average)    Elapsed:         0   CPU:          0

SSID: SYSB  Type: CICS     Name: CICSSYSP  Tran: TRAG      Threads:         2
  COMMIT    Avg Count              4.5   Avg Elapsed    0.016087  Avg CPU      0.000130
  Other     Avg Count              9.0   Avg Elapsed    0.002000  Avg CPU      0.000928
  Jnl/Log   Avg Bytes          10594.0   Avg FORCEs          4.5  Avg WAIT Elp 0.015065   Avg SUSPEND Elp  0.015967

  In-MQ Time (Total)      Elapsed:  0.098241   CPU:   0.002117
  In-MQ Time (Average)    Elapsed:  0.049120   CPU:   0.001058
⋮
```

Example Summary (by transaction, queue) report

```
V1R3M0                                  Transaction Analysis Workbench                                Page   1
                      SMF Type=116-1 IBM MQ Accounting Class 3 Summary (By TRAN,QUEUE)

MQACCT4  Printed at 10:50:30  2/03/2011  Data from 09:00:40 03/03/2010 to 09:59:52 03/03/2010


SSID: SYSB  Type: CICS     Name: CICSSYSP  Tran: TRTI      Threads:         2
  Other     Avg Count              6.0   Avg Elapsed    0.000116  Avg CPU      0.000112

  In-MQ Time (Total)      Elapsed:  0.000233   CPU:   0.000224
  In-MQ Time (Average)    Elapsed:  0.000116   CPU:   0.000112

SSID: SYSB  Type: CICS     Name: CICSSYSP  Tran: TRTL      Threads:         4

  In-MQ Time (Total)      Elapsed:         0   CPU:          0
  In-MQ Time (Average)    Elapsed:         0   CPU:          0

  Queue: APPLICATION_A_REQUEST
  QType: LOCAL    IType: NONE       GDisp: Q_MGR    QCount:         4

               Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
            ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    OPEN          15.0   0.000019   0.000009
    CLOSE         15.0   0.000002   0.000002
    INQ           15.0   0.000009   0.000008

    In-MQ Time (Total)      Elapsed:  0.001861   CPU:   0.001222
    In-MQ Time (Average)    Elapsed:  0.000465   CPU:   0.000305

  Queue: APPLICATION_B_REQUEST
⋮
```

Example Summary (by queue) report

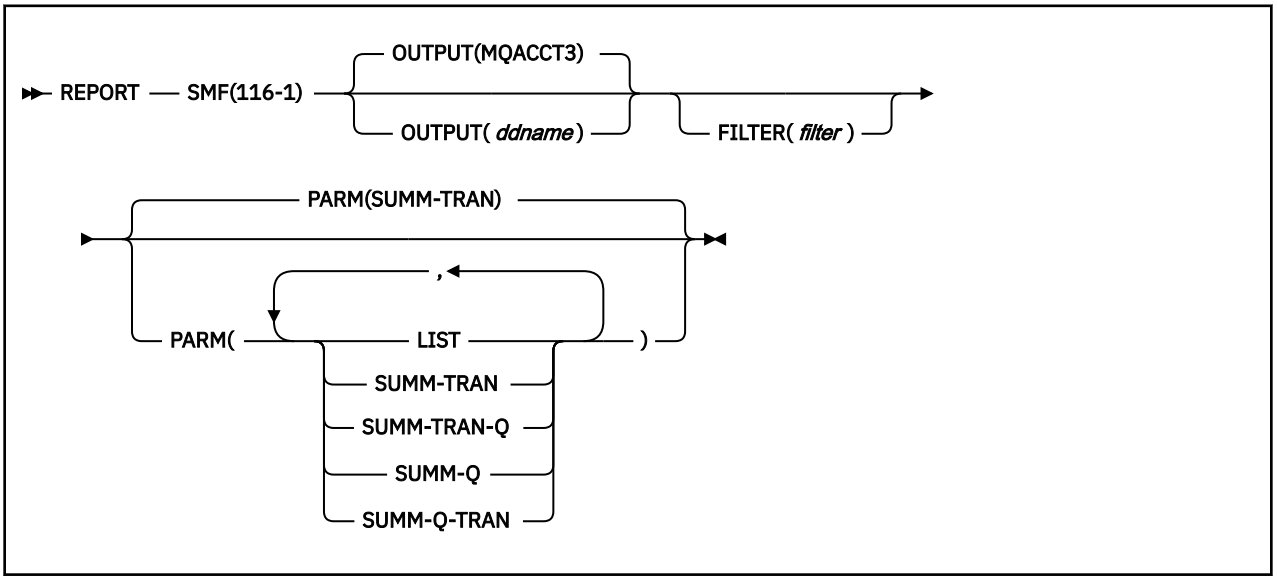
```
V1R3M0                                  Transaction Analysis Workbench                                Page   1
                      SMF Type=116-1 IBM MQ Accounting Class 3 Summary (By QUEUE)

MQACCT3  Printed at 10:57:15  2/28/2011  Data from 09:00:40 03/03/2010 to 09:59:52 03/03/2010


Queue: CICSSYSP.INITIATION.QUEUE
QType: LOCAL    IType: NONE       GDisp: Q_MGR    QCount:         4

             Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs Skip
           ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------  ---------
  OPEN          79.0   0.000459   0.000043
  CLOSE         79.0   0.000104   0.000005
  INQ           79.0   0.000044   0.000020

  In-MQ Time (Total)      Elapsed:  0.192257   CPU:   0.021792
  In-MQ Time (Average)    Elapsed:  0.048064   CPU:   0.005448

Queue: CICSSYSN.INITIATION.QUEUE
QType: LOCAL    IType: NONE       GDisp: Q_MGR    QCount:         4

             Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs Skip
           ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------  ---------
  OPEN          79.0   0.000092   0.000011
  CLOSE         79.0   0.000003   0.000002
  INQ           79.0   0.000010   0.000006

  In-MQ Time (Total)      Elapsed:  0.033775   CPU:   0.006487
  In-MQ Time (Average)    Elapsed:  0.008443   CPU:   0.001621

Queue: CICSSYST.INITIATION.QUEUE
⋮
```

Example Summary (by queue, transaction) report

```
V1R3M0                                  Transaction Analysis Workbench                                  Page   1
                      SMF Type=116-1 IBM MQ Accounting Class 3 Summary (By QUEUE,TRAN)

MQACCT5  Printed at 10:57:15  2/28/2011  Data from 09:00:40 03/03/2010 to 09:59:52 03/03/2010


Queue: CICSSYSP.INITIATION.QUEUE
QType: LOCAL    IType: NONE       GDisp: Q_MGR    QCount:         4

             Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs Skip
           ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------  ---------
  OPEN          79.0   0.000459   0.000043
  CLOSE         79.0   0.000104   0.000005
  INQ           79.0   0.000044   0.000020

  In-MQ Time (Total)      Elapsed:  0.192257   CPU:   0.021792
  In-MQ Time (Average)    Elapsed:  0.048064   CPU:   0.005448

  SSID: SYSB  Type: CICS     Name: CICSSYSN  Tran: TRSP      Threads:         4
    Latch     Max Elapsed Class         24   Max Elapsed    0.062525  Avg Elapsed  0.078056
              Max Count Class           11   Max Count           199  Avg Count       299.0
    Jnl/Log   Avg Bytes            63912.0   Avg FORCEs        167.0  Avg WAIT Elp 0.111191   Avg SUSPEND Elp         0

    In-MQ Time (Total)      Elapsed:  0.756990   CPU:          0
    In-MQ Time (Average)    Elapsed:  0.189247   CPU:          0

Queue: CICSSYSN.INITIATION.QUEUE
QType: LOCAL    IType: NONE       GDisp: Q_MGR    QCount:         4

             Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs Skip
           ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------  ---------
  OPEN          79.0   0.000092   0.000011
  CLOSE         79.0   0.000003   0.000002
  INQ           79.0   0.000010   0.000006

  In-MQ Time (Total)      Elapsed:  0.033775   CPU:   0.006487
  In-MQ Time (Average)    Elapsed:  0.008443   CPU:   0.001621

  SSID: SYSB  Type: CICS     Name: CICSSYSN  Tran: TRSP      Threads:         4
    Latch     Max Elapsed Class         24   Max Elapsed    0.062525  Avg Elapsed  0.078056
              Max Count Class           11   Max Count           199  Avg Count       299.0
    Jnl/Log   Avg Bytes            63912.0   Avg FORCEs        167.0  Avg WAIT Elp 0.111191   Avg SUSPEND Elp         0

    In-MQ Time (Total)      Elapsed:  0.756990   CPU:          0
    In-MQ Time (Average)    Elapsed:  0.189247   CPU:          0

Queue: CICSSYST.INITIATION.QUEUE
⋮
```

Selecting reports

To select a list report or a specific summary report, use the **PARM** parameter of the batch and extract utility **REPORT** command:



Report type	PARM parameter value
List	LIST
Summary (By TRANSACTION)	SUMM-TRAN
Summary (By TRAN,QUEUE)	SUMM-TRAN-Q
Summary (By QUEUE)	SUMM-Q
Summary (By QUEUE,TRAN)	SUMM-Q-TRAN

Specifying `REPORT SMF(116-1)` without a **PARM** parameter generates the default "Summary (By TRANSACTION)" report.

You can request any combination of reports in a single **PARM** parameter. For example:

```
REPORT SMF(116-1) PARM(LIST,SUMM-TRAN,SUMM-Q)
```

However, it is recommended that you specify only one **PARM** parameter value per **REPORT** command, so that you can direct each report to a separate output data set (ddname). For example:

```
REPORT SMF(116-1) PARM(LIST)      OUTPUT(MQAL)
REPORT SMF(116-1) PARM(SUMM-TRAN) OUTPUT(MQAST)
REPORT SMF(116-1) PARM(SUMM-Q)    OUTPUT(MQASQ)
```

Description

The reports consist of combinations of the sections shown in the following table.

Table 28. IBM MQ Accounting Class 3 report sections and where they occur

Section	List report	Summary report: by transaction	Summary report: by transaction, queue	Summary report: by queue	Summary report: by queue, transaction
Task identification	Yes	Yes	Yes	–	Yes
Task-related statistics	Yes	Yes	Yes	–	Yes
Task in-MQ time	Yes	Yes	Yes	–	Yes
Queue identification	Yes	–	Yes	Yes	Yes
Queue call statistics	Yes	–	Yes	Yes	Yes
Queue get/put summary	Yes	–	Yes	Yes	Yes
Queue in-MQ time	–	–	Yes	Yes	Yes

In these reports, a *task* is a unique combination of IBM MQ subsystem name (SSID), connection type and name, and, for some connection types (such as IMS or CICS), transaction ID.

The following figures show how the sections are organized into repeating structures in each report.

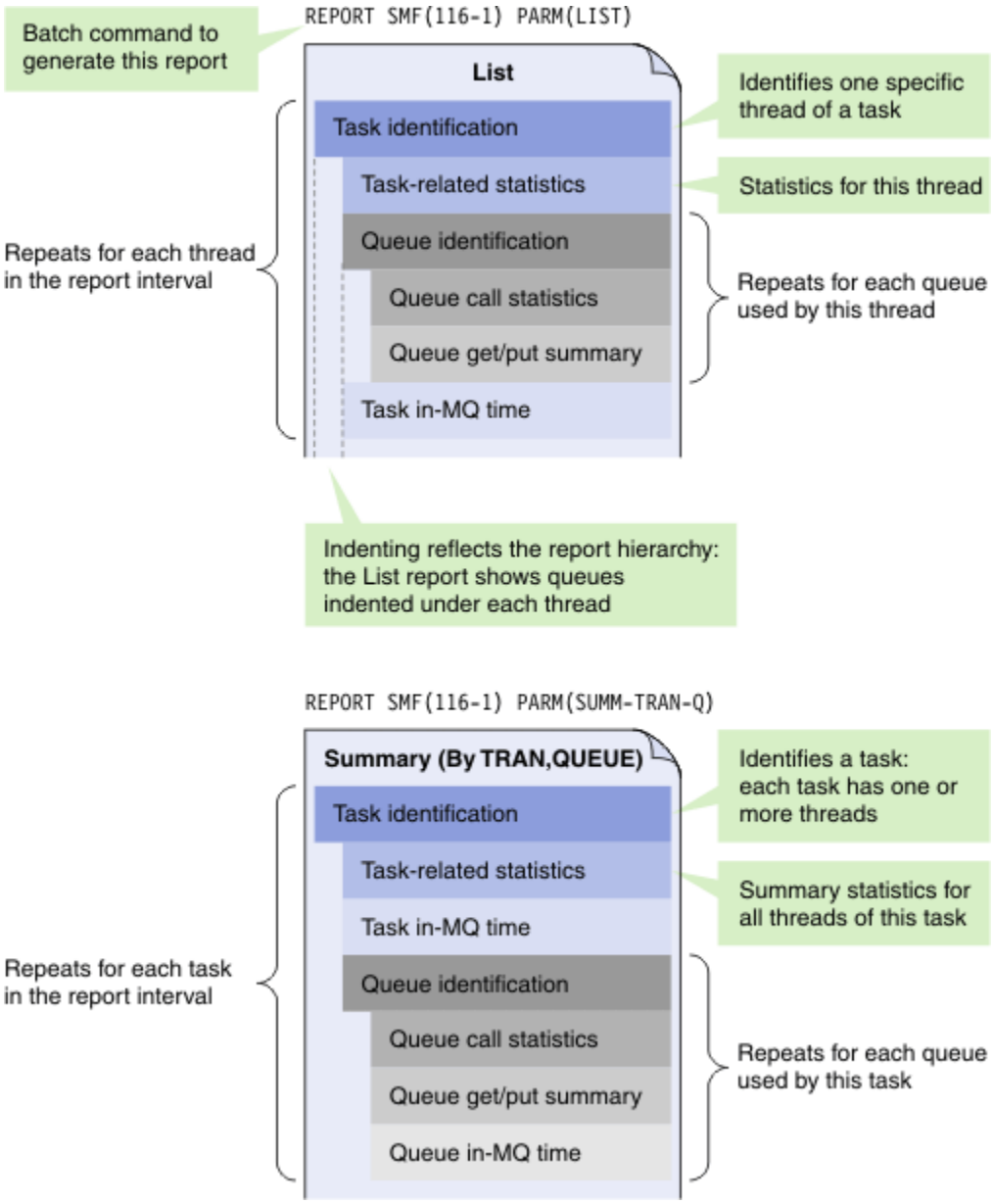


Figure 126. IBM MQ Accounting Class 3 report structures (1 of 2)

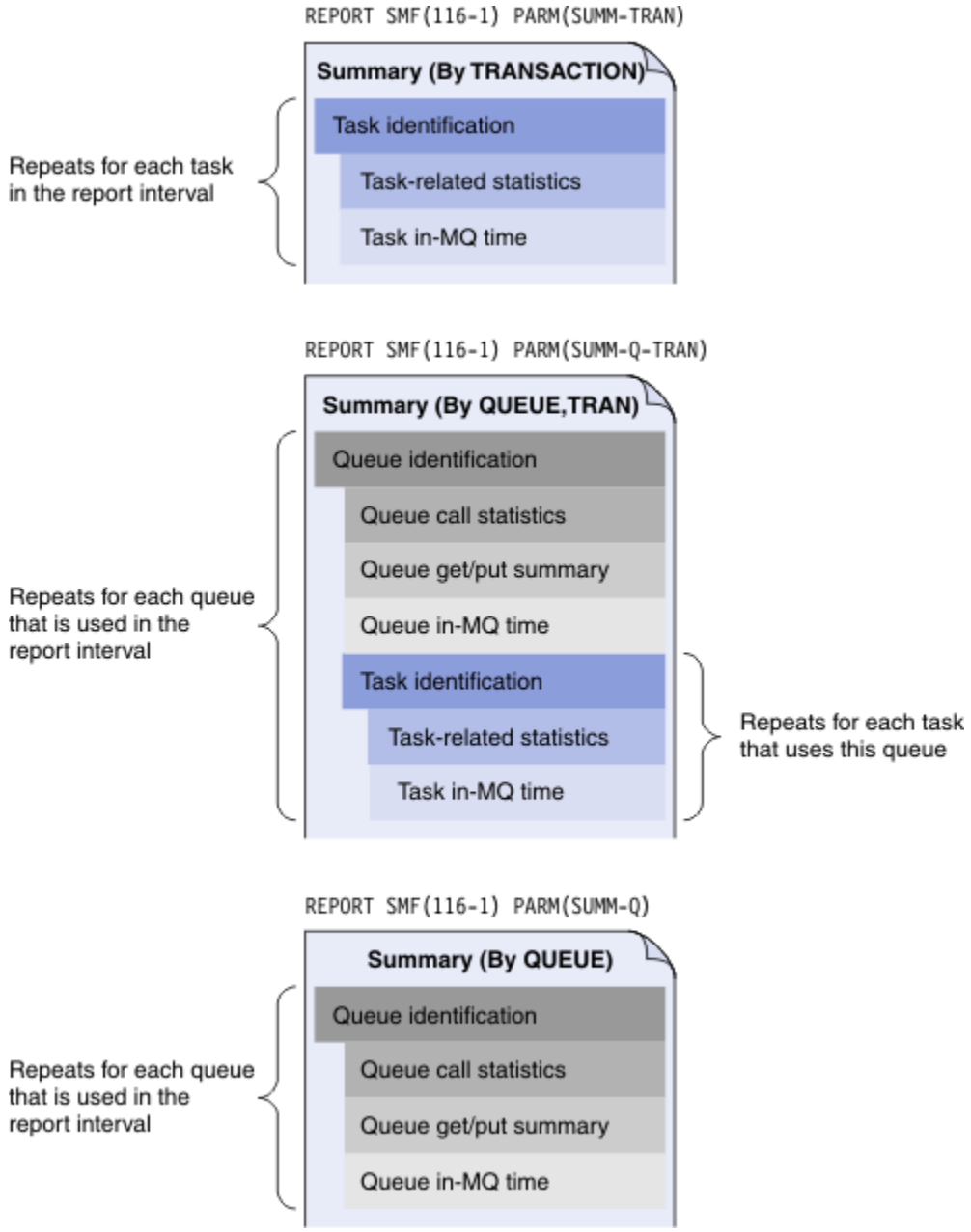


Figure 127. IBM MQ Accounting Class 3 report structures (2 of 2)

The indenting of a section indicates the scope of its data. For example:

- In a "Summary (By QUEUE, TRAN)" report, the task-related statistics sections contain data for a particular task, for a particular queue.
- In a "Summary (By TRAN, QUEUE)" report, the task-related statistics sections contain data for a particular task across all queues.

Task identification

```
SSID: SYSU Type: CICS    Name: CICSSYSN Tran: TRAL     User: CICSSYSN Task:  16067 NetName: N/A              UOWID: N/A
Channel:                      Channel Connection:                                                 Start:  3/03/2010
21:59:54.66
:
```

In list reports, this section identifies a particular thread of a task.

- IBM MQ subsystem name (SSID) – extracted from the instrumentation standard header (described by assembler macro CSQDQWHS in the IBM MQ library SCSQMACS)
- Connection details (such as connection type; connection name; and details specific to a connection type, such as CICS applid and transaction code) – extracted from the instrumentation correlation data (macro CSQDQWHC)
- User ID, netname, NETUOW, channel, channel connection – extracted from the task identification block (macro CSQDWTID)

Column heading	Description	Field
SSID	IBM MQ subsystem name.	QWHSSSID
Type	**QMGR** Queue manager **CICS** CICS **BAT/TSO** Batch or TSO **IMSCTL** IMS control region **IMSDEP** IMS MPP or BMP **CMDSERV** Command server **CHIN** Channel initiator **RRSBAT** RRS batch **UNKNOWN** None of the preceding connection types	WTIDNID
Name	Connection name. The value depends on the connection type: **CICS** CICS region applid **IMSDEP or IMSCTL** IMS subsystem ID Blank for other connection types.	WTIDCORI
PSB	IMSDEP connection type only: PSB name.	WTIDCORI
Tran	CICS connection type only: CICS transaction code, extracted from the IBM MQ correlation ID.	WTIDCORI
Task	CICS connection type only: CICS task number, extracted from the IBM MQ correlation ID.	WTIDCORI
User	User (or Operator) ID.	WTIDOPID
NetName	Network name, extracted from the IBM MQ accounting token.	WTIDACCT
UOWID	Network unit of work ID, extracted from the IBM MQ accounting token.	WTIDACCT
Channel	Channel name for MVS mover.	WTIDCHL

Column heading	Description	Field
Channel Connection	Long connection name for MVS mover.	WTIDCHLC
Start *(List reports only)*	IBM MQ thread start time stamp.	WTASINTS
Threads *(Summary reports only)*	Number of threads for this task.	*(Value is accumulated)*

Task-related statistics

```
  COMMIT    Avg Count              4.5   Avg Elapsed    0.016087  Avg CPU      0.000130
  Other     Avg Count              9.0   Avg Elapsed    0.002000  Avg CPU      0.000928
  Jnl/Log   Avg Bytes          10594.0   Avg FORCEs           4.5  Avg WAIT Elp 0.015065   Avg SUSPEND Elp  0.015967
⋮
```

This section contains commit, backout, journal and logging, page set 00 logging, DB2 Manager, CF Manager and "Other" statistics – extracted from the task-related statistics (macro CSQDWTAS).

In a list report, this section shows individual values for each thread of a task.

In summary reports, unless otherwise stated in the following descriptions, this section shows *averages per thread* for a task. The "Number of threads" value in the following tables depends on where this section appears in the report. For example, in a "Summary (By QUEUE, TRANS)" report, this section appears indented under a queue, so the "Number of threads" is the number of threads of a particular task and for a particular queue. In a "Summary (By TRANS)" report , "Number of threads" is the number of threads of a particular task across all queues.

This section is divided into the types of activity performed by the task, shown in the following list. If a task does not perform an activity, this section omits statistics for that activity. For example, if the task performs no backouts, then no backout statistics are shown.

Commit

List report:

Field label	Description	Field
Count	Number of commit requests by this thread.	WTASCMN
Avg Elapsed	Average commit elapsed time for this thread.	WTASCMET / WTASCMN
Avg CPU	Average commit CPU time for this thread.	WTASCMCT / WTASCMN

Summary reports:

Avg Count	Average number of commit requests per thread.	Sum(WTASCMN) / Number of threads
Avg Elapsed	Average commit elapsed time per thread.	Sum(WTASCMET) / Number of threads
Avg CPU	Average commit CPU time per thread.	Sum(WTASCMCT) / Number of threads

Backout

List report:

Count	Number of backout calls by this thread.	WTASBAN
Avg Elapsed	Average backout elapsed time for this thread.	WTASBAET / WTASBAN

Avg CPU	Average backout CPU time for this thread.	WTASBACT / WTASBAN

Summary reports:

Avg Count	Average number of backout calls per thread.	Sum(WTASBAN) / Number of threads
Avg Elapsed	Average backout elapsed time per thread.	Sum(WTASBAET) / Number of threads
Avg CPU	Average backout CPU time per thread.	Sum(WTASBACT) / Number of threads

P/S 0

Page set 00 logging activity.

List report:

Count	Number of logging requests by this thread.	WTASPSN0
Avg Elapsed	Average logging request elapsed time for this thread.	WTASPSE0 / WTASPSN0
Avg CPU	Average backout CPU time for this thread.	WTASBACT / WTASBAN

Summary reports:

Avg Count	Average number of logging requests per thread.	Sum(WTASPSN0) / Number of threads
Avg Elapsed	Average logging request elapsed time per thread.	Sum(WTASPSE0) / Number of threads
Avg CPU	Average backout CPU time per thread.	Sum(WTASBACT) / Number of threads

Latch

List reports:

Max Elapsed Class	Latch class number (relative entry in WTASLWN array) that had the maximum elapsed latch wait time (see Max Elapsed later in this table).	WTASMLWN
Max Elapsed	Maximum elapsed wait time for a single latch class.	WTASMLW
Tot Elapsed	Total elapsed wait time for all latch classes.	Sum(WTASLWET array entries)
Max Count Class	Latch class number that had the maximum number of waits (see Max Count later in this table).	
Max Count	Maximum number of waits for a single latch class.	Max(WTASLWN array entries)
Tot Count	Total number of waits for all latch classes.	Sum(WTASLWN array entries)

Summary reports:

Max Elapsed Class	Latch class number (relative entry in WTASLWN array) that had the maximum elapsed latch wait time (see Max Elapsed later in this table).	WTASMLWN
Max Elapsed	Maximum elapsed wait time for a single latch class.	Max(WTASMLW)
Avg Elapsed	Average elapsed wait time per thread across all latch classes.	Sum(All WTASLWET array entries) / number. of threads

Max Count Class	Latch class number that had the maximum number of waits (see Max Count later in this table).	
Max Count	Maximum number of waits for a single latch class.	Max(WTASLWN array entries)
Avg Count	Average number of waits per thread across all latch classes.	Sum(All WTASLWN array entries) / number of threads

Other

Non-queue other statistics.

List reports:

Total Calls	Total number of "Other" calls by this thread.	WTASOTN
Avg Elapsed	Average elapsed time per "Other" call for this thread.	WTASOTET / WTASOTN
Avg CPU	Average CPU time per "Other" call for this thread.	WTASOTCT / WTASOTN
#Old Pages	Number of old pages retrieved by this thread.	WTASGPO
#New Pages	Number of new pages retrieved by this thread.	WTASGPN

Summary reports:

Avg Count	Average number of "Other" calls per thread.	Sum(WTASOTN)
Avg Elapsed	Average elapsed time per "Other" call per thread.	Sum(WTASOTET) / Number of threads
Avg CPU	Average CPU time per "Other" call per thread.	Sum(WTASOTCT) / Number of threads
Avg #Old Pages	Average number of old pages retrieved per thread.	Sum(WTASGPO) / Number of threads
Avg #New Pages	Average number of new pages retrieved per thread.	Sum(WTASGPN) / Number of threads

Jnl/Log

List report:

Bytes	Total number of bytes written to the journal for this thread.	WTASJWB
FORCEs	Total number of times the log was forced for this thread.	WTASJCN
Avg WAIT Elp	Average elapsed time waiting for the log to be forced for this thread.	WTASJCET / WTASJCN
Avg SUSPEND Elp	Average suspend time for this thread.	WTASSUSE / WTASJCN

Summary reports:

Avg Bytes	Average number of bytes written to the journal per thread.	Sum(WTASJWB) / Number of threads
Avg FORCEs	Average number of times the log was forced per thread.	Sum(WTASJCN) / Number of threads
Avg WAIT Elp	Average elapsed time waiting for the log to be forced per thread.	Sum(WTASJCET) / Number of threads

Avg SUSPEND Elp	Average suspend time per thread.	Sum(WTASSUSE) / Number of threads

DB2 Mgr

List report:

Requests	Total number of DB2 calls.	WTASDBCT
Avg Jnl/Log Thread Elapsed	Average elapsed time per DB2 call.	WTASDBET / WTASDBCT
Avg Jnl/Log Server Elapsed	Average server elapsed time per DB2 call.	WTASDBES / WTASDBCT
Jnl/Log Thd Elp (Max)	Maximum DB2 thread elapsed time.	WTASDBMT
Jnl/Log Svr Elp (Max)	Maximum DB2 server elapsed time.	WTASDBMS

Summary reports:

Requests	Average number of DB2 calls per thread.	Sum(WTASDBCT) / Number of threads
Avg Jnl/Log Thread Elapsed	Average elapsed time per DB2 call per thread.	Sum(WTASDBET) / Number of threads
Avg Jnl/Log Server Elapsed	Average server elapsed time per DB2 call per thread.	Sum(WTASDBES) / Number of threads
Jnl/Log Thd Elp (Max)	Maximum DB2 thread elapsed time.	WTASDBMT
Jnl/Log Svr Elp (Max)	Maximum DB2 server elapsed time.	WTASDBMS

CF Mgr

Coupling facility manager activity.

List report:

IXLLSTE: Count	Total number of IXLLSTE calls.	WTASCSEC
IXLLSTE: Redrives	Total number of IXLLSTE redrives.	WTASRSEC
IXLLSTM: Count	Total number of IXLLSTM calls.	WTASCMEC
IXLLSTM: Redrives	Total number of IXLLSTM redrives.	WTASRMEC

Summary reports:

IXLLSTE: Count	Average number of IXLLSTE calls per thread.	Sum(WTASCSEC) / Number of threads
IXLLSTE: Redrives	Average number of IXLLSTE redrives per thread.	Sum(WTASRSEC) / Number of threads
IXLLSTM: Count	Average number of IXLLSTM calls per thread.	Sum(WTASCMEC) / Number of threads

IXLLSTM: Redrives	Average number of IXLLSTM redrives per thread.	Sum(WTASRMEC) / Number of threads

Queue identification

```
  Queue: TRR.APPLICATION_ALERT                          (SYSA                                                  )
  QType: REMOTE   IType: NONE       GDisp: Q_MGR    Date:  3/03/2010  Time: 09:00:40  P/Set No:        1  BufferPool
No:        1
  First Opened:  3/03/2010 22:00:40.28   Last Closed:  3/03/2010 22:00:40.28   CF Structure Name:
⋮
```

This section contains the queue name, type, and other identifiers – extracted from the identification section at the start of the queue statistics (macro CSQDWQ).

Field label	Description	Field
Queue	Queue name as specified in OD of MQOPEN request. (Where applicable, followed by parentheses containing the base queue name to which OBJNAME resolved.)	OBJNAME (Where applicable, followed by BASENAME)
QType	Type of queue.	QTYPE
IType	Index type of queue.	INDXTYPE
GDisp	Queue-sharing-Group disposition.	QSGDISP
Date ***(List report only)***	Date from the SMF record time stamp.	SMF116DTE
Time ***(List report only)***	Time from the SMF record time stamp.	SMF116TME
P/Set No ***(List report only)***	Page set number.	NPS
Bufferpool No ***(List report only)***	Buffer pool number.	NBUFFPOOL
First Opened ***(List report only)***	Time queue was first opened.	OPENTIME
Last Closed ***(List report only)***	Time queue was last closed.	CLOSTIME
CF Structure Name ***(List report only)***	Coupling Facility structure name.	CFSTRUCNAME
QCount	Number of IBM MQ accounting records in which a transaction referenced the key for this queue.	

Queue call statistics

```
                Count     Elapsed      CPU      Susp Elp  JnlWrt Elp   PS Req's  PS Rd Elp   Expired   Page Skip  Msgs
Skip
             ---------  ---------  ---------  ---------  ----------  ---------  ---------  ---------  ---------
---------
    PUT1             1   0.000117   0.000117          0           0        0.0          0
⋮
```

This section contains OPEN, CLOSE, GET, PUT, PUT1, INQ, SET and OTHER statistics – extracted from the queue statistics (macro CSQDWQ).

This section is divided into the types of queue activity, shown in the following list. If a type of activity does not occur on a queue, this section omits statistics for that activity.

OPEN

List report:

Column heading	Description	Field
Count	Total number of OPEN calls.	OPENN
Elapsed	Average elapsed time per OPEN call.	OPENET / OPENN
CPU	Average CPU time per OPEN call.	OPENCT / OPENN

Summary reports:

Count	Average number of OPEN calls per queue count.	Sum(OPENN) / QCount
Elapsed	Average elapsed time per OPEN call.	Sum(OPENET) / Sum(OPENN)
CPU	Average CPU time per OPEN call.	Sum(OPENCT) / Sum(OPENN)

CLOSE

List report:

Count	Total number of CLOSE calls.	CLOSEN
Elapsed	Average elapsed time per OPEN call.	CLOSEET / CLOSEN
CPU	Average CPU time per CLOSE call.	CLOSECT / CLOSEN

Summary reports:

Count	Average number of CLOSE calls per queue count.	Sum(CLOSEN) / QCount
Elapsed	Average elapsed time per OPEN call.	Sum(CLOSEET) / Sum(CLOSEN)
CPU	Average CPU time per CLOSE call.	Sum(CLOSECT) / Sum(CLOSEN)

GET

List report:

Count	Total number of GET calls, according to the following types of GET call: **DES ANY** Destructive GET ANY **DES SPE** Destructive GET SPECIFIC **BRW ANY** BROWSE ANY **BRW SPE** BROWSE SPECIFIC	GETN
Elapsed	Average elapsed time per GET call.	GETET / GETN
CPU	Average CPU time per GET call.	GETCT / GETN
Susp Elap	Average suspend time per GET call.	GETSUSET / GETN

JnlWrt Elp	Average elapsed time waiting for a journal write per GET call.	GETJWET / GETN
PS Req's	Average number of reads from a Page Set per GET call.	GETPSN / GETN
PS RD Elp	Average elapsed time waiting for a read from a Page Set per GET call.	GETPSET / GETN
Expired	Average number of expired messages.	GETEXMSG / GETN
Page Skip	Average number of pages skipped processing a GET.	GETEPAGE / GETN
Msgs Skip	Average number of messages skipped processing a GET.	GETSMSG / GETN

Summary report:

Count	Average number of GET calls per queue count, according to the following types of GET call: **DES ANY** Destructive GET ANY **DES SPE** Destructive GET SPECIFIC **BRW ANY** BROWSE ANY **BRW SPE** BROWSE SPECIFIC	Sum(GETN) / QCount
Elapsed	Average elapsed time per GET call.	Sum(GETET) / Sum(GETN)
CPU	Average CPU time per GET call.	Sum(GETCT) / Sum(GETN)
Susp Elap	Average suspend time per GET call.	Sum(GETSUSET) / Sum(GETN)
JnlWrt Elp	Average elapsed time waiting for a journal write per GET call.	Sum(GETJWET) / Sum(GETN)
PS Req's	Average number of reads from a Page Set per GET call.	Sum(GETPSN) / Sum(GETN)
PS RD Elp	Average elapsed time waiting for a read from a Page Set per GET call.	Sum(GETPSET) / Sum(GETN)
Expired	Average number of expired messages.	Sum(GETEXMSG) / Sum(GETN)
Page Skip	Average number of pages skipped processing a GET.	Sum(GETEPAGE) / Sum(GETN)
Msgs Skip	Average number of messages skipped processing a GET.	Sum(GETSMSG) / Sum(GETN)

PUT, PUT1

(In the following tables, *n* is 1 for PUT1 and an empty string for PUT: for example, PUT*n*N represents PUT1N and PUTN.)

List report:

Count	Total number of PUT*n* calls.	PUT*n*N
Elapsed	Average elapsed time per PUT*n* call.	PUT*n*ET / PUT*n*N
CPU	Average CPU time per PUT*n* call.	PUT*n*CT / PUT*n*N
Susp Elap	Average suspend time per PUT*n* call.	PUT*n*SUSET / PUT*n*N

JnlWrt Elp	Average elapsed time waiting for a journal write per PUT*n* call.	PUT*n*JWET / PUT*n*N
PS Req's	Average number of PUT*n* calls from a Page Set per PUT*n* call.	PUT*n*PSN / PUT*n*N
PS RD Elp	Average elapsed time waiting for a read from a Page Set per PUT*n* call.	PUT*n*PSET / PUT*n*N

Summary reports:

Count	Average number of PUT*n* calls per queue count.	Sum(PUT*n*N) / QCount
Elapsed	Average elapsed time per PUT*n* call.	Sum(PUT*n*ET) / Sum(PUT*n*N)
CPU	Average CPU time per PUT*n* call.	Sum(PUT*n*CT) / Sum(PUT*n*N)
Susp Elap	Average suspend time per PUT*n* call.	Sum(PUT*n*SUSET) / Sum(PUT*n*N)
JnlWrt Elp	Average elapsed time waiting for a journal write per PUT*n* call.	Sum(PUT*n*JWET) / Sum(PUT*n*N)
PS Req's	Average number of PUT calls from a Page Set per PUT*n* call.	Sum(PUT*n*PSN) / Sum(PUT*n*N)
PS RD Elp	Average elapsed time waiting for a read from a Page Set per PUT*n* call.	Sum(PUT*n*PSET) / Sum(PUT*n*N)

INQ

List report:

Count	Total number of INQ calls.	INQN
Elapsed	Average elapsed time per INQ call.	INQET / INQN
CPU	Average CPU time per INQ call.	INQCT / INQN

Summary reports:

Count	Average number of INQ calls per queue count.	Sum(INQN) / QCount
Elapsed	Average elapsed time per INQ call.	Sum(INQET) / Sum(INQN)
CPU	Average CPU time per INQ call.	Sum(INQCT) / Sum(INQN)

SET

List report:

Count	Total number of SET calls.	SETN
Elapsed	Average elapsed time per SET call.	SETET / SETN
CPU	Average CPU time per SET call.	SETCT / SETN

Summary reports:

Count	Average number of SET calls per queue count.	Sum(SETN) / QCount
Elapsed	Average elapsed time per SET call.	Sum(SETET) / Sum(SETN)
CPU	Average CPU time per SET call.	Sum(SETCT) / Sum(SETN)

Queue get/put summary

```
    PUT1     Total Bytes          59  #PUT w/Data             1  Min Msg Size        59  Max Msg Siz         59
⋮
```

This section contains additional summary information about GET and PUT calls – extracted from the end of the queue statistics (macro CSQDWQ).

GET

List report:

Field label	Description	Field
Total Bytes	Total number of data bytes read during MQGET.	GETBYTES
#GET w/Data	Total number of successful GET calls.	VALIDGET
Min Msg Size	Minimum message size retrieved by GET calls.	GETMINMS
Max Msg Size	Maximum message size retrieved by GET calls.	GETMAXMS

Summary reports:

Avg Bytes	Average number of data bytes read during MQGET per queue count.	Sum(GETBYTES) / QCount
Avg #GET w/ Data	Average number of successful GET calls per queue count.	Sum(VALIDGET) / QCount
Min Msg Size	Minimum message size retrieved by GET calls.	GETMINMS
Max Msg Size	Maximum message size retrieved by GET calls.	GETMAXMS

PUT

List report:

Total Bytes	Total number of data bytes read during PUT1.	PUTBYTES
#PUT w/Data	Total number of successful PUT calls.	VALIDPUT
Min Msg Size	Minimum message size retrieved by PUT calls.	PUTMINMS
Max Msg Size	Maximum message size retrieved by PUT calls.	PUTMAXMS

Summary reports:

Avg Bytes	Average number of data bytes read during PUT1 per queue count.	Sum(PUTBYTES) / QCount
Avg #PUT w/ Data	Average number of successful PUT calls per queue count.	Sum(VALIDPUT) / QCount
Min Msg Size	Minimum message size retrieved by PUT calls.	PUTMINMS
Max Msg Size	Maximum message size retrieved by PUT calls.	PUTMAXMS

Task and queue in-MQ time

```
  In-MQ Time (Total)      Elapsed:  0.098241   CPU:   0.002117
  In-MQ Time (Average)    Elapsed:  0.049120   CPU:   0.001058
⋮
```

This section is included under each task and under each queue.

The "In-MQ Time (Total)"' is displayed once for each SMF type 116, subtype 1 record reported, and once for each subsequent subtype 2 record that belongs to the same task.

List report:

Field label	Description and fields
(Total) Elapsed	Total elapsed time spent in IBM MQ by this thread. In a List report, task-related in-MQ elapsed times consist of the following components: **WTASCMET** Commit calls elapsed time **WTASBAET** Backout calls elapsed time **WTASPSE0** Page set zero logging elapsed time **Sum(latch elapsed)** Sum of the elapsed time for all latch classes for the task **WTASOTET** "Other" calls elapsed time **WTASJCET** Journal wait elapsed time **WTASSUSE** Suspend elapsed time **WTASDBET** DB2 elapsed time for thread **WTASDBES** DB2 elapsed time for server In a List report, queue-related in-MQ elapsed times consist of the following components: **OPENET** MQOpen calls elapsed time **CLOSEET** MQClose calls elapsed time **GETET** MQGet calls elapsed time **GETSUSET** MQGet calls suspend elapsed time **GETJWET** MQGet calls elapsed time waiting for journal writes to complete **GETPSET** MQGet calls elapsed time waiting for a pageset read **PUT*n*ET** MQPut*n* calls elapsed time **PUT*n*SUSET** MQPut*n* calls suspend elapsed time **PUT*n*JWET** MQPut*n* calls elapsed time waiting for journal writes to complete **PUT*n*PSET** MQPut*n* calls elapsed time waiting for a pageset **INQET** MQInq calls elapsed time **SETET** MQSet calls elapsed time

Field label	Description and fields
(Total) CPU	Total CPU time spent in IBM MQ by this thread. In a List report, task-related in-MQ CPU times consist of the following components: **WTASCMCT** Commit calls CPU time **WTASBACT** Backout calls CPU time **WTASOTCT** "Other" calls CPU time In a List report, queue-related in-MQ CPU times consist of the following components: **OPENCT** MQOpen calls CPU time **CLOSECT** MQClose calls CPU time **GETCT** MQGet calls CPU time **PUT*n*CT** MQPut*n* calls CPU time **INQCT** MQInq calls CPU time **SETCT** MQSet calls CPU time (where *n* is 1 for PUT1 and an empty string for PUT: for example, PUT*n*CT represents PUT1CT and PUTCT)

Summary reports:

(Total and Average) Elapsed	Total and average elapsed time spent in IBM MQ by these threads. In Summary reports, task-related in-MQ elapsed times consist of the following components: **Sum(COMMIT elapsed)** Sum of elapsed time for all commit calls **Sum(BACKOUT elapsed)** Sum of elapsed time for all backout calls **Sum(PS0 elapsed)** Sum of page set zero elapsed time **Sum(Latch elapsed)** Sum of all latch elapsed time (across all classes) for all tasks **Sum(Other elapsed)** Sum of all "other" calls elapsed time **Sum(Journal elapsed)** Sum of all journal wait elapsed time **Sum(Suspend elapsed)** Sum of all suspend elapsed time **Sum(DB2 elapsed: thread)** Sum of all DB2 elapsed time for all threads **Sum (DB2 elapsed: server)** Sum of all DB2 elapsed time for the server In Summary reports, queue-related in-MQ elapsed times consist of the following components: **Sum(OPEN elapsed)** Sum of all MQOpen calls elapsed time **Sum(CLOSE elapsed)** Sum of all MQClose calls elapsed time **Sum(GET elapsed)** Sum of all MQGet calls elapsed time **Sum(GET suspend)** Sum of all MQGet calls suspend elapsed time **Sum(GET journal write)** Sum of all MQGet calls elapsed time waiting for journal writes to complete **Sum(GET pageset)** Sum of all MQGet calls elapsed time waiting for a pageset **Sum(PUT*n* elapsed)** Sum of all MQPut*n* calls elapsed time **Sum(PUT*n* suspend)** Sum of all MQPut*n* calls suspend elapsed time **Sum(PUT*n* journal write)** Sum of all MQPut*n* calls elapsed time waiting for journal writes to complete **Sum(PUT*n* pageset)** Sum of all MQPut*n* calls elapsed time waiting for a pageset **Sum(INQ elapsed)** Sum of all MQInq calls elapsed time **Sum(SET elapsed)** Sum of all MQSet calls elapsed time

(Total and Average) CPU	Total and average CPU time spent in IBM MQ by these threads. In Summary reports, task-related in-MQ CPU times consist of the following components: **Sum(COMMIT CPU)** Sum of CPU time for all commit calls **Sum(BACKOUT CPU)** Sum of CPU time for all backout calls **Sum(Other CPU)** Sum of all "other" calls CPU time In Summary reports, queue-related in-MQ CPU times consist of the following components: **Sum(OPEN CPU)** Sum of all MQOpen calls CPU time **Sum(CLOSE CPU)** Sum of all MQClose calls CPU time **Sum(GET CPU)** Sum of all MQGet calls CPU time **Sum(PUT*n* CPU)** Sum of all MQPut*n* calls CPU time **Sum(INQ CPU)** Sum of all MQInq calls CPU time **Sum(SET CPU)** Sum of all MQSet calls CPU time

Average in-MQ times appear only in Summary reports, not List reports.

In Summary reports:

- Total in-MQ times are a grand total of the total in-MQ times for each thread, not an average of the total in-MQ times for each thread.
- Similarly, average in-MQ times are calculated from a grand total of the in-MQ times for each thread divided by the number of threads, rather than being an average of the average in-MQ times for each thread.

Chapter 75. Report and extract utility

The report and extract utility processes logs to create reports and extracts; convert logs to comma-separated values (CSV) or or JavaScript Object Notation (JSON); runs REXX execs that process log files; and exports or imports controls (filters, forms, and object lists) between control repositories.

Related concepts

Report and extract utility

The report and extract utility is a batch program that creates reports, extracts for further processing with Transaction Analysis Workbench, and JavaScript Object Notation (JSON) or comma-separated values (CSV) files for use with other applications. The Transaction Analysis Workbench ISPF dialog generates JCL to run the utility.

Related reference

CICS-DBCTL report JCL

To create CICS-DBCTL reports, use the **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands of the report and extract utility. The multi-step JCL presented here generates all CICS-DBCTL reports.

DB2 accounting exception report and extract JCL

To create DB2 accounting exception reports and extracts, use the **REPORT DB2X** command of the report and extract utility.

SMF report JCL

To create an SMF report, use the **SMF** parameter of the **REPORT** command of the report and extract utility.

Report and extract utility JCL

You can use the Transaction Analysis Workbench ISPF dialog to create JCL for the utility, or you can write the JCL yourself.

Example: Formatted record report from an IMS log file

The following JCL reports all IMS Application Termination Statistics (type 07) log records where the transaction ID matches the pattern "STOK*". A version of this JCL is supplied in member FUWIMS01 of the sample library SFUWSAMP.

In this example, the LOGRPT DD statement and corresponding OUTPUT(LOGRPT) parameter of the **REPORT** command are unnecessary. If you omit them, the result would be identical: the report goes to the default sysout data set LOGRPT.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW      EXEC PGM=FUWBATCH,PARM='V<IMS version>'
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//LOGIN    DD  DISP=SHR,DSN=<input.IMS.log>
//LOGRPT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
REPORT OUTPUT(LOGRPT)
  CODE(IMS:07)
    COND TRANCODE EQ 'STOK*'
/*
```

Figure 128. JCL for the report and extract utility: using an IMS log file

Example: OPERLOG report

The following JCL reports all RACF (ICH prefix) messages written today to OPERLOG. CA52 is the Transaction Analysis Workbench log code for OPERLOG records. There is no DD statement such as LOGIN to identify an input log file; instead, the **LOGSTREAM** command in the SYSIN data set identifies OPERLOG.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW      EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
LOGSTREAM OPERLOG:SYSPLEX.OPERLOG
START 0
REPORT OPERLOG
  CODE(MVS:CA52)
    COND TEXT(2) EQ 'ICH'
/*
```

Figure 129. JCL for the report and extract utility: using an OPERLOG

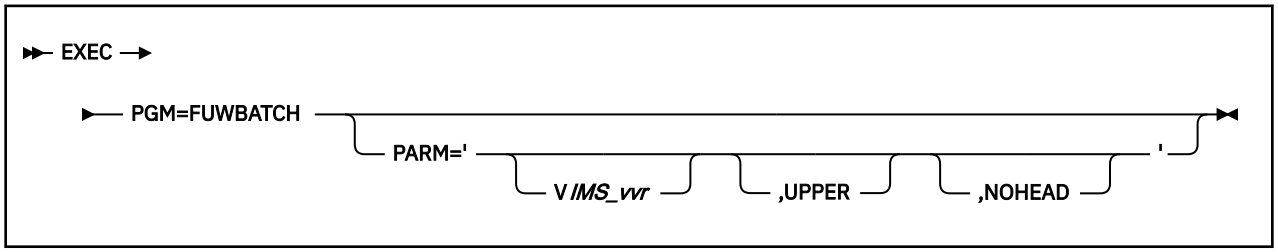
The JCL for the report and extract utility consists of the following statements. If you use the Transaction Analysis Workbench ISPF dialog to generate the JCL, the ISPF dialog uses your profile settings (option 0 **Profile**) to complete the statements.

Job statement

If you use the ISPF dialog to generate the JCL, the JCL contains the value of the **Job Statement Information** field under ISPF dialog option 0.1 **Workbench Personal Settings**.

EXEC statement

The **EXEC** statement runs the Transaction Analysis Workbench report and extract utility program, FUWBATCH.



V*IMS_vvr*
For jobs that read IMS logs, you must specify the IMS version of the IMS subsystem that created the log files: either with this parameter or with an **IMSVRM** command in the SYSIN data set.

For example, for IMS V15.4, specify PARM='V154'.

If you use the ISPF dialog to generate the JCL, the JCL contains the value that you specified for the input file; if you did not specify a value, the JCL contains the value of the **IMS Release** field under ISPF dialog option 0.5 **IMS Tools Settings**.

UPPER
Optional.

Requests report output in upper case. The default is mixed case (**UPPER** omitted). For example, specify **UPPER** if your printer does not support mixed case.

If you use the ISPF dialog to generate the JCL, the JCL contains the value of the **Reports in Upper Case** field under ISPF dialog option 0.1 **Workbench Personal Settings**; if the field value is Yes, the JCL contains the **UPPER** parameter.

NOHEAD
Optional.

Omits page headings from the report.

To specify this option from the ISPF dialog, select **Edit JCL before submit**, then add this value to the **EXEC PARM** before submitting the JCL.

Miscellaneous input DD statements

STEPLIB
The library (SFUWLINK) containing the Transaction Analysis Workbench executable load modules. You do not need to specify the name of this library if the modules reside in the system LNKLST.

If you use the ISPF dialog to generate the JCL, the JCL contains the value of the **Workbench Load Library** field under ISPF dialog option 0.1 **Workbench Personal Settings**.

SYSIN
Data set that contains commands instructing the utility what to do. The Transaction Analysis Workbench ISPF dialog generates commands based on your input, or you can create the SYSIN data set yourself.

The SYSIN data set can contain comment lines, inline comments, and blank lines. A comment line begins with an asterisk (*) in column one. An inline comment begins after column one, and after any command on the line, with a slash followed by an asterisk (/*).

```
//SYSIN DD *
* Comment line
REPORT /* Inline comment

/*
```

FUWCDS
A control repository, which might be an IMS Problem Investigator control data set (CDS).

Required only if the SYSIN data set contains commands that refer to controls: filters, forms, object lists, or IMS user log records.

If you use the ISPF dialog to generate the JCL, the JCL contains the value of the **Control** (problem analysis) field under ISPF dialog option 0.2 **Repositories**.

FUWPROBR
A session repository.

Required only if the SYSIN data set contains a **SESSION** command.

If you use the ISPF dialog to generate the JCL, the JCL contains the value of the **Session** (problem analysis) field under ISPF dialog option 0.2 **Repositories**.

DD statements for input logs

The following DD statements specify the log files to be used as input to the report and extract utility.

If the SYSIN data set contains a **LOGSTREAM** command, the report and extract utility ignores these DD statements, and instead uses the specified log stream.

The utility recognizes the following ddnames for input log files. Follow the conventions described here to match ddnames to particular file types.

LOGIN, LOGIN*nnn*, L*xxxxnnn*
IMS log files.

If you use any of these L-prefix ddnames, you must use one of the following methods to identify the IMS release that created the IMS log files:

- PARM='V*IMS_vvr*' parameter on the **EXEC** statement
- **IMSVRM** command in the SYSIN data set

Even if the files to which these ddnames refer are not IMS log files, you must still specify an IMS release; for this reason, it is recommended that you use these L-prefix ddnames only for IMS log files.

For job steps that use only one IMS log file, use the ddname LOGIN or LOGIN001.

For job steps that use multiple IMS log files to be processed sequentially, use ddnames LOGIN*nnn*, where *nnn* is a 3-digit number with leading zeros that reflects the chronological order of the log

files. Higher numbers reflect more recent data. For example, if LOGIN001 contains log records for yesterday, then LOGIN002 should contain records for today.

For job steps that uses multiple IMS log files to be processed as a merged sequence of records in time stamp order (for example, from a shared-queue or data sharing environment), use ddnames L*xxxxnnn*. Files with the same value of *xxxx* are processed sequentially, according to their *nnn* suffix. Files with different values of *xxxx* are processed in parallel. Typically, *xxxx* is the ID of the IMS subsystem that created the log file.

For example, suppose you have a shared-queue environment with two IMS subsystems, IMSA and IMSB. For each subsystem, you have a sequence of three log files that you want to process, as shown in the following DD statements:

```
//LIMSA001 DD ...
//LIMSA002 DD ...
//LIMSA003 DD ...
//LIMSB001 DD ...
//LIMSB002 DD ...
//LIMSB003 DD ...
```

The report and extract utility begins by opening LIMSA001 and LIMSB001, and then merging the files so that their records are in time stamp sequence. When the utility has copied the last record of LIMSA001 into the merged sequence, it opens LIMSA002; similarly, when the utility has copied the last record of LIMSB001, it opens LIMSB002. The time spans of the files from IMSA do not have to match the files from IMSB: for example, the utility might finish copying all records from LIMSB001, LIMSB002, and LIMSB003 into the merged sequence before it copies all (or any) records from LIMSA001.

CEXIN, CEXIN*nnn*
IMS Connect Extensions journals.

For a single file, use CEXIN or CEXIN001. For multiple files to be processed sequentially, use CEXIN001 - CEXIN999.

NTHIN, NTHIN*nnn*
Tivoli OMEGAMON XE for DB2 Performance Expert near-term history.

For a single file, use NTHIN or NTHIN001. For multiple files to be processed sequentially, use NTHIN001 - NTHIN999.

SMFIN, SMFIN*nnn*
SMF files.

For a single file, use SMFIN or SMFIN001. For multiple files to be processed sequentially, use SMFIN001 - SMFIN999.

INPUT, INPUT*nnn*
Extracts created by Transaction Analysis Workbench and for any other types of input log file not described by the ddnames listed previously: INPUT or INPUT001 for a single file; INPUT001 - INPUT999 for multiple files to be processed sequentially.

Output DD statements

SYSPRINT
Data set for Transaction Analysis Workbench messages and run-time information. This data set is required. You should check this data set for error messages. For an explanation of these error messages, see “FUW-prefixed messages” on page 326.

This data set is usually defined as SYSOUT=* or SYSOUT=A.

LOGRPT
Optional.

Default data set for report output.

To write a report to a different data set, specify the ddname in the **OUTPUT** parameter of the report and extract utility **REPORT** command.

You can request more than one report in a SYSIN data set; each report must specify a unique ddname.

If you specify a ddname in an **OUTPUT** parameter without a matching DD statement, the report goes to a sysout data set with that ddname, as if you had specified a DD statement with SYSOUT=*.

For example, if you specify a **REPORT** command without an **OUTPUT** parameter, and you do not specify a LOGRPT DD statement, then the report output goes to a sysout data set with the default ddname LOGRPT.

EXTRACT
Optional.

Default data set for extract output.

The ddname for extract output is specified in the **OUTPUT** parameter of the **EXTRACT** command. The default ddname is EXTRACT. You can request more than one extract, but each must specify a unique ddname.

CSV
Optional.

Default data set for CSV file output.

The ddname for CSV file output is specified in the **OUTPUT** parameter of the **CSV** command. The default ddname is CSV. You can request more than one CSV file, but each must specify a unique ddname.

JSON
Optional.

Default data set for JSON file output.

The ddname for JSON file output is specified in the **OUTPUT** parameter of the **JSON** command. The default ddname is JSON. You can request more than one JSON file, but each must specify a unique ddname.

Miscellaneous DD statements

The following DD statements are relevant only to the **EXPORT** and **IMPORT** commands:

AMAWORK1
Work data set used by the z/OS-supplied AMATERSE service aid program. The **EXPORT** command uses AMATERSE to create the tersed output file specified by the FUWTERSE DD statement.

AMAPRINT
Data set for messages from AMATERSE, if any.

FUWTERSE
Tersed output file created by the **EXPORT** command.

Related reference

CSV command examples
These examples present complete JCL for converting logs to CSV format.

JSON command examples
These examples present complete JCL for converting log data to JSON format.

Related information

FUW0041E

Input DD is missing (LOGIN or SMFIN)

Types of report and extract utility commands

The JCL to run the report and extract utility must specify a SYSIN data set that contains commands to the utility. There are several types of commands: global, action, qualifying, and administrative. You need to know which types of commands apply to the task you want to perform, and specify those commands in the appropriate order.

- Global commands affect the overall behavior of the report and extract utility.
- Action commands perform some action that produces output, such as a report. Typically, action commands process the logs that you have specified in the JCL.
- Qualifying commands restrict or add conditions to an action command. Qualifying commands affect only the preceding action command in the SYSIN data set.
- Administrative commands copy filters, forms, or object lists (collectively known as *controls*) between Transaction Analysis Workbench control repositories.

Processing logs

To process logs, specify commands in the following order:

1. Zero or more global commands.
2. An action command.
3. Zero or more qualifying commands.
4. Zero or more additional action commands, each followed by zero or more qualifying commands.

You can specify multiple action commands in a SYSIN data set, in any order, with one exception: a SYSIN data set can contain only one **REXX** command. A SYSIN data set that contains a **REXX** command cannot contain any other action commands.

Exporting and importing controls

To export or import controls, use the **EXPORT** or **IMPORT** administrative commands.

A SYSIN data set can contain only one administrative command. If specified, the administrative command must be the only command in the SYSIN data set.

Your choice of parameter syntax: *name*(*value*) or *name*=*value*

Some commands have parameters that specify values. For example, the **CSV** action command has, among other parameters, an **EOL** parameter that specifies the end-of-line delimiter in CSV output.

You can specify parameters with values in your choice of the following two forms:

***name*(*value*)**
: The parameter name, an opening parenthesis, the parameter value, and then a closing parenthesis. For example: EOL(LF)

name*=*value
: The parameter name, an equal sign, and then the parameter value. For example: EOL=LF

You can use whichever form you like, and you can mix the two forms in the same command. There is one exception: if a parameter value contains a space (embedded blank), then you must use the form with parentheses.

Typically, syntax diagrams and examples in this documentation show the form with parentheses.

Example: Creating a report

The following SYSIN data set creates a formatted record report of up to 999 pages for the specified interval, for log records that meet the specified filtering criteria. BANKTRAN is an object list of transaction IDs stored in the control repository:

```
PAGELIM(999) 1
START 2010-06-14-12.00
 STOP 2010-06-14-15.00
REPORT OUTPUT(LOGRPT1) 2
CODE(ALL) 3
  COND TRANCODE EQ 'STOCK'
  COND USERID   EQ 'JOHN'
CODE(01)
  COND MSGIDSTN EQ 'LTERM* '
  COND MSGODSTN EQ &BANKTRAN
  COND USERID   EQ 'JOHN'
CODE(07) FORM(STAT07)
```

1
Start of global commands

2
Action command

3
Start of qualifying commands, affecting the preceding action command

Example: Running a REXX exec

The following SYSIN data set runs a REXX exec using log records starting from yesterday, specified by the relative date value -1 on the **START** global command, to the end of the input log files; that is, with no explicitly specified stop time:

```
START -1
REXX MYEXEC
```

Global commands

Global commands affect the overall behavior of the report and extract utility.

Specify global commands at the start of the SYSIN data set, before any action commands.

ATF command

Specifies the IMS log record code that OMEGAMON for IMS on z/OS uses to write Application Trace Facility (ATF) summary records to IMS log data sets.

You only need to use this command if you want to process ATF summary records that have been written to an IMS log data set. This command is not required to process ATF summary records that have been written to an ATF log stream.

Specify **ATF** only once in a SYSIN data set, before all action commands.

Format

Parameters

ims_log_code
2-digit hexadecimal number that identifies ATF summary records in IMS log data sets.

Example

The following JCL extracts the first five ATF summary records from an IMS log to a CSV file:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//LOGIN    DD DISP=SHR,DSN=IMSP.SLDS1
//CSV      DD SYSOUT=*
//SYSIN    DD *
ATF(02)
IMSVRM(154)
CSV CODE(ATF:04) LABELS FIELDCASE(ASIS) OUTLIM(5)
FIELDS(
  AESTRAN:Transaction
  AESXSRSP:"Response time"
)
/*
```

To process ATF summary records that have been written to an IMS log, in addition to specifying an **ATF** command, you must also specify an **IMSVRM** command to identify the IMS release.

The following JCL is similar, except that it extracts the ATF summary records from an ATF log stream:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//CSV      DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM ATF:MVS1IMS.IMSP.SM
START 0 /* Today
CSV CODE(ATF:04) LABELS FIELDCASE(ASIS) OUTLIM(5)
FIELDS(
  AESTRAN:Transaction
  AESXSRSP:"Response time"
)
/*
```

To process ATF summary records that have been written to an ATF log stream, you do not need to specify either an **ATF** command or an **IMSVRM** command.

Related reference

IMSVRM command
Specifies the IMS version of the IMS subsystem that created the input IMS log records.

ATF: OMEGAMON for IMS ATF codes
The ATF log type consists of log codes for Tivoli OMEGAMON XE for IMS (OMEGAMON for IMS) Application Trace Facility (ATF) journal records.

CONNECT command

Identifies the log code prefix used by IMS Connect Extensions journal records, enabling Transaction Analysis Workbench to recognize and correctly format these records. If you do not identify this prefix, Transaction Analysis Workbench presents IMS Connect Extensions journal records in dump format. These records contain information about IMS Connect events, such as TCP/IP-based clients using IMS databases.

You can specify **CONNECT** anywhere in a SYSIN data set, but only once.

If you use the ISPF dialog to generate JCL for a formatted record report, the dialog generates a **CONNECT** command using the **IMS Connect Extensions: Log Code** field under dialog option 0.5 **IMS Tools Settings**.

Format

Parameters

log_code_prefix
2-digit hexadecimal number. Identifies the prefix that IMS Connect Extensions used to write journal records. The default value is A0.

Examples

The following JCL creates a formatted record report of IMS Connect Extensions journal records that are related to the transaction code MYTRAN1, excluding OTMA trace (log code 00A3) and IRM trace (00A4) records.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CEXIN    DD DISP=SHR,DSN=HLQ.CEX.JOURNAL
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CONNECT(A0)
REPORT
  CODE(CON:ALL)
    COND TRANCODE EQ 'MYTRAN1'
  CODE(CON:00A3) EXCLUDE
  CODE(CON:00A4) EXCLUDE
  TRACK
/*
```

The following JCL creates a report similar to the previous example, except that this report uses a log token to track related records, and this report includes related records from an IMS log (specified by the LOGIN ddname).

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH,PARM='V154'
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CEXIN    DD DISP=SHR,DSN=HLQ.CEX.JOURNAL
//LOGIN    DD DISP=SHR,DSN=HLQ.IMS.SLDS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CONNECT(A0)
REPORT
  CODE(CON:ALL)
    COND LOGTOKEN EQ C74415061C931542
  CODE(CON:00A3) EXCLUDE
  CODE(CON:00A4) EXCLUDE
  TRACK
/*
```

IMSINDEX command

Consolidates IMS log records into IMS transaction index records (log type IMS, log code CA01).

Specify **IMSINDEX** only once in a SYSIN data set, before all action commands.

IMSINDEX creates each IMS transaction index record when the corresponding transaction completes in the IMS log. After creating an index record, **IMSINDEX** passes the index record to subsequent report and extract utility commands, as if the index record had been read directly from an input file, rather than being created on-the-fly. In reports or extracts that process both the original IMS log records and IMS transaction index records, the original log records and index records will be interspersed. At the end of the

IMS log, the **IMSINDEX** command creates index records for any transactions are still processing or have incomplete details.

Format

Example

This example uses the input IMS log file IMSA.SLDS.LOG.INPUT to create the following output:

- Formatted record report of all IMS transaction index records
- IMS transaction index MY.IMS.INDEX containing only those transactions that either had a response time greater than 2 seconds or abended
- IMS log extract containing the original IMS log records

  Note: Without commands such as **START**, **STOP**, or **CODE** to reduce the data, this IMS log extract is effectively a copy of the original IMS log file; data retention to a new location, rather than data reduction.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//LIMSA001 DD  DISP=SHR,DSN=IMSA.SLDS.LOG.INPUT
//SYSPRINT DD  SYSOUT=*
//IMSEXTRA DD  DISP=SHR,DSN=MY.IMS.EXTRACT
//IMSINDEX DD  DISP=SHR,DSN=MY.IMS.INDEX
//SYSIN    DD  *
IMSVRM(154)
* Create IMS transaction index IMS:CA01 records from the input IMS log file
IMSINDEX

* Report formatted IMS transaction index records
REPORT
  CODE(IMS:CA01)
  FORMAT=SHORT

* Extract IMS transaction index records that either
* had a response time greater than 2 seconds or abended
EXTRACT OUTPUT(IMSINDEX)
  CODE(IMS:CA01)
    COND TOTALTM GT 2.0
  CODE(IMS:CA01)
    COND ABEND NE ' '

* Extract IMS log records, excluding IMS transaction index records
EXTRACT OUTPUT(IMSEXTRA)
  CODE(IMS:CA01) EXCLUDE
/*
```

Figure 130. JCL to create an IMS transaction index

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

IMSVRM command

Specifies the IMS version of the IMS subsystem that created the input IMS log records.

Specify **IMSVRM** only once in a SYSIN data set, before all action commands.

IMSVRM is an alternative to specifying the IMS version in the **PARM** parameter of the **EXEC** statement.

Format

Examples

To specify IMS V15.4:

```
IMSVRM(154)
```

When you are developing session workflow task JCL, to use the IMS version established by processing the FILE substitution variable:

```
IMSVRM(%IMSVRM)
```

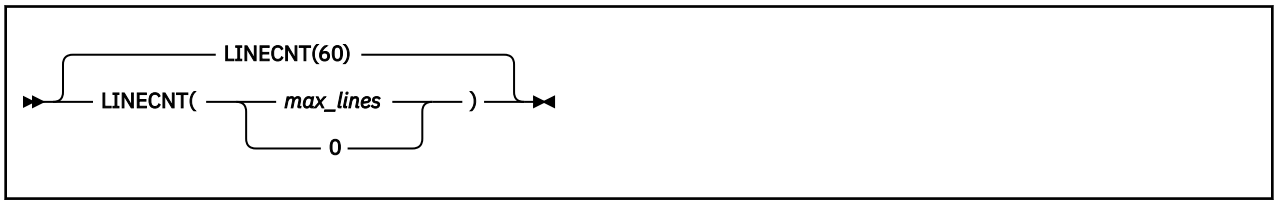
LINECNT command

Specifies the maximum number of lines, including headings, to print on each page of a report.

LINECNT applies only to reports created by the **REPORT** command.

A SYSIN data set can contain zero or more **LINECNT** commands. Each **REPORT** command uses the value of the most recently specified **LINECNT** command.

Format



Parameters

LINECNT(*max_lines*)
A positive integer specifying the maximum number of lines, including headings, to print on each page of a report. Print the report heading at the start of each page. The default is 60 lines.

LINECNT(0)
Do not paginate reports. Print the report heading only once, at the start of a report.

LOGSTREAM command

Specifies that the input log file is a log stream rather than a data set.

Specify **LOGSTREAM** only once in a SYSIN data set, before all action commands.

If you specify **LOGSTREAM**, the following conditions apply:

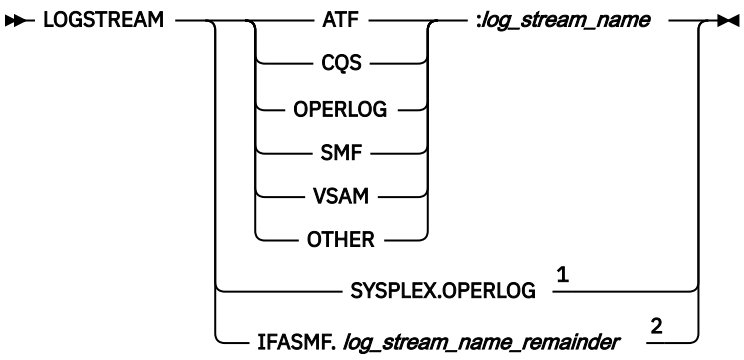
- The report and extract utility ignores any DD statements for input data sets.
- You must specify a **START** command.

  Log streams can potentially cover a long period of time. **START** is required to prevent you from accidentally reporting more data than you intended.

Tip: If you want to use records from a log stream together with records from other data sources as input to the report and extract utility, first use the **LOGSTREAM** command with an **EXTRACT** command to create an extract (data set) of the log stream records. In a subsequent job step, use the **EXTRACT** command to

create another extract that combines the records extracted from the log stream with records from other data sources.

Format



Notes:

[1] If the log stream name is SYSPLEX.OPERLOG, you can omit the OPERLOG: prefix.
[2] If the log stream name begins with IFASMF., you can omit the **SMF:** prefix.

Parameters

log_stream_name

The name of a log stream. Prefix the log stream name with one of the types of log stream whose records Transaction Analysis Workbench can format or, for other types of log stream, OTHER. Use a colon (:) to separate the prefix from the name. For example:

```
LOGSTREAM OPERLOG:SYSPLEX.OPERLOG
```

To get a list of the log streams that are accessible to you (that are on the same system, or sysplex, as the one you are using), enter the following MVS system command:

```
D LOGGER,C
```

Example: SMF report using a log stream

The following JCL creates a DB2 thread accounting report, for 9 a.m. to 4 p.m. today (the day that you submit this JCL), using SMF type 101 records from the SMF log stream IFASMF.PRODPLEX:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0-09.00
STOP  0-16.00
REPORT SMF(101)
/*
```

Figure 131. JCL to create a report using an SMF log stream

Example: OPERLOG report

The following JCL creates an OPERLOG report, for yesterday (the day before you submit this JCL), of RACF messages that describe when user ID "GXH" last accessed MVS system ID "MVS1".

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
LOGSTREAM OPERLOG:SYSPLEX.OPERLOG
START -1
STOP  0
REPORT OPERLOG
  CODE(MVS:CA52)
    COND SYSID EQ 'MVS1'
    COND TEXT(2) EQ 'ICH70001I GXH'
/*
```

Figure 132. JCL to create an OPERLOG report

Example: formatted record report using a CICS primary system log stream

The following JCL creates a formatted record report, for today (the day you submit this JCL), of the records in a CICS primary system log stream (DFHLOG).

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM OTHER:CICSPROD.DFHLOG
START 0
REPORT
/*
```

Figure 133. JCL to create a formatted record report for a CICS primary system log stream

The formatted record report consists of a dump listing of each record in the log stream.

Example: Formatted record report of CICS performance records using an SMF log stream

The following example creates a formatted record report of CICS performance records (SMF type 110, subtype 1, class 3) for transaction MYTR, from 11:30 p.m. yesterday to 1:30 am today (the day that you submit this JCL), from the SMF log stream IFASMF.PRODPLEX.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START -1-23.30
STOP   0-01.30
REPORT
  CODE(CMF:6E13)
    COND TRAN EQ 'MYTR'
/*
```

Figure 134. JCL to create a formatted record report of CICS performance records from an SMF log stream

For comparison, the following JCL creates the same report from an SMF data set instead of a log stream. Note the use of the SMFIN DD statement instead of the **LOGSTREAM** command.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,
//            DSN=SYSA.PROD.SMF(0)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
START -1-23.30
STOP   0-01.30
REPORT
  CODE(CMF:6E13)
    COND TRAN EQ 'MYTR'
/*
```

Figure 135. JCL to create a formatted record report of CICS performance records from an SMF data set

Related reference

Log stream types
Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

REPORT command for OPERLOG reports
Prints selected records from OPERLOG, the z/OS operations log (log stream).

NOHEADING command

Omits page headings from formatted record reports. Has no effect on "brief" formatted record reports or any other reports.

NOHEADING applies to all **REPORT** commands for formatted record reports in a SYSIN data set, except those that specify FORMAT(BRIEF).

Format

OUTZONE command

Specifies the time zone of time stamps in output CSV and JSON data.

For CSV and JSON output only, the **OUTZONE** command overrides the output time zone that is specified by the **ZONE** command. For other outputs, such as reports, **OUTZONE** is ignored.

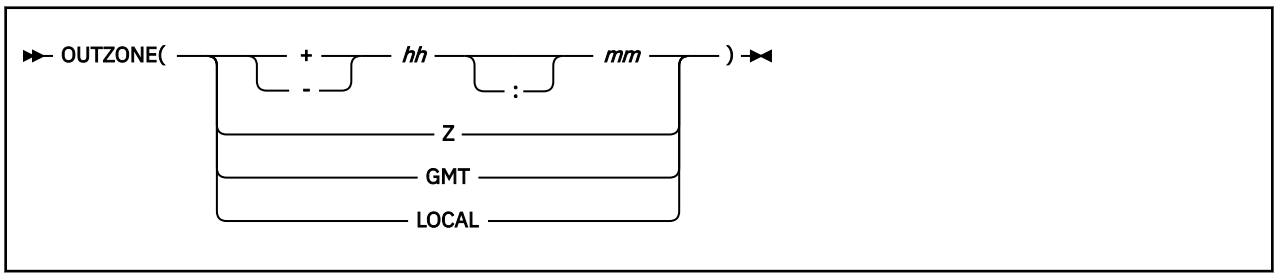
A typical use of **OUTZONE** is OUTZONE(Z), to output time stamps in CSV and JSON output as UTC.

Tip: OUTPUT sets the time zone, but not the time stamp format. The time stamp format determines whether time stamps in CSV and JSON output have a zone designator. To include a trailing zone designator in time stamps, use the **TIMEFORMAT** parameter of the **CSV** or **JSON** command. For details, see Chapter 48, “Time stamps in CSV and JSON,” on page 239.

If you omit **OUTZONE**, then time stamps in CSV and JSON output match the time zone specified by **ZONE**. If you omit **ZONE**, then time stamps in CSV and JSON output default to the local time zone of the system on which you are running the report and extract utility.

Specify **OUTZONE** only once in a SYSIN data set, before all action commands.

Format



Parameters

+hh*:*mm* | *-hh*:*mm
UTC offset: *hh* hours and *mm* minutes ahead of (+) or behind (-) UTC. For example: `+10:00` for Sydney, Australia time; `-08:00` for US Pacific time.

Z | GMT | +00:00 | -00:00
All of these values specify the same time zone: Coordinated Universal Time (UTC).

LOCAL
The time zone of the system on which you are running the report and extract utility.

Example

Suppose you have a dumped SMF data set containing SMF type 30 records that were created on a system whose local time is US Pacific time (UTC-08:00). You want to forward those records in JSON Lines format to an analytics platform, and you want to represent all time stamps in the JSON as UTC in ISO 8601 format, with the zone designator Z.

The following example JCL instructs Transaction Analysis Workbench to interpret local time stamps in the input records as UTC-08:00, and then output them in the JSON as UTC in ISO 8601 format:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
ZONE(-08:00)
OUTZONE(Z)
STREAM NAME(ELASTIC) HOST(myserver) PORT(6789) +
  TIMEFORMAT(ISO8601)
JSON CODE(SMF:30.) STREAM(ELASTIC)
/*
```

For instance, to create the "time" field value in the output JSON, Transaction Analysis Workbench uses the local time stamp, consisting of the fields SMF30DTE (date) and SMF30TME (time since midnight), from the standard SMF record header. Transaction Analysis Workbench interprets the local time stamp according to the time zone specified by **ZONE**, and then converts the time stamp to the time zone specified by **OUTZONE**.

Related concepts

Time stamps in CSV and JSON
CSV and JSON output by Transaction Analysis Workbench includes the time stamp of the corresponding input log record, also known as the *event* time stamp. The event time stamp is always included in the output, regardless of any **CSV** or **JSON** command parameters, such as **FIELDS**, that specify which other fields to include. Depending on the input record type and the selected fields, other fields in the output might also be time stamps. All time stamps in the output, including the event time stamp, have the same format.

Related reference

ZONE command

Specifies the time zone that Transaction Analysis Workbench uses for the default time zone, the output time zone, and the time zone for the **START** and **STOP** commands.

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

PAGELIM command

Specifies the maximum number of pages to be reported. Ignored for extracts and CSV files.

PAGELIM applies only to reports created by the **REPORT** command.

You can specify **PAGELIM** only once in a SYSIN data set, before all action commands.

Format

Parameters

max_pages
A positive integer, 1-9,999,999, specifying the maximum number of pages to be reported. The default is 10,000 pages. It is recommended that you include a page limit in your report job because a formatted log file can produce a large volume of output.

SCHEMAONLY command

Allows the **CSV** and **JSON** commands to produce some outputs, such as table schemas, without any input logs. These "schema-only" outputs rely on knowledge about log record types that is built into Transaction Analysis Workbench.

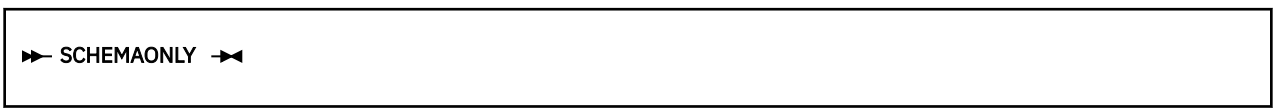
Specify **SCHEMAONLY** only once in a SYSIN data set, before all action commands.

Specifying **SCHEMAONLY** has the following effects:

- Input logs are ignored and can be omitted from the JCL. DD statements for input logs are ignored. The **LOGSTREAM** command is ignored.
- Only the following parameters of the **CSV** and **JSON** commands produce output:

  DB2LOAD
  HCATALOG
  LSCONFIG
  METADATA
  SCHEMA
- Other action commands in the SYSIN data set are ignored.

Format



Example

The following example JCL writes a DB2 DDL **CREATE TABLE** statement for CICS monitoring facility performance class records:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//DDL      DD  SYSOUT=*
//SYSIN    DD  *
SCHEMAONLY
CSV CODE(CMF) SCHEMA(DDL) TABLE(MYSCHEMA.CICS) FIELDCASE(UPPER)
FIELDS(
  SMFSID:LPAR
  APPLID
  Tran
  Program
  Userid
  ⋮
  )
/*
```

Notice that this JCL does not specify any input logs.

The generated **CREATE TABLE** statement contains the table name specified by the **TABLE** parameter, `MYSCHEMA.CICS`.

The field names in the **CREATE TABLE** statement match the **FIELDS** command in the JCL, but with the following differences:

- The column order in the **CREATE TABLE** statement matches the field order in the Transaction Analysis Workbench knowledge module, not the **FIELDS** command in the JCL.
- The column names in the **CREATE TABLE** statement are all uppercase, as specified by the **FIELDCASE(UPPER)** parameter in the JCL.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

START, STOP (FROM, TO) commands

Specify the report interval. Only log records with an event time stamp within the report interval are included in the output.

The **START** and **STOP** commands of the batch report and extract utility, FUWBATCH, have similar syntax to the **FROM** and **TO** commands of the automated file selection batch utility, FUWFILES.

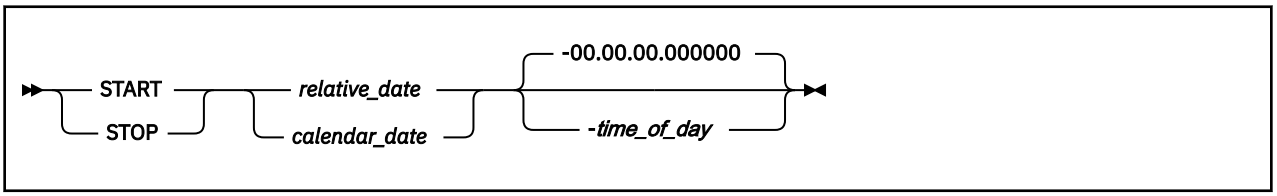
The following conditions apply to the **START** and **STOP** commands of the batch report and extract utility:

- If the input logs are data sets, the report interval is optional: you can omit either or both **START** and **STOP**.
- If the input log is a log stream, specified by a **LOGSTREAM** command in the SYSIN data set, **START** is required and **STOP** is optional. Log streams can cover a long period of time. **START** is required to prevent you from accidentally reporting more data than you intended.
- If specified, **START** and **STOP** can occur only once each, and must precede all **REPORT**, **EXTRACT**, **CSV**, **JSON**, **MWP**, or **REXX** commands. All output requested in the same SYSIN data set has the same report interval.

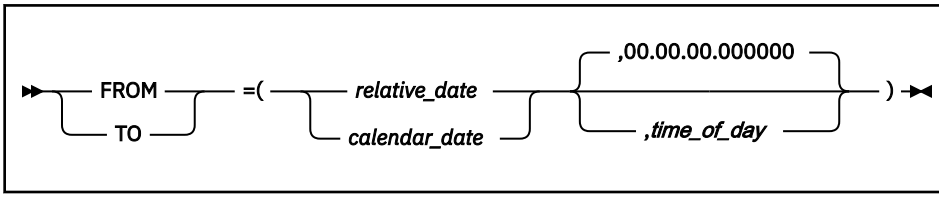
For automated file selection, the report interval is required: you must specify the **FROM** and **TO** commands. Similarly, the report interval applies to all file selections specified in the SYSIN data set.

Format

For the report and extract utility:



For the automated file selection utility:



Parameters

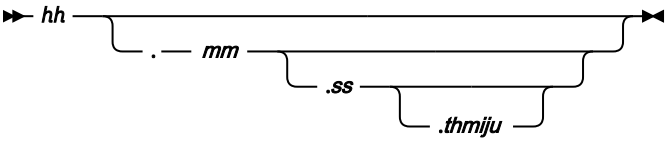
relative_date
Zero or a negative integer representing a date relative to today. 0 represents today, -1 represents yesterday, and so on, to -999 (999 days ago).

calendar_date
A date in the following format:

yyyy-mm-dd

For example, the value 2015-11-30 represents the date 30 November 2015.

time_of_day
A time of day on the specified date, in the following 24-hour clock format:



The default is midnight, 00.00.00.000000.

If you omit trailing levels of precision, zero values are assumed. For example, all of the following values represent 6 a.m.:

06
06.00
06.00.00
06.00.00.000000

In **START** and **STOP** commands, separate the date and time with a hyphen (-). In **FROM** and **TO** commands, separate the date and time with a comma (,).

For example, the following commands refer to midday on 30 November 2015:

- For the report and extract utility:

```
START 2015-11-30-12.00
```

- For the automated file selection utility:

```
FROM=(2015-11-30,12.00)
```

The following commands refer to 5:30 p.m yesterday:

- For the report and extract utility:

```
START -1-17.30
```

- For the automated file selection utility:

```
FROM=(-1,17.30)
```

Examples

The following example creates an OPERLOG report, for yesterday, of RACF messages (prefix ICH) for MVS system ID MVS1:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH     EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
LOGSTREAM OPERLOG:SYSPLEX.OPERLOG
START -1
STOP 0
REPORT OPERLOG
  CODE(MVS,CA52)
    COND SYSID EQ 'MVS1'
    COND TEXT(2) EQ 'ICH'
```

To create an equivalent report for today, specify:

```
START 0
```

with no **STOP** command.

To create an equivalent report for today, from 6:30 a.m. to 1 p.m., specify:

```
START 0-06.30
STOP 0-13.00
```

Related concepts

Event time stamps versus log record time stamps

Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Related reference

UNSORTED command

Indicates that records in the input log files are unsorted; their records are not in time sequence. Specifying **UNSORTED** causes Transaction Analysis Workbench to test each input log record and process it if falls within the period specified by the **START** and **STOP** commands.

STREAM command

Forwards log data in JSON Lines or CSV format to a TCP socket.

The **STREAM** command defines a stream that sends log data in JSON Lines or CSV format to a TCP socket. Typically, the TCP socket is on a remote host that is running an analytics platform such as Elastic or Splunk. The analytics platform ingests the data that you send. The stream definition includes connection details such as the destination host and port number and, optionally, security (SSL/TLS) details.

The **STREAM** command performs no action unless it is referred to by a **JSON** or **CSV** command. To stream JSON Lines or CSV data, you use a **JSON** or **CSV** command with a **STREAM** parameter that refers to the name of a **STREAM** command.

A SYSIN data set can contain multiple **STREAM** commands. Each **STREAM** command has a **NAME** parameter that uniquely identifies the stream definition in the SYSIN data set.

The report and extract utility opens one network connection for each **STREAM** command, not for each **JSON** or **CSV** command.

Multiple **JSON** or **CSV** commands can send data in the same stream.

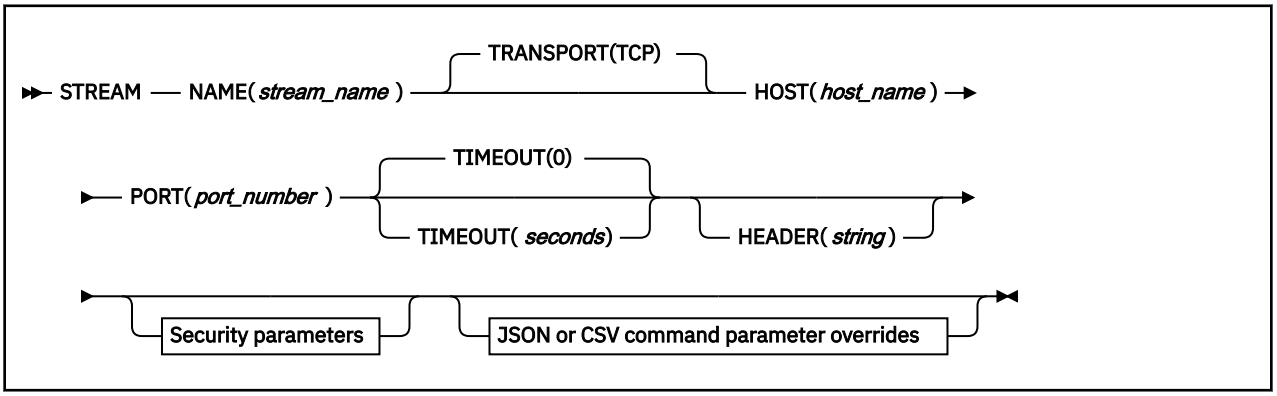
Sending output from multiple CSV commands to the same stream

If you send output from multiple **CSV** commands to the same stream:

- Do not specify the **LABELS** parameter on any of the **CSV** commands. Otherwise, the output might contain header rows interleaved with data rows.

  Tip: If you need to include a header row of field labels as the first line of the stream, use the **HEADER** parameter of the **STREAM** command to explicitly specify the contents of the header row.
- Ensure that all **CSV** commands output the same fields in the same order. The report and extract utility does not check this for you.

Format



Parameters

NAME
The stream name that uniquely identifies this stream definition in the SYSIN data set. A stream name consists of 1 - 8 alphanumeric characters: A - Z, a - z, 0 - 9. Stream names are case-insensitive: the values `Stash1` and STASH1 are considered to be identical.

TRANSPORT
The transport protocol for the stream. The only supported value is TCP.

HOST
The destination hostname or IP address, up to 255 characters.

PORT
The destination port number.

TIMEOUT
How long to wait before timing out. Either:

- An integer number of seconds
- 0 (the default) to wait forever (no timeout)

HEADER
A string to send as the first line of the stream.

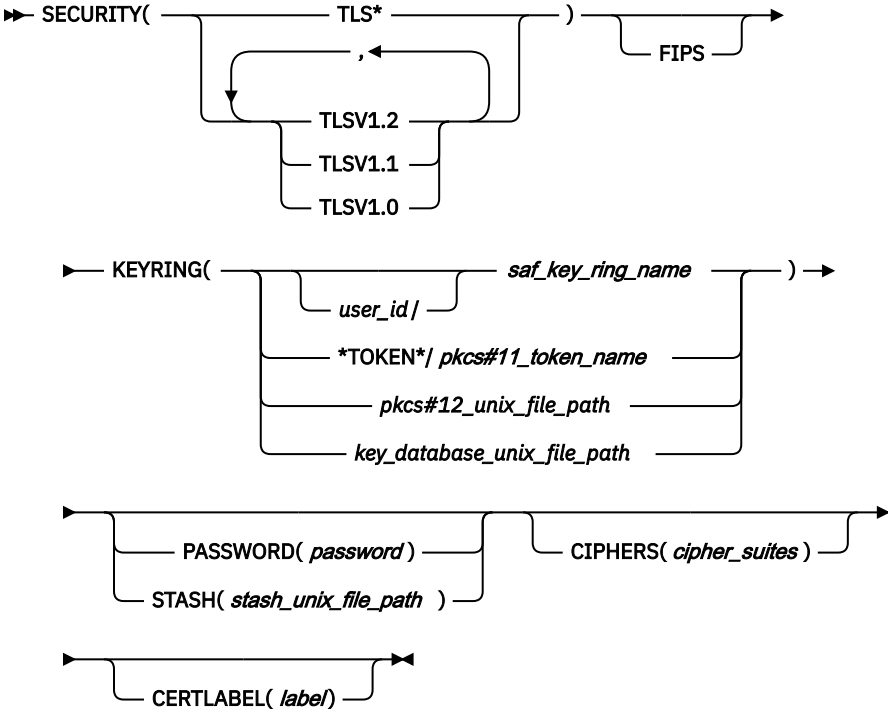
HEADER is useful for overriding Splunk default field values. For example, to override the default index for a Splunk TCP data input:

```
HEADER(***SPLUNK*** index=my_alt)
```

This header line is terminated by the same end-of-record marker as the first event. For example, if the **JSON** command that writes the first event specifies the **LF** parameter, then the header line is also terminated by **LF**.

Security parameters

Security parameters are required only for secure (SSL/TLS) connections.



SECURITY
Specifies one or more security protocols to try, in order. The special value TLS* represents the list of all supported versions of TLS, starting with the latest version:

```
TLSV1.2,TLSV1.1,TLSV1.0
```

If you omit the **SECURITY** parameter, the **STREAM** command attempts to open a connection without SSL/TLS.

FIPS
Sets z/OS System SSL Federal Information Processing Standards (FIPS) mode on. For information about FIPS mode, see the z/OS System SSL documentation.

KEYRING
Specifies a collection of certificates that includes the certificates required for this connection. Can be one of the following:

SAF key ring
Specified in the format *owner_user_id/key_ring_name* or, if the current user owns the key ring, just the key ring name. For example:

```
my/fuw_keyring
```

If the current user owns the key ring, the current user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class. If another user owns the key ring, the current user must have UPDATE access to that resource.

Certificate private keys are not available when using a SAF key ring owned by another user, except for SITE certificates where CONTROL authority is given to IRR.DIGTCERT.GENCERT

in the FACILITY class or for user certificates where READ or UPDATE authority is given to *ring_owner.ring_name*.LST resource in the RDATALIB class.

Key database
A key database created by the z/OS gskkyman utility. The key database is specified as a z/OS UNIX file path. For example:

```
/u/my/sslcerts/fuw.kdb
```

PKCS #12 file
Specified as a z/OS UNIX file path. For example:

```
/u/my/sslcerts/fuw.p12
```

PKCS #11 token
Specified in the format *TOKEN*/*token_name*. For example:

```
*TOKEN*/fuw.pkcs11.token
```

The *TOKEN* qualifier indicates that the value refers to a PKCS #11 token rather than a SAF key ring.

If you specify a key database or PKCS #12 file, but you do not specify either a **STASH** parameter or a **PASSWORD** parameter, then the **STREAM** command looks for a stash file in the same directory as the key database or PKCS #12 file, and with the same base file name, but with `.sth` extension. For example, if the **KEYRING** parameter specifies the following z/OS UNIX file path:

```
/u/my/sslcerts/fuw.kdb
```

or:

```
/u/my/sslcerts/fuw
```

(with no extension)

then the **STREAM** command looks for a stash file at the following path:

```
/u/my/sslcerts/fuw.sth
```

STASH
Specifies the z/OS UNIX path of the stash file that contains the password for the key database or PKCS #12 file.

If **KEYRING** specifies a SAF key ring or PKCS #11 token, **STASH** is ignored.

The stash file name must have a `.sth` extension. If the specified file name has a different extension, that extension is ignored and replaced with the `.sth` extension.

If the **PASSWORD** parameter is specified, **STASH** is ignored.

PASSWORD
Specifies the password for the key database or PKCS #12 file.

If **KEYRING** specifies a SAF key ring or PKCS #11 token, **PASSWORD** is ignored.

CIPHERS
Specifies a list of candidate cipher suites to try, in order. The list is a concatenation of 4-digit hexadecimal cipher suite numbers supported by z/OS System SSL. For example:

```
CIPHERS(000A000D001000130016)
```

If you omit **CIPHERS**, the **STREAM** command uses the system default list of cipher suites. That list changes depending on whether or not FIPS mode is on.

Tip: To match a z/OS System SSL cipher suite number to the corresponding OpenSSL cipher suite name, go to the z/OS System SSL documentation and look up the "short name" for that cipher suite in the table of cipher suite definitions. The short name is the name that is defined in the associated RFC.

Then go to the OpenSSL documentation for the **ciphers** command, and use the RFC name to find the corresponding OpenSSL name.

For more information on cipher suite definitions, see the z/OS System SSL documentation.

CERTLABEL
Specifies the label of the client certificate that is used to authenticate Transaction Analysis Workbench (the client) to the destination host (server). The client certificate, and its private key, must be in the collection that is specified by the **KEYRING** parameter.

CERTLABEL is only used if the destination host requires client authentication.

If the destination host requires client authentication, but you omit **CERTLABEL**, then the **STREAM** command uses the default certificate from the collection that is specified by the **KEYRING** parameter.

Example: Streaming JSON Lines over unsecure TCP

The following JCL selects three record types from the records written today to an SMF log stream, and then forwards them all in a single stream of JSON Lines over unsecure TCP to a remote analytics platform, such as Elastic or Splunk:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0 /* Today
OUTZONE(Z)
STREAM NAME(S1) HOST(analytics) PORT(6789) +
       OUTLIM(1000) + /* Limit data volume for testing
       TIMEFORMAT(ISO8601)
JSON STREAM(S1) CODE(CMF)
JSON STREAM(S1) CODE(SMF:30.)
JSON STREAM(S1) CODE(DTR:003)
/*
```

Typically, each **JSON** command is followed by a **FIELDS** qualifying command that specifies which fields from the input log records to include in the output. For brevity, this example omits the **FIELDS** command; all fields are included in the output.

Example: Secure TCP

For a secure TCP port, add the following parameters to the **STREAM** command in the previous example:

```
SECURITY(TLS*) FIPS KEYRING(my/fuw.analytics)
```

Related concepts

Log forwarding
You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

Basic Splunk configuration for streaming JSON Lines over TCP
To stream JSON Lines to Splunk over TCP, you need to configure a Splunk TCP data input that breaks each line of the stream into a separate event, recognizes event time stamps, and specifies the event data format as JSON.

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Parameters of the JSON or CSV command that you can override on the STREAM command

Each **JSON** or **CSV** command can specify parameters that affect the formatting of its output. To ensure consistent formatting in a stream of data from multiple **JSON** or **CSV** commands, you can specify some **JSON** and **CSV** command parameters on the **STREAM** command.

When specified on the **STREAM** command, the following parameters set the corresponding parameters of the **JSON** or **CSV** commands:

- Parameters that can be specified on either a **JSON** or a **CSV** command:

  ASCII | **EBCDIC**
  EOL
  FIELDCASE
  OUTLIM
  TIMEFORMAT

- **JSON**-only parameters:

  FLAT
  LINES | **ARRAY**
  MISSING

- **CSV**-only parameters:

  DELIMITER
  LABELS | **NOLABELS**
  TYPECOLUMN

Parameters specified on the **STREAM** command override parameters specified on the **JSON** or **CSV** command.

For more details on these parameters, see the documentation of the **JSON** and **CSV** commands.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Adding a CA certificate to a RACF key ring

To connect to a secure TCP port on a remote system, you need the CA certificate used by that system. If you store certificates in RACF key rings, you need to add that CA certificate to a key ring.

This procedure condenses information that is also available in the RACF product documentation. For more details on each step in this procedure, see the z/OS Security Server RACF product documentation for security administrators.

1. Copy the certificate-authority (CA) certificate `.pem` text file from the remote system to z/OS.

   For example, transfer the text file `myCACertificate.pem` from the remote system to the z/OS MVS sequential data set 'MY.CACERT.PEM'.

   The first line of the file must be:

   ```
   -----BEGIN CERTIFICATE-----
   ```

and the last line must be:

```
-----END CERTIFICATE-----
```

2. If the key ring does not exist, use the RACF **RACDCERT ADDRING** command to create it.

   Example:

   ```
   RACDCERT ID(my) ADDRING(fuw.splunk)
   ```

   where `my` is the user ID of the key ring owner and `fuw.splunk` is the key ring name.
3. Use the RACF **RACDCERT ADD** command to define the certificate to RACF as a trusted certificate.

   Example:

   ```
   RACDCERT ID(my) ADD('MY.CACERT.PEM') TRUST
   WITHLABEL('FUW Splunk CA certificate 1')
   ```

4. Use the RACF **RACDCERT CONNECT** command to connect the certificate to the key ring.

   Example:

   ```
   RACDCERT ID(my) CONNECT(LABEL('FUW Splunk CA certificate 1') RING(fuw.splunk)
   USAGE(CERTAUTH))
   ```

5. Use the RACF **SETROPTS** command to refresh the classes for certificates and key rings.

   Example:

   ```
   SETROPTS RACLIST(DIGTCERT, DIGTRING) REFRESH
   ```

6. Use the RACF **RACDCERT LISTRING** command to confirm that the certificate has been added to the key ring.

   Example:

   ```
   RACDCERT ID(my) LISTRING(*)
   ```

Refer to the key ring in the **KEYRING** parameter of the **STREAM** command.

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

TRACE command

Specifies how to process IMS log records of type 67FA, and 67FF records for IMS trace tables: in their original format, or as individual IMS trace table entries (log type ITR).

You can specify **TRACE** anywhere in a SYSIN data set, but only once.

Format

Parameters

ON
Process IMS trace records as individual trace entries (log type ITR).

OFF
Process IMS trace records in their original (67FA or 67FF) format.

For **REPORT** and **CSV** commands, the default is ON.

For **EXTRACT** commands, the default is OFF.

Example: Extracting the original trace records

The following JCL extracts the IMS trace log records from LOGIN and writes them to EXTRACT in their original format.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH     EXEC PGM=FUWBATCH,PARM=V121
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
//LOGIN    DD  DISP=SHR,DSN=IMS.LOG
//EXTRACT  DD  DISP=OLD,DSN=MY.IMS.TRACE.EXTRACT
//SYSIN    DD  *
EXTRACT
CODE(IMS,67FA)
CODE(IMS,67FF)
TRACE OFF
/*
```

Figure 136. JCL to extract original IMS (67FA and 67FF) trace records

Example: Reporting individual IMS trace entries

The following JCL prints the individual IMS trace entries in the IMS log:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH     EXEC PGM=FUWBATCH,PARM=V121
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=IMS.LOG
//LOGRPT   DD  SYSOUT=*
//SYSIN    DD  *
REPORT
CODE(ITR,ALL)
TRACE(ON)
FORMAT(BRIEF)
/*
```

Figure 137. JCL to report individual IMS trace entries

Related reference

Traces
IMS log records related to traces.

UNSORTED command

Indicates that records in the input log files are unsorted; their records are not in time sequence. Specifying **UNSORTED** causes Transaction Analysis Workbench to test each input log record and process it if falls within the period specified by the **START** and **STOP** commands.

Use **UNSORTED** only when both of the following conditions are true:

- You are processing input files whose records are not in time sequence.
- You have specified a **START** command, a **STOP** command, or **START** and **STOP**

If you have not specified either **START** or **STOP**, then **UNSORTED** is ignored.

Only use **UNSORTED** when you need to. Otherwise, processing might take longer than necessary. For example, if input records are in time sequence and you have specified **STOP**, then specifying **UNSORTED** will cause Transaction Analysis Workbench to unnecessarily continue reading records past the stop time until the end of the file.

You can specify **UNSORTED** anywhere in a SYSIN data set, but only once.

Tip: To determine whether records in an SMF data set are in time sequence, request an SMF Recap report.

Format

Example (part 1 of 2): Extracting records from an unsorted SMF file

Typically, records in a log file are in time sequence. However, some sites might post-process log files so that their records are no longer in time sequence. For example, a site might combine SMF files from individual systems, for the same time period, into a single large file, by concatenating the files rather than merging their records in time sequence. In this case, although the set of records for each system are in time sequence, the resulting file as a whole is not in time sequence. To extract all records for a particular time period from such a combined SMF file, you must specify the **UNSORTED** command.

The following JCL extracts records within the specified time period from an unsorted SMF file.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH     EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=SYSA.PROD.SMF(0)
//EXTRACT  DD  DISP=SHR,DSN=MY.SMF.EXTRACT.UNSORTED
//SYSIN    DD  *
START 0-08.00
STOP  0-08.15
UNSORTED /* SMF file is not in time sequence
EXTRACT
REPORT SMF(RECAP) OUTPUT(SMFRECAP) /* SMF Recap report
/*
```

Figure 138. JCL to extract records from an unsorted SMF file

Tip: When creating an extract of an SMF file, request an SMF Recap report. This report provides an overview of the type of records in the extract; this is useful for determining the types of analysis that you can perform using the extract.

Example (part 2 of 2): Sorting an SMF extract

This example is a continuation of the previous example; rather than demonstrating the **UNSORTED** command, this example demonstrates how to sort the SMF extract that was created by the previous example from an unsorted SMF file.

The following JCL sorts an SMF extract into time sequence.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//SORT   EXEC PGM=SORT
//SORTIN   DD DISP=SHR,DSN=MY.SMF.EXTRACT.UNSORTED
//SORTOUT  DD DISP=SHR,DSN=MY.SMF.EXTRACT.SORTED
//SORTWK01 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK02 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK03 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK04 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
```

Figure 139. JCL to sort records in an SMF extract into time sequence

You could also use this JCL to sort the original SMF file, but it is more efficient to extract the records from the time period that you want, and then sort that smaller set of records in the extract.

Tip: When creating an extract of an SMF file, also request an SMF Recap report. This report provides an overview of the type of records in the extract; this is useful for determining the types of analysis that you can perform using the extract.

Related reference

START, STOP (FROM, TO) commands
Specify the report interval. Only log records with an event time stamp within the report interval are included in the output.

ZONE command

Specifies the time zone that Transaction Analysis Workbench uses for the default time zone, the output time zone, and the time zone for the **START** and **STOP** commands.

The time zone specified by **ZONE** has three uses:

Default time zone
To interpret time stamps whose time zone cannot be determined from their original log records. For example, time stamps that are in local time, with no zone designator.

Output time zone
To represent time stamps in reports, or in CSV or JSON output.

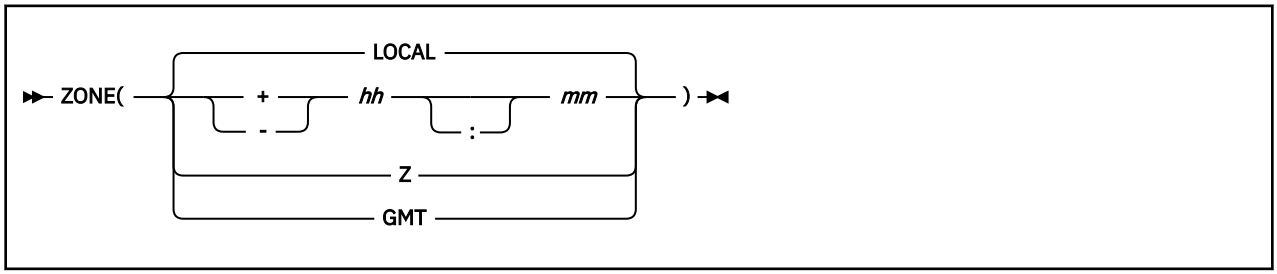
Tip: To specify a different output time zone for CSV or JSON output, use the **OUTZONE** command.

START and STOP commands
To interpret values specified by the **START** and **STOP** commands.

Specify **ZONE** only once in a SYSIN data set, before all action commands.

When generating JCL, the ISPF dialog uses the value of the **Time Zone** field under option 0.4 **Time Control**.

Format



Parameters

LOCAL
The time zone of the system on which you are running the report and extract utility. If you omit **ZONE**, this is the default value.

Z | GMT | +00:00 | -00:00
All of these values specify the same time zone: Coordinated Universal Time (UTC).

+hh*:*mm* | *-hh*:*mm
UTC offset: *hh* hours and *mm* minutes ahead of (+) or behind (-) UTC. For example: `+10:00` for Sydney, Australia time; `-08:00` for US Pacific time.

Example

```
ZONE(+08:00)
```

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

OUTZONE command
Specifies the time zone of time stamps in output CSV and JSON data.

Action commands

Action commands typically process logs, and then produce output such as a report or an extract, or log data converted to CSV or JSON format.

If you specify the **SCHEMAONLY** global command, then you can use the **CSV** and **JSON** action commands to produce some types of output, such as table schemas, without any input logs, based on knowledge that is built into Transaction Analysis Workbench.

Action commands cannot share output ddnames. If an action command specifies a ddname as its output destination, no other action command in the same SYSIN data set can output to that ddname.

However, some action commands can share output destinations that are not ddnames. For example:

- Multiple **JSON** commands can write to the same network stream.
- Multiple **JSON** commands can write to stdout.

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Except for a few parameters that apply only to CSV or only to JSON, the difference between a command that converts log data to CSV and a command that converts the same log data to JSON is the command keyword, **CSV** or **JSON**. If you already have a **CSV** command, but you want to output JSON instead, then, as a starting point, simply replace the **CSV** command keyword with **JSON**.

To create a CSV file specifically for the Mobile Workload Reporting Tool (MWRT), use the **MWP** command instead of the **CSV** command.

To convert logs to a delimiter-separated values (DSV) format that uses a delimiter other than a comma, use the **CSV** command with a **DELIMITER** parameter.

Each **CSV** or **JSON** command processes log records of one specific combination of log type and code. To specify the log type and code, you must use either a **CODE** parameter or a **FORM** parameter.

A SYSIN data set can contain multiple **CSV** and **JSON** commands for the same log type and code or any combination of different log types and codes.

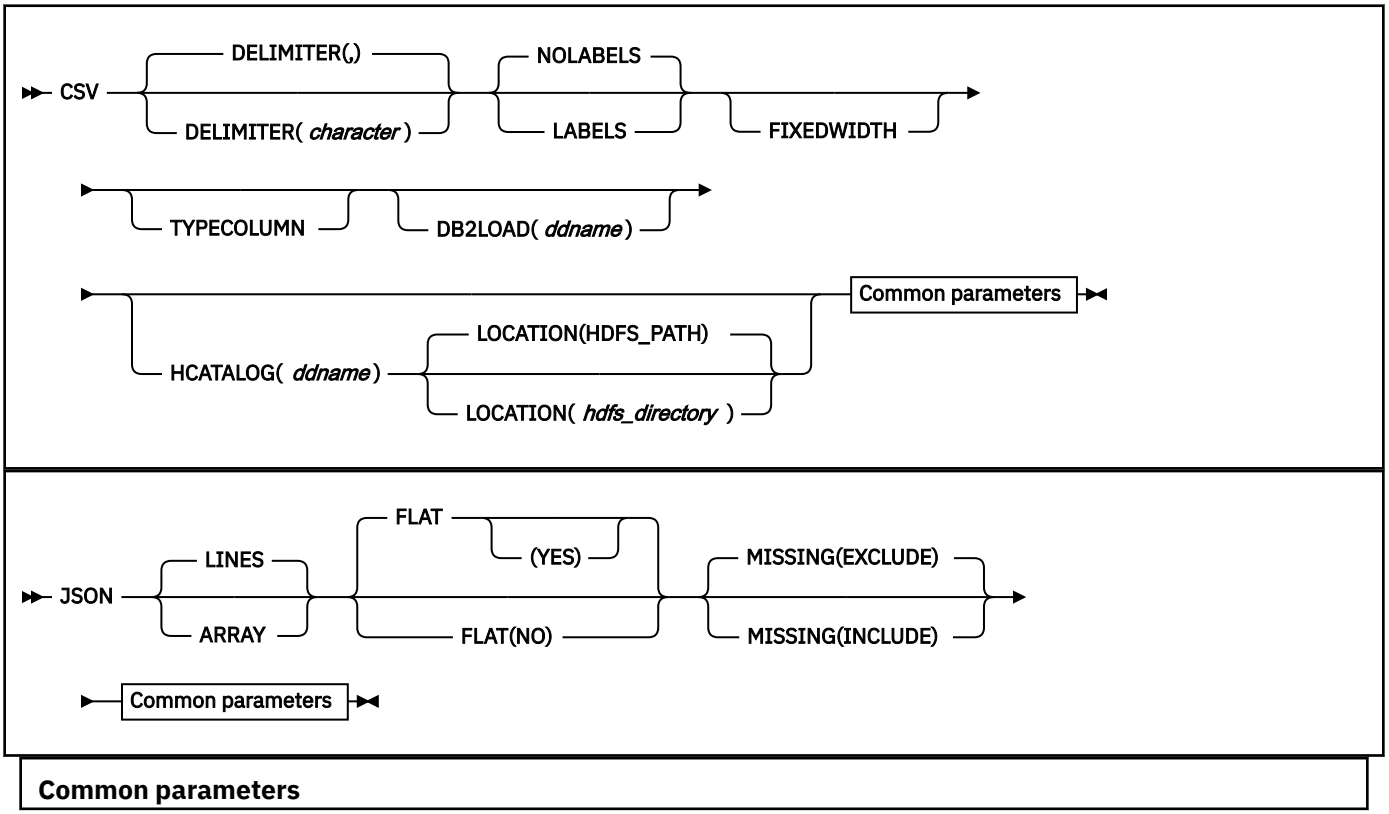
A **CSV** or **JSON** command that writes to a ddname cannot write to the same ddname as another command. However, multiple **JSON** commands can write to the same network stream or to stdout.

To filter the input log records that are processed by a **CSV** or **JSON** command, use one of the following methods:

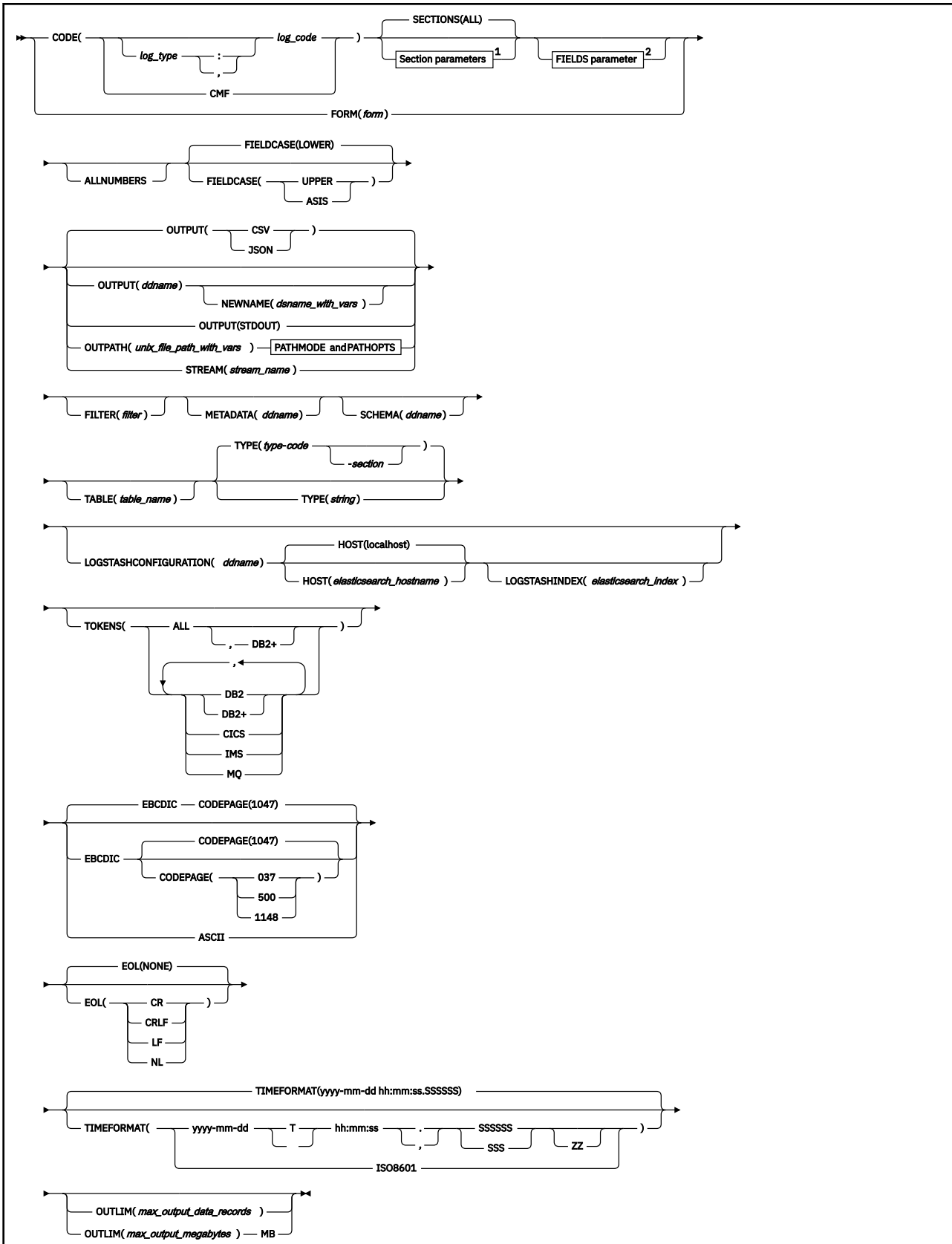
- Immediately after the **CSV** or **JSON** command, specify filter conditions in **CODE** qualifying commands
- In the **CSV** or **JSON** command, specify a **FILTER** parameter that refers to a filter in a control repository

You can use the **CSV** or **JSON** command to produce some outputs without any input logs. For details, see the **SCHEMAONLY** global command.

Format



Common parameters



Notes:

[1] Section parameters specify which sections of a log record to include in the output, and how to handle repeating sections.

[2] The **FIELDS** parameter specifies which fields of a log record to include in the output. The **FIELDS** parameter has the same syntax as the **FIELDS** qualifying command.

CSV-only parameters

The following parameters apply only to the **CSV** command. If you specify these parameters on a **JSON** command, they are ignored.

There is one exception: the **FIXEDWIDH** parameter applies to both the **CSV** command and the **JSON** command. **FIXEDWIDTH** is presented here as a CSV-only parameter because, while you can use it with the **JSON** command, its behavior is much more likely to be useful for CSV output than JSON output.

DELIMITER
The field delimiter, also known as the column delimiter: the character that separates field values. The default delimiter is a comma, with one exception: if the **TIMEFORMAT** parameter specifies a comma as the decimal sign (the character that separates the integer number of seconds from the decimal fraction), the default delimiter is a semicolon.

Strictly, the output is comma-separated values (CSV) only if the delimiter is a comma. Otherwise, the output is more correctly known as delimiter-separated values (DSV). CSV is a specific type of DSV that uses a comma as the delimiter.

NOLABELS | LABELS
LABELS includes a header row with field labels. The default **NOLABELS** omits the header row.

FIXEDWIDTH
Right-pads field values, including numeric field values, with blanks to the maximum width of each field.

The maximum output width of numeric fields is 9 characters, including any decimal point. For example, the floating-point value `12.456`, which has an output width of 6 characters (2-digit integer part, a decimal point, and a 3-digit decimal fraction), will be right-padded with 3 blanks. The 4-digit integer value 1234 will be right-padded with 5 blanks.

By default, the maximum output width of a character field depends on the original log record definition. To set a different maximum output width, use the **WIDTH** parameter of the **FIELDS** qualifying command. For example, if a character field has a maximum width of 8 characters, then the 4-character value `MYZ1` will be right-padded with 4 blanks.

Specifying **FIXEDWIDTH** causes the **DB2LOAD** parameter to create field specifications that use the **POSITION** option to determine column boundaries, rather than relying on delimiters.

TYPECOLUMN
Inserts the **TYPE** parameter value as the second column of each output CSV data row.

Some analytics platforms can use data in each incoming event to detect the event type, and then interpret the data accordingly. Using **TYPECOLUMN** to include such data as a column in each CSV row enables analytics platforms to interpret a stream of heterogeneous CSV rows.

If the **LABELS** parameter is also specified, then the corresponding header row label is `Type`, `TYPE`, or `type`, depending on whether the **FIELDCASE** parameter specifies `ASIS`, `UPPER`, or the default value `LOWER`. However, if you are writing heterogeneous CSV rows (rows with different column layouts) from multiple **CSV** commands to the same destination, such as a stream, then it is unlikely that you want to include header rows in the output.

DB2LOAD
Writes a DB2 load utility **LOAD** control statement that loads the CSV data output by this **CSV** command into a DB2 table.

To create JCL that runs the DB2 load utility with this generated **LOAD** statement, see Transaction Analysis Workbench ISPF dialog option 5 **Analytics**.

By default, **DB2LOAD** writes a **LOAD** control statement that uses **FORMAT DELIMITED** to determine column boundaries. To write a **LOAD** control statement that uses **POSITION** instead, specify **DB2LOAD** and **FIXEDWIDTH**.

To specify the DB2 table name in the **LOAD** statement, use the **TABLE** parameter. If you omit the **TABLE** parameter, the **LOAD** statement uses the value of the **OUTPUT** parameter as a placeholder for you to replace later with an actual table name.

To write a corresponding DB2 DDL **CREATE TABLE** statement, specify the **SCHEMA** parameter.

HCATALOG
Writes a Hive DDL **CREATE TABLE** statement (also referred to as an HCatalog table schema) that corresponds to the output CSV data, for use with Hadoop and Hadoop-based software such as IBM BigInsights and Cloudera. The **CREATE TABLE** statement creates a catalog table; specifically, an external table that uses data in the CSV files stored in the HDFS directory referred to by the **LOCATION** parameter.

Abbreviation: **HCAT**

To specify the table name, use the **TABLE** parameter. If you omit **TABLE**, the default table name is the ddname specified by the **HCATALOG** parameter.

To specify the value of the **LOCATION** keyword of the **CREATE TABLE** statement, use the **LOCATION** parameter:

LOCATION
The path of the HDFS directory that contains, or will contain, one or more CSV files that match the HCatalog table schema.

If you omit the **LOCATION** parameter from the **CSV** command, then the **LOCATION** keyword in the **CREATE TABLE** statement specifies the placeholder value HDFS_PATH for you to replace later with an actual HDFS path.

JSON-only parameters

The following parameters apply only to the **JSON** command. If you specify these parameters on a **CSV** command, they are ignored, with one exception: you cannot specify **HORIZONTALSECTIONS** on a **CSV** command.

LINES | ARRAY
LINES writes log data in JSON Lines format. Each output line is a JSON object that represents a log record. This is the default behavior.

ARRAY writes log data in a single JSON array. Each element of the array is a JSON object that represents a log record.

FLAT(YES|NO)
FLAT(YES) writes "flat" JSON: each JSON object is a sequence of key-value pairs without nested structures. This is the default behavior. You can abbreviate FLAT(YES) as FLAT.

FLAT(NO) writes JSON objects that reflect the structure of the original log records. Fields are nested in sections.

For details, see “Flat JSON versus JSON with nested structures” on page 233.

MISSING(EXCLUDE|INCLUDE)
Specifies whether to exclude or include fields that are selected for output but are missing from the input log record.

MISSING(EXCLUDE) excludes such missing fields from the output. This is the default behavior.

MISSING(INCLUDE) includes such missing fields in the output, with the JavaScript value null.

HORIZONTALSECTIONS
Only relevant to log records that contain repeating sections. Writes instances of the specified sections as array elements, keeping the values of each instance separate, rather than merging them into a single set of values for the section. For details, see “Section parameters” on page 521.

Common parameters

The following parameters apply to the **CSV** command and the **JSON** command.

CODE
Specifies the log type and code of the records to write to the CSV or JSON output.

You can limit the fields included in the output by specifying a **FIELDS** parameter, section parameters, or both.

If you do not specify **CODE**, you must specify **FORM**.

`CODE(CMF)` is equivalent to `CODE(CMF:6E13)`.

SECTIONS
SECTIONS is one of several parameters that specify which sections of the input log records to include in the output, and how to handle repeating sections.

By default, the output includes all sections.

For details, see “Section parameters” on page 521.

FIELDS
Specifies which fields of the input log records to include in the output.

By default, the output includes all fields.

You can specify **FIELDS** either as a parameter of a **CSV** or **JSON** command, or as a qualifying command immediately following a **CSV** or **JSON** command.

For details, see the **FIELDS** command.

FORM
Specifies the name of a form that identifies the log type and code of the records to write to the CSV or JSON output, and which fields to write.

The form must be defined in the control repository that is specified by the FUWCDS ddname.

If you do not specify **FORM**, you must specify **CODE**.

ALLNUMBERS
Selects all numeric fields from the input log records.

To select additional, non-numeric, fields by name, use the **FIELDS** or **FORM** parameter.

FIELDCASE
Specifies whether to write field names or labels "as is", in all uppercase, or in all lowercase (the default):

ASIS
The case specified by the **FIELDS** parameter. For example, `FIELDS(CPUTIME:cpuTIME)` and `FIELDS(cpuTIME)` both result in `cpuTIME`

Otherwise, if the **FIELDS** parameter is omitted, the case of the field name as defined by Transaction Analysis Workbench.

UPPER
All uppercase. For example: `CPUTIME`

LOWER
All lowercase. For example: `cputime`

OUTPUT(*ddname*)
Writes the CSV or JSON output to the specified ddname. For the **CSV** command, the default ddname is CSV. For the **JSON** command, the default ddname is JSON.

To include the current date or time in the output MVS data set name or z/OS UNIX file path:

- When saving to an MVS data set, use **OUTPUT** with a **NEWNAME** parameter.
- When saving to a z/OS UNIX file path, use the **OUTPATH** parameter, not **OUTPUT**.

Note: Only use **OUTPATH** if you need the output z/OS UNIX file path to include the current date or time. Otherwise, use **OUTPUT** to refer to a ddname that identifies a z/OS UNIX file path.

NEWNAME
Renames the CSV or JSON output after it has been saved to the MVS data set specified by the **OUTPUT** parameter.

Specify **NEWNAME** if you want the output MVS data set name to contain the current date or time.

The **NEWNAME** parameter value specifies an MVS data set name that can contain the following substitution variables:

+DATE
Substituted with the current date in the format D*yymmdd* (the letter D followed by 2-digit year, month, and day)

+TIME
Substituted with the current time in the format T*hhmmss* (the letter T followed by 2-digit hour in 24-hour format, minutes, and seconds)

For example, if the date 30 November 2015 is and the time is 14:45:57, then the following parameter:

```
NEWNAME(SMFX.+DATE.+TIME.CSV)
```

renames the MVS data set that is specified by the **OUTPUT** parameter to:

```
SMFX.D151130.T144557.CSV
```

To append the date and time to the data set name that is specified by the **OUTPUT** parameter, specify two consecutive asterisks (**) as the high-level qualifier in **NEWNAME**. For example, if the output data set name is BIGDATA.CSV, then the following parameter:

```
NEWNAME(**.+DATE.+TIME)
```

renames the data set to:

```
BIGDATA.CSV.D151130.T144557
```

The **NEWNAME** parameter uses IDCAMS (the z/OS DFSMS Access Method Services IDCAMS utility) to rename the data set. IDCAMS messages are written to the FUWPRINT data set. Message IDC0531I indicates that the rename was successful.

OUTPUT(STDOUT)
Writes the CSV or JSON output to stdout, the standard output stream. Only applicable if the report and extract utility program, FUWBATCH, is running in a z/OS UNIX environment, such as a z/OS UNIX shell script.

OUTPATH
Writes the CSV or JSON output to the specified z/OS UNIX file path.

The **OUTPATH** parameter value can contain the same +DATE and +TIME substitution variables as the **NEWNAME** parameter.

For example, if the date 30 November 2015 is and the time is 14:45:57, then the following parameter:

```
OUTPATH(/u/analytics/+DATE+TIME.csv)
```

creates the following file:

```
/u/analytics/D151130.T144557.csv
```

See also "PATHMODE and PATHOPTS parameters" on page 525.

OUTPUT(STDOUT)
Writes the CSV or JSON output to stdout, the standard output stream. Only applicable if FUWBATCH is running in a z/OS UNIX environment, such as a z/OS UNIX shell script.

STREAM
Writes the CSV or JSON output to the network stream that is defined by the named **STREAM** command.

Specifying **STREAM** sets the following default parameters: `ASCII EOL(LF)`. You can override these defaults by explicitly specifying different parameters.

FILTER
Refers to the name of a filter that selects the input log records to write to the CSV or JSON output. The filter must be defined in the control repository specified by the FUWCDS ddname.

The filter must, as a minimum, select the log type and code specified by the **FORM** or **CODE** parameter. Otherwise, there will be no CSV or JSON output.

If the filter refers to forms, the forms are ignored.

A filter is optional, but is recommended to reduce output. If you do not specify a filter, all log records of the log type and code specified by the **FORM** parameter or **CODE** parameter are selected.

METADATA
Writes a JSON object containing metadata that describes the data in the CSV or JSON output. For each field in the output, the metadata includes the data type, such as numeric or string, and a short description. Applications can use this metadata to interpret and format the data.

SCHEMA
Writes a DB2 DDL **CREATE TABLE** statement (sometimes referred to as a table schema) that corresponds to the data in the CSV or JSON output. You can use the schema to create a DB2 table into which you can import the data.

To specify the table name, use the **TABLE** parameter. If you omit the **TABLE** parameter, the **CREATE TABLE** statement uses the value of the **OUTPUT** parameter as a placeholder for you to replace later with an actual table name.

TABLE
Specifies the table name used by the **DB2LOAD**, **SCHEMA**, and **HCATALOG** parameters. For example:

```
TABLE(MYSCHEMA.CICSPERF)
```

TYPE
Specifies a string that characterizes the type of data in the CSV or JSON output.

The effect of the **TYPE** parameter depends on whether it is specified on a **CSV** command or a **JSON** command:

- On a **CSV** command, the **TYPE** parameter has the following effects:
  - If the **LOGSTASHCONFIG** parameter is also specified, **TYPE** sets the value of the **document_type** setting in the Logstash config.
  - If the **TYPECOLUMN** parameter is also specified, **TYPE** sets the value of the type column in the CSV data.

  Otherwise, if a **CSV** command specifies neither **LOGSTASHSCONFIG** nor **TYPECOLUMN**, the **TYPE** parameter has no effect.
- On a **JSON** command, the **TYPE** parameter sets the value of the `type` property in each output event.

The default value is the log type and code of the input log records in lowercase, separated by a hyphen. If the **VERTICALSECTION** parameter is specified, the default value ends with the section name in lowercase, separated from the log code by another hyphen.

Examples:

- The following **JSON** command, without a **TYPE** parameter:

```
JSON … CODE(DTR:001) …
```

  writes events that contain the following field:

```
"type":"dtr-001"
```

- The following **JSON** command, with a **VERTICALSECTION** parameter:

```
JSON … CODE(DTR:001) SECTIONS(HEADER) VERTICALSECTION(QWSA) …
```

  writes events that contain the following field:

```
"type":"dtr-001-qwsa"
```

- The following **JSON** command, with an explicit **TYPE** parameter:

```
JSON … TYPE(DB2-stats-control) …
```

  writes events that contain the following field:

```
"type":"DB2-stats-control"
```

LOGSTASHCONFIGURATION

Writes a Logstash configuration file ("config") that corresponds to the CSV or JSON output data, for use with Elasticsearch.

Abbreviations: **LOGSTASHCONFIG**, **LSCONFIG**

The Logstash config for JSON is the same for all log record types. If you forward logs as JSON (more specifically, JSON Lines), then you can use a single Logstash config to ingest all log record types from Transaction Analysis Workbench.

However, Logstash configs for CSV are specific not only to each log record type, but also to the particular set of fields that you select to forward with each **CSV** command. Logstash configs for CSV contain a `columns` option that identifies numeric fields. Without the `convert` option, Logstash would forward all values as strings. The Logstash config for JSON does not require a `convert` option, because JSON inherently distinguishes between numeric and string values.

HOST

The value of the **hosts** option in the output section of the Logstash config. The **hosts** option refers to the system running Elasticsearch. The default value is localhost, the system running Logstash.

LOGSTASHINDEX

The value of the **index** property in the output section of the Logstash config. The **index** property specifies the name of the Elasticsearch index.

Abbreviation: **LSINDEX**

Default:

```
index => "fuw-%{type}-%{+YYYY.MM.dd}"
```

The Logstash config created by the **LOGSTASHCONFIGURATION** parameter contains the following date filter:

```
date {
  match => ["time", "ISO8601"]
}
```

The date filter uses the value of the `time` (event time stamp) field in the CSV or JSON data to set the value of the Logstash `@timestamp` field. The date filter assumes that the value of the `time` field matches the pattern specified by the Logstash format literal `ISO8601`. That is true only if the **CSV** or **JSON** command specifies a corresponding **TIMEFORMAT** parameter, such as `TIMEFORMAT(ISO8601)`. If the **CSV** or **JSON** command outputs time stamps in a different format, such as the default JDBC format, then you must edit the Logstash config and specify an appropriate pattern in the date filter.

TOKENS

Appends fields known as "correlation tokens" to the end of each output record:

TOKENS parameter value	Field name inserted in output	Description
ALL	All token fields: `ACCTOKEN`, `LUWID`, `IMSTOKEN`	A synonym for `TOKENS(CICS,DB2,IMS)`. All correlation tokens are appended to the record. **ALL** is recommended when you need to correlate CSV or JSON outputs that contain data from different subsystems. For example, the DB2 accounting record will contain the IMS recovery token when the connection to DB2 is IMS ("MASS"). **Tip:** If you want all correlation tokens, but with the `LUWID` value as generated by `TOKENS(DB2+)` (that is, decremented), specify `TOKENS(ALL,DB2+)`.
CICS	`ACCTOKEN`	CICS accounting token.
DB2	`LUWID`	DB2 logical unit of work ID (LUWID).
DB2+	`LUWID`	Same as `TOKENS(DB2)` except that the LUWID commit count in the log record is decremented by 1. For some records that DB2 writes at commit time, DB2 increments the LUWID commit count *before* writing the record, so the LUWID commit count in the record is incorrect. For these records, such as DB2 accounting records (log type DB2, log code 003), specify **DB2+**. The adjusted LUWID will reflect the correct unit of work, and can be used for correlation between subsystems. For example, the LUWID `FTS3/DBA6LU/CC17F01EF3DA/2` will be converted to `FTS3/DBA6LU/CC17F01EF3DA/1`, which is the actual LUWID of the DB2 thread.
IMS	`IMSTOKEN`	IMS recovery token.

For more information about these tokens, see Chapter 55, “Token fields to correlate activity within and between subsystems,” on page 255.

ASCII | EBCDIC

The output encoding: ASCII or EBCDIC. The default is EBCDIC, with one exception: if the **STREAM** parameter is specified, the default is ASCII.

CODEPAGE

Only applies if the **EBCDIC** parameter is specified. **CODEPAGE** specifies the EBCDIC code page that is used to encode square brackets in the following outputs:

- Log data converted to JSON by the **JSON** command
- JSON-format metadata created by the **METADATA** parameter
- Logstash configuration file created by the **LOGSTASHCONFIGURATION** parameter of the **CSV** command

You must specify the code page identifiers exactly as shown in the syntax. The default is 1047.

You can specify **CODEPAGE** either as a parameter of a **CSV** or **JSON** command, or as a separate command. The **CODEPAGE** command applies to all subsequent **CSV** and **JSON** commands. Specifying **CODEPAGE** as a parameter overrides the **CODEPAGE** command, but only for the current **CSV** or **JSON** command.

If you specify the **ASCII** parameter, **CODEPAGE** is ignored for this **CSV** or **JSON** command.

EOL(CR|CRLF|LF|NL|NONE)
The end-of-line (EOL) delimiter. Appends one or more bytes to the end of each output record. The default value is `EOL(NONE)` (no delimiter), with one exception: if **STREAM** is specified, the default is `EOL(LF)`.

The actual bytes appended depend on which **EOL** parameter value you specify, and whether you specify the **ASCII** or **EBCDIC** parameter.

EOL parameter value	Description	ASCII	EBCDIC
CR	Carriage return (\r)	X'0D'	X'0D'
CRLF	Carriage return and line feed (\r\n): the Windows EOL character pair	X'0D0A'	X'0D25'
LF	Line feed (\n): the default UNIX EOL character	X'0A'	X'25'
NL	Newline	*Not applicable*	X'15'

When writing to a z/OS UNIX file, the appropriate EOL delimiter depends on whether you want to write output encoded in ASCII or EBCDIC:

- To write ASCII-encoded output to a z/OS UNIX file:
  - On the DD statement for the file, use the default **FILEDATA** parameter value, `FILEDATA=BINARY`. Do not specify `FILEDATA=TEXT`. If you specify `FILEDATA=TEXT`, the output file will contain an EBCDIC newline character (X'15') at the end of each line, which is not appropriate for an ASCII-encoded file.
  - On the **CSV** or **JSON** command, specify the **ASCII** parameter and your choice of **EOL** parameter value: **CR**, **CRLF**, or **LF**.
- To write EBCDIC-encoded output to a z/OS UNIX file, *either*:
  - On the DD statement for the file, use the default **FILEDATA** parameter value, `FILEDATA=BINARY`, and on the **CSV** or **JSON** command, specify `EOL(NL)`.
  - On the DD statement for the file, specify `FILEDATA=TEXT`, and on the **CSV** or **JSON** command, specify `EOL(NONE)` (the default value).

  Do not specify the combination of `FILEDATA=TEXT` on the DD statement *and* an end-of-line delimiter on the **CSV** or **JSON** command; if you do, each output record will have two EOL delimiters: the delimiter specified by the **CSV** or **JSON** command followed by the X'15' appended by `FILEDATA=TEXT`.

  The default output EBCDIC code page is 1047. For details, see the **CODEPAGE** parameter.

TIMEFORMAT
Specifies the format of time stamp field values in output CSV or JSON data.

For more details, see Chapter 48, “Time stamps in CSV and JSON,” on page 239.

OUTLIM
Limits the number of output CSV or JSON records. The default is no limit.

The **OUTLIM** value must be a positive integer up to 99999999.

The **OUTLIM** value refers to the number of output records *containing data*; it does not include the header row in a CSV file, if requested, or records at the start and end of a JSON file that ensure valid JSON syntax.

If you specify the **VERTICALSECTION** parameter, then a single input log record containing multiple instances of the specified section will generate multiple output records. **OUTLIM** refers to the multiple output records, not the single input log record.

MB
Changes the behavior of the `OUTLIM(n)` parameter to limit output to *n* megabytes rather than *n* output records. Output ends at the last complete record that fits inside the limit.

Related concepts

Log forwarding
You can run Transaction Analysis Workbench in batch jobs to forward logs in CSV or JSON format to analytics platforms off z/OS. The jobs can either stream data over a network to a TCP socket, or write to files on z/OS, and then transfer the files.

DB2 table schema
When extracting log data to CSV or JSON, you can optionally request a DB2 DDL **CREATE TABLE** statement (sometimes referred to as a table schema) that describes the field names and data types in the CSV or JSON output. You can use the schema to create a DB2 table into which you can import the data.

JSON-format metadata that describes the CSV or JSON data
When extracting logs to CSV or JSON, you can optionally request metadata that describes the fields in the CSV or JSON output. This metadata is in JSON format.

Related tasks

Forwarding logs to analytics platforms
You can use Transaction Analysis Workbench to forward logs in comma-separated values (CSV) or JavaScript Object Notation (JSON) format to various analytics platforms.

Creating CSV files from log files in your ad hoc list
You can use Transaction Analysis Workbench to save data from any supported log record type to a comma-separated values (CSV) file. You can use the CSV file with other applications, such as PC-based spreadsheet applications. A CSV file created by Transaction Analysis Workbench contains selected fields from a single log record type and code.

Using forms to specify the columns in CSV files
When using the ISPF dialog to create a comma-separated values (CSV) file, you must specify a form name. The form identifies the type of log record and which fields to write to the CSV file.

Extracting logs to CSV or JSON
Transaction Analysis Workbench can extract logs from their original z/OS-based formats to comma-separated values (CSV) or JavaScript Object Notation (JSON) format for ingestion by analytics platforms and other applications.

Related reference

MWP command
Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

STREAM command
Forwards log data in JSON Lines or CSV format to a TCP socket.

ZONE command

Specifies the time zone that Transaction Analysis Workbench uses for the default time zone, the output time zone, and the time zone for the **START** and **STOP** commands.

FIELDS command
Specifies which fields to include in the output of the **CSV** and **JSON** commands.

OUTZONE command
Specifies the time zone of time stamps in output CSV and JSON data.

SCHEMAONLY command
Allows the **CSV** and **JSON** commands to produce some outputs, such as table schemas, without any input logs. These "schema-only" outputs rely on knowledge about log record types that is built into Transaction Analysis Workbench.

Section parameters

The section parameters of the **CSV** and **JSON** commands specify which sections of a log record to include in the output, and how to handle repeating sections.

Section parameters are for specialized use cases only. In many cases, you can ignore section parameters, and use the **FIELDS** command to specify which fields of a log record to include in the output.

Section parameters are only for use with the **CODE** parameter. If you specify a **FORM** parameter instead of **CODE**, section parameters are ignored.

If you do not specify any section parameters, the default is `SECTIONS(ALL)`: all sections are written to the output, repeating sections are merged.

Specifying which sections, and which fields from those sections, to output

Section parameters are cumulative; together, they define which sections to write to the output.

To specify which fields of the input log records to include in the output, you can use section parameters, the **FIELDS** command, or both. In the following rules, "specify fields" means "specify a **FIELDS** command" and "specify sections" means "specify one or more section parameters":

Table 29. Specifying combinations of fields and sections

If you specify...	Then the output contains...
Fields and sections	Only the specified fields that belong to specified sections. If you specify a field but not its section, the field is ignored. Fields in a repeating section are handled according to which section parameter specified that section.
Fields but not sections	The specified fields. Fields in repeating sections are merged as if those sections had been specified using the **SECTIONS** parameter.
Sections but not fields	All fields in the specified sections.

For JSON only: if a selected section does not occur in an input log record, then that section will not be represented in the output in any way; neither the section property nor any of its field properties will appear in the output.

Handling repeating sections

Section parameters offer different ways of handling *repeating sections*: sections that occur multiple times in a single record. To merge multiple instances of a repeating section into one, use **SECTIONS** or **EXCLUDESECTIONS**. To output each instance of a repeating section separately, use **VERTICALSECTION** or **HORIZONTALSECTIONS**.

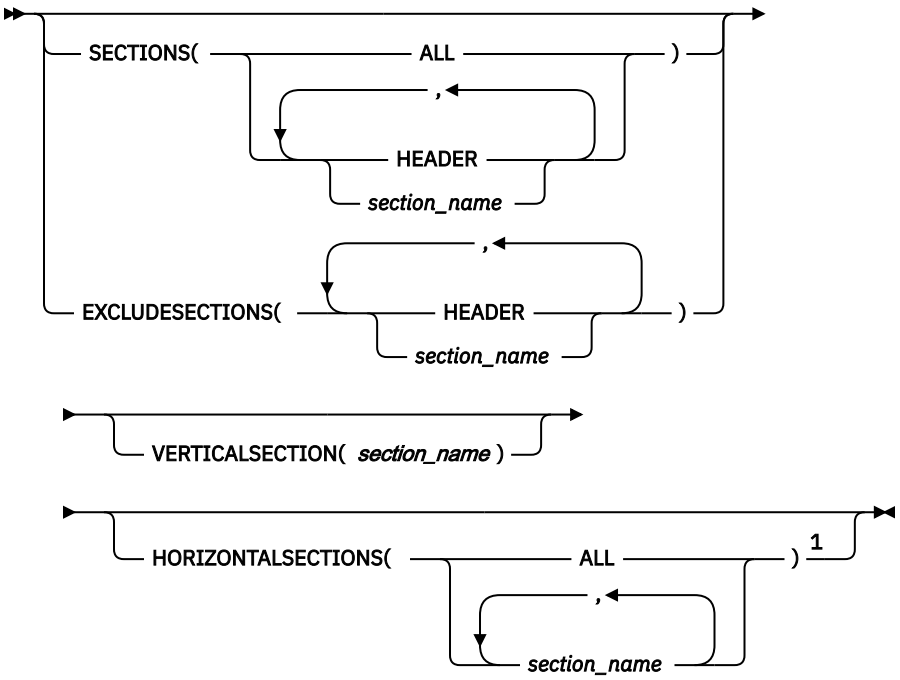
Reserved section names

The following section names are reserved and have specific meanings:

HEADER
Refers to the starting section of any record

ALL
Refers to all sections of a record

Format



Notes:

[1] The **HORIZONTALSECTIONS** parameter applies only to the **JSON** command.

SECTIONS | EXCLUDESECTIONS
SECTIONS includes one or more sections in the output.

EXCLUDESECTIONS includes all sections except for one or more specified sections.

SECTIONS and **EXCLUDESECTIONS** merge repeating sections into a single composite section in the output according to the following rules:

- Character fields are concatenated
- Numeric fields (times and counts) are accumulated
- Timestamp fields take the value from the first instance of the repeating section

VERTICALSECTION
Specifies a section in the log record to include in the output. If the section repeats, each instance of the section is written to a separate output record.

The field values of any other sections in these output records are fixed, and are repeated in each output record. For example, if an input log record contains six instances of the specified section, the output file represents that single input log record as six separate output records.

You can use **VERTICALSECTION** for CSV or JSON. However, for JSON, you would typically use **HORIZONTALSECTIONS** instead of **VERTICALSECTION**. **VERTICALSECTION** works around the

limitations of the two-dimensional CSV format, whereas **HORIZONTALSECTIONS** uses the inherent features of JSON to represent hierarchical data structures.

HORIZONTALSECTIONS
For JSON only. Specifies one or more sections in the log record to include in the output. If a section repeats, all instances of the section are written to the same JSON object.

Tip: To use this method for all sections, specify `HORIZONTALSECTIONS(ALL)`.

The syntax of the JSON output depends on whether the **FLAT** parameter is also specified:

- Without **FLAT**, **HORIZONTALSECTIONS** writes each instance of a repeating section as an array element. For example:

```
"data": {
  "dsecta": [
    {"smfxxa1": "First instance of dsecta", "smfxxa2": 1},
    {"smfxxa1": "Second instance of dsecta", "smfxxa2": 2}
  ]
}
```

  where `dsecta` is the name of a repeating section.
- With **FLAT**, **HORIZONTALSECTIONS** writes each instance of a repeating section as a flat sequence of fields, one after another, with no nested structures, and no indication of where each instance begins or ends:

```
"smfxxa1": "First instance of dsecta",
"smfxxa2": 1,
"smfxxa1": "Second instance of dsecta",
"smfxxa2": 2
```

  The resulting JSON contains duplicate keys. How that JSON is interpreted depends on which parser you use. See your JSON parser documentation for details.

SECTIONS and **EXCLUDESECTIONS** are mutually exclusive.

VERTICALSECTION and **HORIZONTALSECTIONS** override **SECTIONS** and **EXCLUDESECTIONS**. For example:

```
SECTIONS(ALL)
VERTICALSECTION(QWSB)
```

writes all sections of the log record to the output. Each instance of the QWSB repeating section is written to a separate output record. Values of all other repeating sections are merged.

If you specify both **HORIZONTALSECTIONS** and **VERTICALSECTION**, then whichever of these two parameters is specified later in the SYSIN data set takes effect; the earlier parameter is ignored.

Listing sections in a record type

To get the list of section names for a particular combination of log type and code, use one of the following options in the Transaction Analysis Workbench ISPF dialog:

- Browse a log record of that type and code. A blank line precedes each section. The section names are displayed in yellow, followed by a description in white.
- Create a form for that log type and code. A horizontal line precedes each section. Each section name starts a new ID column value.

Browsing a log record only shows sections that contain data; a new form shows all sections. For log type and code combinations that can contain many sections, you might prefer to browse log records, so that you can see which sections contain data in the log records that you are working with, rather than specifying sections that are empty.

Example: Repeating "control address space" sections in DB2 systems statistics trace records

Each DB2 systems statistics trace (SMF type 100, IFCID 1; log type DTR, log code 001) record contains repeating "control address space" sections: one per address space. To make these statistics more accessible to analytics platforms, you can use the **VERTICALSECTION** parameter of the **JSON** command to output each instance of the repeating section as a separate event:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD  PATH='/u/myid/fuw.json',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             FILEDATA=TEXT
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
OUTZONE(Z)
JSON CODE(DTR:001) +
  SECTIONS(HEADER) VERTICALSECTION(QWSA) +
  TIMEFORMAT(ISO8601)
FIELDS(
  SM102SID,SM102SSI,
  QWSAPROC,QWSAEJST,QWSASRBT,QWSAPSRB,QWSAPSRB_zIIP
)
/*
```

Example snippet of JSON Lines output (presented here with additional line breaks and indenting):

```
{"time":2017-01-13T08:30:03.141592Z,"type":"dtr-001-qwsa",←
    "sm102ssi":"DB2A","qwsaproc":"MSTR","qwsaejst": … }
{"time":2017-01-13T08:30:03.141592Z,"type":"dtr-001-qwsa",←
    "sm102ssi":"DB2A","qwsaproc":"DBM1","qwsaejst": … }
{"time":2017-01-13T08:30:03.141592Z,"type":"dtr-001-qwsa",←
    "sm102ssi":"DB2A","qwsaproc":"DIST","qwsaejst": … }
{"time":2017-01-13T08:30:03.141592Z,"type":"dtr-001-qwsa",←
    "sm102ssi":"DB2A","qwsaproc":"IRLM","qwsaejst": … }
{"time":2017-01-13T08:30:03.70796Z,"type":"dtr-001-qwsa",←
    "sm102ssi":"DB2B","qwsaproc":"MSTR","qwsaejst": … }
⋮
```

Note:

- Events from repeating sections in the same input log record have the same the time stamp (`time` field value).
- In this example, the `qwsaproc` field values are the last four characters of the procedure name used to start the address space.

Example: Repeating Repeating "DASD data set I/O statistics" sections in DFSMS SMF records

Each DFSMS (SMF type 42 subtype 6) record contains repeating "DASD data set I/O statistics" sections: one per data set. The following example JCL outputs the statistics for each data set as a separate CSV row:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV      DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(SMF:42.6) +
  LABELS +
  SECTIONS(HEADER,SMF42JOB,SMF42IOS) VERTICALSECTION(SMF42DSH)
FIELDS(
  SMF42SID, S42JDJNM,
* Data set section
  S42DSN,   S42DSTYP, S42DSCOD, S42DSFL1, S42DSVOL, S42DSDEV, S42DSSC,
  S42DSBSZ, S42DSTRP,
* I/O section
  S42DSIOR, S42DSIOC, S42DSIOP, S42DSIOD, S42DSIOQ, S42DSION, S42DSCND,
  S42DSHTS, S42DSWCN, S42DSWHI, S42DSSEQ, S42DSRLC, S42DSICL, S42DSDAO,
  S42DSMXR, S42DSMXS, S42DSRDD, S42DSRDT
```

```
)
/*
```

Note: Section SMF42IOS does not repeat, but is nested inside the repeating SMF42DSH section.

Related concepts

Log records that contain repeating sections

When converting logs to CSV or JSON, Transaction Analysis Workbench offers several methods for handling *repeating sections*: groups of fields that occur more than once in a single record.

Related reference

FIELDS command

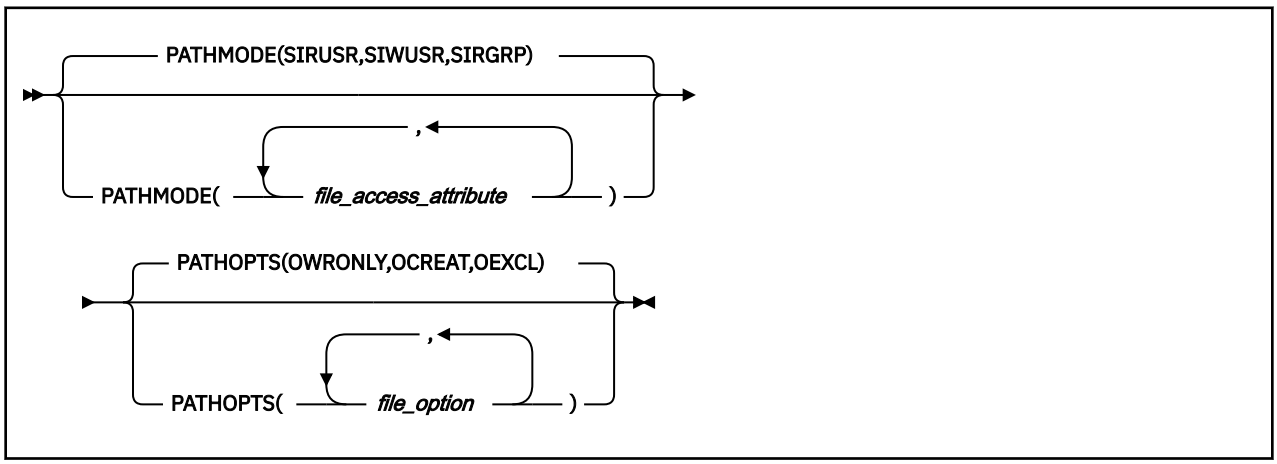
Specifies which fields to include in the output of the **CSV** and **JSON** commands.

PATHMODE and PATHOPTS parameters

The **PATHMODE** and **PATHOPTS** parameters of the **CSV** and **JSON** commands specify values for the z/OS MVS JCL DD statement parameters of the same name. Use **PATHMODE** and **PATHOPTS** to override the default values for these parameters when writing output to a z/OS UNIX file path.

You only need to use the **PATHMODE** and **PATHOPTS** parameters when all of the following conditions are true:

- You want to write CSV or JSON to a z/OS UNIX path.
- You want the file path to include the current date or time. So, you need to use the **OUTPATH** parameter to specify the path, rather than using **OUTPUT** to refer to a ddname.
- You want to override the default **PATHMODE** and **PATHOPTS**. For details, see the following parameter descriptions.



PATHMODE
File access attributes for the z/OS UNIX file named in the **OUTPATH** parameter. The default value ensures that a new file is always created. The allowed values match the **PATHMODE** parameter of the z/OS MVS JCL DD statement.

PATHOPTS
Access and status for the z/OS UNIX file named in the in the **OUTPATH** parameter. The default value allows user and group access. The allowed values match the **PATHOPTS** parameter of the z/OS MVS JCL DD statement.

CSV command examples

These examples present complete JCL for converting logs to CSV format.

Where an example shows only a SYSIN data set, the JCL is the same as the preceding example.

Converting all fields to CSV

The following JCL extracts a single CICS monitoring facility (CMF) performance class record from the dumped SMF data set 'HLQ.SMF.DAILY', converts all of its fields to CSV, and then writes the CSV data to SYSOUT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV      DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(CMF) LABELS OUTLIM(1)
/*
```

The **LABELS** parameter causes the **CSV** command to write a header row with field names. By default, the **CSV** command does not write a header row.

The **OUTLIM** parameter is useful for testing, to restrict processing to a limited number of output records.

The CSV data is encoded in EBCDIC code page 1047, the default output encoding of the **CSV** command.

The following JCL produces similar output, but reads today's first CMF record from an SMF log stream, instead of reading the first CMF record in a data set. A **LOGSTREAM** command at the start of the SYSIN data set replaces the SMFIN DD statement in the previous JCL.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CSV      DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0 /* Today
CMF CODE(CMF) LABELS OUTLIM(1)
/*
```

The following JCL extracts all DB2 system statistics trace (IFCID 001) records from a dumped SMF data set, converts the data to CSV, and then writes the data to an MVS sequential data set:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD DSN=MY.FUW.CSV,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(DTR:001) LABELS
/*
```

The previous examples write each CSV row to a separate record in EBCDIC, with no end-of-line markers.

The following JCL writes CSV to a z/OS UNIX file in EBCDIC, with a newline character (X'0A') at the end of each line. The output DD statement specifies FILEDATA=TEXT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV      DD PATH='/u/myid/fuw.csv',
//            FILEDATA=TEXT,
//            PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(DTR:001) LABELS
/*
```

The following JCL writes CSV to a z/OS UNIX file in ASCII, with a line feed character (X'0A') at the end of each line. The output DD statement specifies FILEDATA=BINARY and the **CSV** command specifies ASCII EOL(LF):

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV      DD PATH='/u/myid/fuw.csv',
//            FILEDATA=BINARY,
//            PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV CODE(DTR:001) LABELS ASCII EOL(LF)
/*
```

Specifying which fields to extract

By default, the **CSV** command extracts all fields of the selected record type.

There are several methods for specifying which fields to extract. The most straightforward method is to follow the **CSV** command with a **FIELDS** command that specifies a list of field names:

```
CSV CODE(SMF:30.)
FIELDS(
  SMF30JBN
  SMF30SIT
  SMF30STD
  SMF30CPT
)
```

Selecting which records to extract

To select which records to extract, follow the **CSV** command with a **CODE** command and subsequent **COND** statements that specify filter conditions.

The following example selects SMF type 30 records where both of the following conditions are true:

- The record is for a job step termination
- The job owner is MYID

```
CSV CODE(SMF:30.)
FIELDS(
  SMF30JBN
  SMF30SIT
  SMF30STD
  SMF30CPT
)
CODE(SMF:30.)              /* Must match CODE parameter of CSV command
  COND SMF30STP EQ 5       /* Record subtype for job step termination
  COND SMF30RUD EQ 'MYID'  /* RACF user ID of the job owner
```

Tip: Filter conditions can refer to fields that are not extracted.

Creating a CSV file using a form

The following JCL creates a CSV file from a dumped SMF data set, based on a form named SMF30 stored in the control repository MY.FUW.CONTROLS.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//FUWCDS   DD DISP=SHR,DSN=MY.FUW.CONTROLS
//CSV      DD DSN=MY.FUW.CSV,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
```

```
CSV FORM(SMF30)
/*
```

To overwrite an existing CSV file, use the following CSV DD statement:

```
//CSV DD DSN=MY.FUW.CSV,DISP=SHR
```

To append to an existing CSV file:

```
//CSV DD DSN=MY.FUW.CSV,DISP=MOD
```

Creating a CSV file from SMF type 30 records filtered by job name

The following JCL creates a CSV file of SMF type 30 job termination records for job names that begin with "UID":

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD  DISP=SHR,DSN=SYSA.PROD.SMF(0)
//CSV      DD  DSN=MY.SMF30.CSV,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
CSV CODE(SMF:30.) LABELS FIELDCASE(ASIS)
FIELDS(
   SMF30JBN:"Job name"
   SMF30CPT:"CPU time"
)
CODE(SMF:30.)
  COND SMF30JBN EQ 'UID*' /* Job name
  COND SMF30STP EQ 5      /* Job termination
/*
```

Creating CSV files for CICS, DB2, and MQ accounting

The following JCL creates three CSV files from the following log records:

- SMF 101: DB2 accounting classes 1, 2, 3, 7, and 9 (IFCID 003)
- SMF 110: CICS CMF performance class (transaction accounting)
- SMF 115: MQ accounting class 1 or 3

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=HLQ.SMF.DAILY
//CSVDB2   DD  DSN=MY.CSV,DB2,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//CSVCICS  DD  DSN=MY.CSV.CICS,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//CSVMQ3    DD DSN=MY.CSV.MQ.CLASS3,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//CSVMQ1    DD DSN=MY.CSV.MQ,CLASS1,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//LOGRPT   DD  SYSOUT=*
//SYSIN    DD  *
CSV OUTPUT(CSVDB2) CODE(DTR:003) SCHEMA(DB2S) +
    LABELS +
    SECTIONS(HEADER,QWAC,QMDA,QMDAINFO) +
    SECTIONS(QTXA,QTGA,QXST,QBAC,QWHS,QWHC,QWHA) +
    TOKENS(ALL,DB2+)

CSV OUTPUT(CSVCICS) CODE(CMF:6E13) SCHEMA(CICSS) +
    LABELS +
    SECTIONS(HEADER,DFHTASK,IMSDBCTL) +
    TOKENS(ALL)
```

```
CSV OUTPUT(CSVMQ3) CODE(SMF:74) SCHEMA(MQ1S) +
    LABELS +
    SECTIONS(HEADER,WTID,WTAS,WQSTAT,QWHS,QWHC) +
    TOKENS(ALL)
CODE(SMF:116.)
COND SM116STF EQ +1  /* Class 3 detailed statistics

CSV OUTPUT(CSVMQ1) CODE(SMF:74) SCHEMA(MQ3S) +
    LABELS +
    SECTIONS(HEADER,QWAC,QMAC,QWHS,QWHC) +
    TOKENS(ALL)
CODE(SMF:116.)
COND SM116STF EQ +0  /* Class 1 brief statistics
/*
```

Creating a CSV file for IMS accounting from the IMS log

The following JCL creates a CSV file for IMS transactions from the IMS log:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=IMSP.SLDS
//CSVIMS   DD  DSN=MY.IMS.CSV,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//LOGRPT   DD  SYSOUT=*
//SYSIN    DD  *
IMSVRM=131 /* IMS version 13
IMSINDEX   /* Create IMS transaction index records from the log
CSV OUTPUT(CSVIMS) CODE(IMS:CA01) SCHEMA(S1) +
    LABELS +
    SECTIONS(ALL) TOKENS(IMS)
/*
```

Extracting DB2 IFCID 001 to CSV and JSON

The DB2 system statistics record (IFCID 001) is an example of a complex record that contains many (unrelated) sections that sometimes repeat.

The following JCL creates four output files:

CSV1
Contains fixed (non-repeating) sections.

CSV2
Contains instances of the repeating section QWSA in separate records.

JSON1
Same as CSV1, but in JSON format.

JSON2
Contains instances of the repeating section QWSA as array elements.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=HLQ.SMF.DAILY
//CSV1     DD  DSN=MY.DTR001.CSV1,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//CSV2     DD  DSN=MY.DTR001.CSV2,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//JSON1    DD  DSN=MY.DTR001.JSON1,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//JSON2    DD  DSN=MY.DTR001.JSON2,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//LOGRPT   DD  SYSOUT=*
//SYSIN    DD  *
```

```
CSV OUTPUT(CSV1) CODE(DTR:001) SCHEMA(S1) +
    LABELS +
    SECTIONS(HEADER,Q9ST,QVLS,QVAS,QSST,QLST,QJST,QDST)

CSV OUTPUT(CSV2) CODE(DTR:001) SCHEMA(S2) +
    LABELS +
    SECTIONS(HEADER) VERTICALSECTION(QWSA)

JSON OUTPUT(JSON1) CODE(DTR:001) SCHEMA(S3) METADATA(M3) +
     SECTIONS(HEADER,Q9ST,QVLS,QVAS,QSST,QLST,QJST,QDST)

JSON OUTPUT(JSON2) CODE(DTR:001) SCHEMA(S4) METADATA(M4) +
     SECTIONS(HEADER) HORIZONTALSECTIONS(QWSA)
/*
```

Streaming heterogeneous CSV to a TCP socket

The following JCL streams three log record types in CSV format to a TCP socket on a remote system:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CSV      DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0 /* Today
STREAM NAME(STREAM1) HOST(splunk) PORT(6789) +
       TIMEFORMAT(ISO8601) TYPECOLUMN
* CICS performance
CSV CODE(CMF) STREAM(STREAM1)
FIELDS(
  TRAN, USRCPUT /* More fields...
)
* DB2 accounting
CSV CODE(DTR:003) STREAM(STREAM1)
FIELDS(
  SM101SID, SM101SSI,
  QWHCCTR, QWHCPLAN, QWHCOPID, QWHCCN,
  ET1, CPU1, ET2, CPU2 /* More fields...
)
* Job termination
CSV CODE(SMF:30.) STREAM(STREAM1)
FIELDS(
  SMF30JBN, SMF30CPT,  /* More fields...
)
CODE(SMF:30.)
  COND SMF30STP EQ 5
/*
```

The remote system must be configured to parse each CSV row based on the value of the "type" field, which is the second column in each row.

Creating CSV files for viewing in a spreadsheet-like table in the plug-in

To create a CSV file that you can view in a spreadsheet-like table in the Transaction Analysis Workbench plug-in, use a **CSV** command with the following parameters:

LABELS
Includes a header row of column labels.

OUTPUT(CSV*x*)
The ddname of the CSV file must begin with the characters CSV. For example: CSV, CSV1, CSVXYZ.

METADATA(META*x*)
The **CSV** command must create metadata for the CSV. The ddname of the metadata must begin with the characters META, and the remaining characters must match the ddname for the corresponding CSV file. For example, META, META1, METAXYZ.

DELIMITER(,)
The field delimiter must be a comma. This is the default value, so you do not need to explicitly specify this parameter.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//FUWCDS   DD DISP=SHR,DSN=MY.FUW.CONTROLS
//CSV1     DD SYSOUT=*
//META2    DD SYSOUT=*
//CSV2     DD SYSOUT=*
//META2    DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
CSV1 FORM(SMF30) OUTPUT(CSV1) METADATA(META1) LABELS
CSV2 FORM(CMF)   OUTPUT(CSV2) METADATA(META2) LABELS
/*
```

Related tasks

Scrambling character field values in CSV and JSON output
When converting logs to CSV or JSON, you can specify which character fields to scramble. Scrambling a field makes its original value unrecognizable.

Related reference

Report and extract utility JCL
You can use the Transaction Analysis Workbench ISPF dialog to create JCL for the utility, or you can write the JCL yourself.

JSON command examples

These examples present complete JCL for converting log data to JSON format.

Where an example shows only a SYSIN data set, the JCL is the same as the preceding example.

Converting all fields to JSON Lines

The following JCL extracts a single CICS monitoring facility (CMF) performance class record from the dumped SMF data set 'HLQ.SMF.DAILY', converts the data to JSON Lines, and then writes the JSON Lines data to SYSOUT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
JSON CODE(CMF) OUTLIM(1)
/*
```

The **OUTLIM** parameter is useful for testing, to restrict processing to a limited number of output records.

The JSON output is encoded in EBCDIC code page 1047, the default encoding of the **JSON** command.

The following JCL produces similar output, but reads today's first CMF record from an SMF log stream, instead of reading the first CMF record from a data set. A **LOGSTREAM** command at the start of the SYSIN data set replaces the SMFIN DD statement in the previous JCL.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//JSON     DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0 /* Today
JSON CODE(CMF) OUTLIM(1)
/*
```

The following JCL extracts all DB2 system statistics trace (IFCID 001) records from a dumped SMF data set, converts the data to JSON Lines, and then writes the data to an MVS sequential data set:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD DSN=MY.FUW.JSON,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
JSON CODE(DTR:001)
/*
```

The previous examples write each line of JSON Lines to a separate record in EBCDIC, with no end-of-line markers.

The following JCL writes JSON Lines to a z/OS UNIX file in EBCDIC, with a newline character (X'0A') at the end of each line. The output DD statement specifies `FILEDATA=TEXT`:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD PATH='/u/myid/fuw.json',
//            FILEDATA=TEXT,
//            PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
JSON CODE(DTR:001)
/*
```

The following JCL writes JSON Lines to a z/OS UNIX file in ASCII, with a line feed character (X'0A') at the end of each line. The output DD statement specifies `FILEDATA=BINARY` and the **JSON** command specifies `ASCII EOL(LF)`:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD PATH='/u/myid/fuw.json',
//            FILEDATA=BINARY,
//            PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
JSON CODE(DTR:001) ASCII EOL(LF)
/*
```

Specifying which fields to extract

By default, the **JSON** command extracts all fields of the selected record type.

There are several methods for specifying which fields to extract. The most straightforward method is to follow the **JSON** command with a **FIELDS** command that specifies a list of field names:

```
JSON LINES CODE(SMF:30.)
FIELDS(
  SMF30JBN
  SMF30SIT
  SMF30STD
  SMF30CPT
)
```

Selecting which records to extract

To select which records to extract, follow the **JSON** command with a **CODE** command and subsequent **COND** statements that specify filter conditions.

The following example selects SMF type 30 records where both of the following conditions are true:

- The record is for a job step termination
- The job owner is MYID

```
JSON LINES CODE(SMF:30.)
FIELDS(
  SMF30JBN
  SMF30SIT
  SMF30STD
  SMF30CPT
)
CODE(SMF:30.)              /* Must match CODE parameter of JSON command
  COND SMF30STP EQ 5       /* Record subtype for job step termination
  COND SMF30RUD EQ 'MYID'  /* RACF user ID of the job owner
```

Tip: Filter conditions can refer to fields that are not extracted.

Extracting multiple record types from an SMF log stream to z/OS UNIX files

The following JCL extracts two types of record written today to an SMF log stream, converts them to JSON Lines in ASCII, and then writes each record type to a separate z/OS UNIX file:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CMF      DD PATH='/u/myid/cmf.json',
//            PATHOPTS=(OWRONLY,OCREAT),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP),
//            FILEDATA=BINARY
//SMF30    DD PATH='/u/myid/smf30.json',
//            PATHOPTS=(OWRONLY,OCREAT),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP),
//            FILEDATA=BINARY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOGSTREAM SMF:IFASMF.PROD
START 0
JSON CODE(CMF) OUTPUT(CMF) +
     TIMEFORMAT(ISO8601) ASCII EOL(LF)
JSON CODE(SMF:30.) OUTPUT(SMF30) +
     TIMEFORMAT(ISO8601) ASCII EOL(LF)
/*
```

Writing JSON that preserves the structure of the original log records

Some types of log record can contain multiple instances of the same section, also known as repeating sections. By default, repeating sections are merged in the JSON output into a single section with a single set of fields. To keep repeating sections separate, represented in JSON as array elements, use the **HORIZONTALSECTIONS** parameter. For example, to keep all repeating sections separate, specify `HORIZONTALSECTIONS(ALL)`.

The following JCL creates a JSON file on z/OS UNIX that you can parse to reproduce the structure of the original log record, including repeating sections:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SMFIN    DD DISP=SHR,DSN=HLQ.SMF.DAILY
//JSON     DD  PATH='/u/myid/fuw.json',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             FILEDATA=BINARY
//METADATA DD SYSOUT=*
//SCHEMA   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
JSON CODE(DTR:001) +
     HORIZONTALSECTIONS(ALL) +
     METADATA(METADATA) +
```

```
     TIMEFORMAT(ISO8601) ASCII EOL(LF)
/*
```

This example also demonstrates using the **OUTPATH** parameter to write directly to a z/OS UNIX file, as an alternative to using the **OUTPUT** parameter to write to a ddname that refers a z/OS UNIX file.

Related tasks

Streaming logs as JSON Lines over TCP to an analytics platform
You can run Transaction Analysis Workbench batch jobs that stream log data in JSON Lines format over a network to a TCP socket on an analytics platform, such as Elastic or Splunk. The TCP socket can be secure or unsecure.

Scrambling character field values in CSV and JSON output
When converting logs to CSV or JSON, you can specify which character fields to scramble. Scrambling a field makes its original value unrecognizable.

Related reference

Flat JSON Lines for ingestion by analytics platforms
You can configure the **JSON** command with parameters to create JSON with different structures for different purposes. By default, the **JSON** command creates "flat" JSON Lines that is optimized for ingestion by analytics platforms such as Elastic or Splunk.

Report and extract utility JCL
You can use the Transaction Analysis Workbench ISPF dialog to create JCL for the utility, or you can write the JCL yourself.

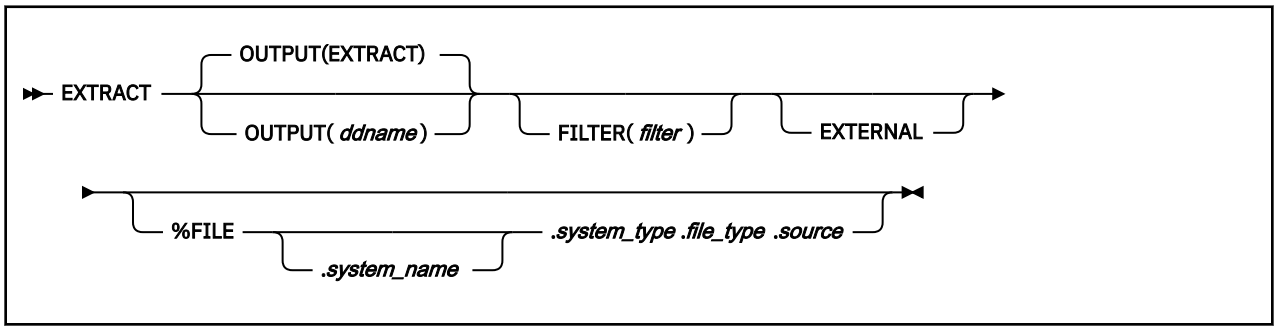
EXTRACT command

Writes the selected log records to an extract data set.

A SYSIN data set can contain zero or more **EXTRACT** commands.

To filter the input log records that are processed by this command, either use the **FILTER** parameter to refer to a filter in the control repository, or specify filter conditions in **CODE** commands and related **COND** statements immediately after this command.

Format



Parameters

ddname
The ddname of the data set where you want to save the extracted records. The default ddname is EXTRACT.

filter
The name of a filter that selects the log records you want to extract. The filter must be defined in the control repository specified by the FUWCDS ddname.

If the filter contains forms, the forms are ignored by the extract. Extract records have the same format as log records.

A filter is optional, but is recommended to reduce the size of the extract data set. If you do not specify a filter, all log records are selected.

EXTERNAL
Omits the control information that Transaction Analysis Workbench normally writes to an extract to identify different log record types. The format of this control information is specific to Transaction Analysis Workbench. Transaction Analysis Workbench uses the control information when it reads the extract.

If you plan to use the extract with another tool, then specify the **EXTERNAL** parameter to omit this control information; the extract will contain the requested record types only.

For example, specify **EXTERNAL** when creating an extract of SMF records for use with the Mobile Workload Reporting Tool (MWRT) to support Mobile Workload Pricing (MWP).

system_name*, *system_type*, *file_type*, *source
The details to be used when adding the extract to a session.

If you omit *system_name*, then the extract is added to the session using the system name from the first record in the extract.

Adding the extract to a session

To add the extract to the list of files for a session:

- In the JCL, insert an FUWPROBR DD statement that identifies the session repository
- In the SYSIN data set, insert a **SESSION** command that identifies an existing session. For example:

```
SESSION=00000001
```

- Add a **%FILE** parameter to the **EXTRACT** command. For example, for an SMF extract that you want to associate with an MVS image named ZOS1:

```
EXTRACT %FILE.ZOS1.IMAGE.SMF.EXTRACT
```

Example: Extracting SMF records

The following JCL extracts from an SMF file the records that were written today between 8:00 a.m. and 8:15 a.m. The JCL also creates an SMF Recap report that provides an overview of the contents of the extract, such as the types of SMF record that it contains.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SMFIN    DD DISP=SHR,DSN=SYSA.PROD.SMF(0)
//EXTRACT  DD DSN=MY.FUW.EXTRACT,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSIN    DD *
START 0-08.00
STOP  0-08.15
EXTRACT
REPORT SMF(RECAP) OUTPUT(SMFRECAP) /* SMF Recap report
/*
```

To overwrite an existing extract, use the following EXTRACT DD statement:

```
//EXTRACT  DD DSN=MY.FUW.EXTRACT,DISP=SHR
```

To append to an existing extract:

```
//EXTRACT  DD DSN=MY.FUW.EXTRACT,DISP=MOD
```

Example: Extracting SMF records for MWRT

The following JCL extracts from an SMF file the records required by the Mobile Workload Reporting Tool (MWRT):

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//SMFIN    DD DISP=SHR,DSN=SYSA.PROD.SMF(0)
//EXTRACT  DD DSN=MY.FUW.MWRT.SMF,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//* Insert DD statements here for MWP command log input and CSV output
//SYSIN    DD *

* Insert MWP commands here

EXTRACT EXTERNAL
  CODE(SMF:70-1)
  CODE(SMF:89-1)
  CODE(SMF:89-2)
/*
```

Typically, you would include in the same SYSIN data set one or more **MWP** commands to create the CSV files also required by MWRT.

Example: Extracting summary log records from an ATF log stream

The following JCL extracts OMEGAMON for IMS on z/OS Application Trace Facility (ATF) summary log records from an ATF log stream. The JCL extracts records that were written to the log stream today.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//EXTRACT  DD DSN=MY.FUW.ATF.EXTRACT,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SYSIN    DD *
LOGSTREAM ATF:MVS1.ATF
START 0 /* Today
EXTRACT
  CODE(ATF:04) /* Summary log records
/*
```

Related concepts

Mobile Workload Pricing

Mobile Workload Pricing (MWP) for z/OS is a pricing model that reduces the cost of running mobile workloads on z/OS. To take advantage of MWP, you use the Mobile Workload Reporting Tool (MWRT) to submit monthly reports to IBM. MWRT requires two types of input files: CSV and SMF. You can use Transaction Analysis Workbench to create these files.

Related tasks

Creating extracts of log files in a session

Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Creating transaction indexes

Creating transaction indexes from original log files is a useful preliminary step for problem analysis. Where original log files typically contain many types of record, a transaction index contains only a single type of record. Analyzing transaction index records is a useful starting point for analyzing problems with transactions.

Creating extracts of log files in your ad hoc list

Rather than working with original log files, you might find it more convenient to work with smaller extracts that contain only the log records that interest you.

Related reference

MWP command

Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

MWP command

Creates a customer mobile CPU consumption CSV (comma-separated values) file for use with the Mobile Workload Reporting Tool (MWRT), to support Mobile Workload Pricing for z/OS (MWP). The CSV file reports CPU time for mobile transactions that are associated with one of the following subsystems (also referred to as "product families"): CICS, DB2, IMS, WebSphere Application Server for z/OS, or IBM MQ.

MWRT requires two types of input files: CSV and SMF. You can use Transaction Analysis Workbench to create both types of files in a single batch job that performs a single pass of the log data. To create the CSV files, use the **MWP** command. To create the SMF files, use the **EXTRACT** command.

A SYSIN data set can contain multiple **MWP** commands. The same SYSIN data set can also contain multiple **EXTRACT** commands.

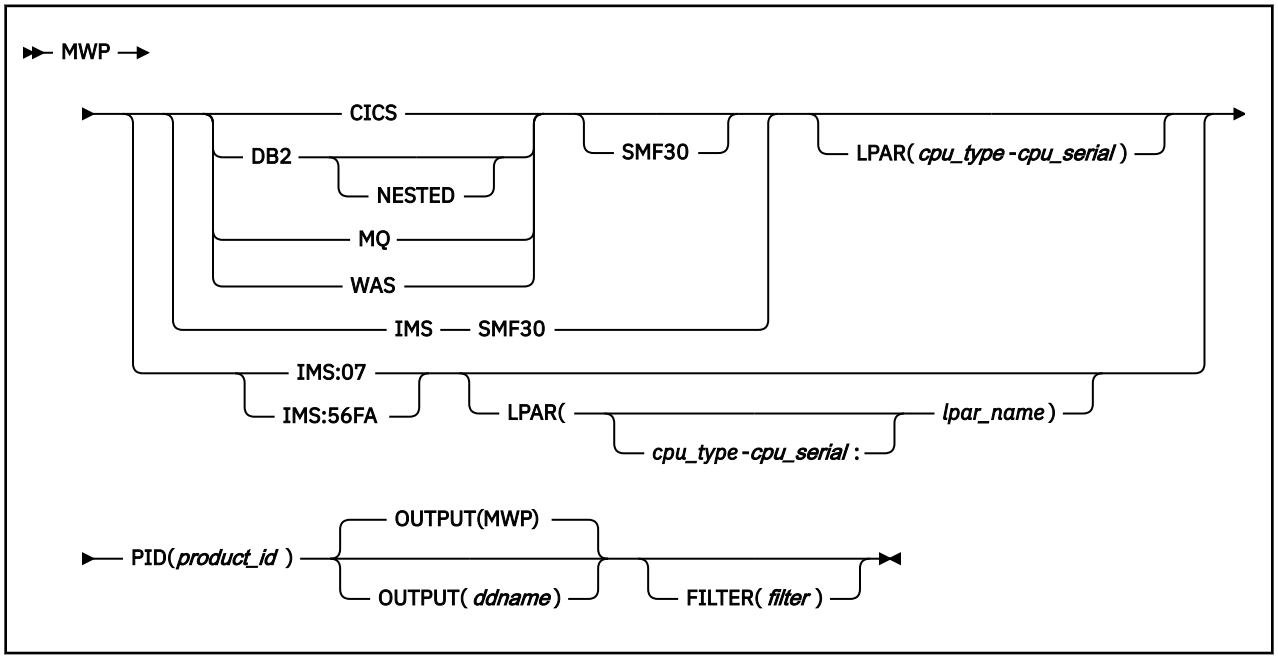
Each **MWP** command creates a single CSV file for one type of subsystem.

To filter the input log records that are processed by this command, either use the **FILTER** parameter to refer to a filter in the control repository, or specify filter conditions in **CODE** commands and related **COND** statements immediately after this command.

Filter conditions for the **MWP** command are also known as *mobile workload tags*.

The CPU time reported in CSV files created by the **MWP** command is central processor (CP) time only. Specialty processor (SP; zIIP and zAAP) time is not included. CPU time is reported in hourly intervals. For more details of the contents of CSV files created by the **MWP** command, see the MWRT documentation.

Format



Parameters

The **MWP** command creates a CSV file for a specific type of subsystem, referred to in the MWRT documentation as a *product family*.

After the parameter that identifies the product family, you can optionally specify the **SMF30** parameter.

CICS, DB2, MQ, WAS, IMS, IMS:07, IMS:56FA
Identifies the subsystem (or "product family").

SMF30
Calculates CPU time based on address space-level accounting, using SMF 30 records. The default behavior, if you do not specify **SMF30**, is to calculate CPU time based on transaction-level accounting.

NESTED
DB2 transaction-level accounting only. Only CPU time that is attributable to a nested or WLM task is reported.

Using **NESTED** avoids double-accounting of CPU time in DB2.

The **NESTED** parameter applies only if you specify the **DB2** parameter without the **SMF30** parameter. If you specify **SMF30**, **NESTED** is ignored.

LPAR
Identifies the logical partition on which the subsystem was running.

For **MWP** commands that read IMS log records (**MWP** commands that specify the **IMS:07** or **IMS:56FA** parameter):

- If you specify the **LPAR** parameter, the value must be the CPU type and CPU serial number separated by a hyphen. For example, `2827-0F4D7`. When writing the LPAR name to the CSV file, the **MWP** command appends a colon followed by the system identifier (SID) from the SMF record header. for example, `2827-0F4D7:SYSA`.
- If you omit the **LPAR** parameter, the LPAR specified in the CSV file is `IMS`.

For all other **MWP** commands (**MWP** commands that read SMF records):

- If you specify the **LPAR** parameter, the value must be either of the following:
  - The LPAR name. For example, SYSA.
  - The CPU type and CPU serial number separated by a hyphen, followed by a colon, and then the LPAR name. For example, `2827-0F4D7:SYSA`.
- If you omit the **LPAR** parameter, the LPAR specified in the CSV file is the system identifier (SID) from the SMF record header. For example, SYSA.

PID
The product identifier of the specified subsystem for this **MWP** command.

The **MWP** command writes the value that you specify to the CSV file. The **MWP** command does not check that this value corresponds to the specified subsystem.

If you do not know the product identifier, contact your z/OS system administrator.

*Table 30. **MWP** command: product identifiers for each subsystem*

Parameter of the MWP command that identifies the product family	Product family	Product identifier format	Version-specific example product identifiers
CICS	CICS Transaction Server for z/OS	5655-*xxx*	V5.3: 5655-Y04
DB2	DB2 for z/OS	5615-DB2	Same for all versions
MQ	IBM MQ for z/OS	5655-*xxx*	V9: 5655-R36
WAS	WebSphere Application Server for z/OS	5655-*xxx*	V8.5: 5655-W65
IMS, **IMS:07**, **IMS:56FA**	IMS	5635-*xxx*	V13: 5635-A04

OUTPUT
Identifies the ddname that specifies where to save the CSV file. The default ddname is MWP.

FILTER

Refers to the name of a filter that selects the log records to process.

The filter must be defined in the control repository specified by the FUWCDS ddname.

The filter must select the log type and code of the input log records required by this **MWP** command, otherwise no records will be processed.

If the filter refers to forms, the forms are ignored.

Example

The following JCL creates two files for input to MWRT, both based on records in the same input SMF file:

- A CSV file, based on CMF records, that reports CPU times for CICS transaction IDs that match the pattern "MOB*"
- A corresponding SMF file for input to MWRT

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=MVS1.SMF.WEEKLY
//MWP      DD  DSN=MWP.CICS.CSV,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//EXTRACT  DD  DSN=MWP.SMF,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *
MWP CICS PID(5655-Y04)
CODE(CMF)
COND TRAN EQ 'MOB*'

EXTRACT EXTERNAL
CODE(SMF:70.1)
CODE(SMF:89.1)
CODE(SMF:89.2)
/*
```

Related concepts

Mobile Workload Pricing

Mobile Workload Pricing (MWP) for z/OS is a pricing model that reduces the cost of running mobile workloads on z/OS. To take advantage of MWP, you use the Mobile Workload Reporting Tool (MWRT) to submit monthly reports to IBM. MWRT requires two types of input files: CSV and SMF. You can use Transaction Analysis Workbench to create these files.

Related reference

CSV and JSON commands

The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

EXTRACT command

Writes the selected log records to an extract data set.

Example: Mobile Workload Pricing using address space-level accounting, for all subsystems

This example JCL uses SMF 30 records to generate CSV files for the Mobile Workload Reporting Tool (MWRT), for all supported subsystems. The mobile workload tags, represented by **COND** statements, select the job names that are known to run only mobile workloads.

Example: Mobile Workload Pricing using transaction-level accounting from SMF records

This example JCL uses SMF records from various subsystems to generate CSV files for the Mobile Workload Reporting Tool (MWRT). The tags, represented by **COND** statements, select transactions that match specific criteria.

Example: Mobile Workload Pricing for IMS using transaction-level accounting

This example JCL uses IMS log records to generate a CSV file for the Mobile Workload Reporting Tool (MWRT).

REPORT command

Requests a report.

A SYSIN data set can contain zero or more **REPORT** commands.

By default, a report stops after 10,000 pages are written. To set a different page limit, specify a **PAGELIM** command at the start of the SYSIN data set.

The **REPORT** command requests one of the following types of report:

Formatted record reports
These reports reproduce how the ISPF dialog log browser displays log records.

CICS-DBCTL reports
These reports process CMF records, IMS log records, or both, to help diagnose problems with CICS-DBCTL transactions.

DB2 accounting exception reports
These reports list the DB2 accounting (SMF type 101) records that match your exception criteria. Exception criteria specify threshold values for fields such as CPU time. You can also request an extract that contains these records.

OPERLOG reports
These reports list selected records from OPERLOG, the z/OS operations log (log stream).

SMF reports (including the SMF Recap report)
These reports process SMF files.

The SMF Recap report provides a high-level overview of the contents of an SMF file, such as which systems and subsystems wrote records to that file, and which record types the file contains.

The layout of other SMF reports is specific to a particular SMF record type and, if applicable, subtype.

These reports are available only for some combinations of SMF record type and subtype.

Rather than showing all fields from a record, SMF reports show fields that are relevant to analyzing transactions. In some cases, the reports show derived values: information calculated from the values of several fields. The layout of SMF reports is fixed: you cannot tailor them by specifying a form or a **FORMAT** command.

Related concepts

CICS-DBCTL transactions
You can analyze response time problems in CICS-DBCTL transactions using CICS monitoring facility (CMF) performance class records (SMF type 110 records), IMS log records, or a combined view of both.

Related tasks

Creating reports
You can use the Transaction Analysis Workbench ISPF dialog to generate JCL for the following types of batch report: CICS-DBCTL, SMF, DB2, and OPERLOG reports using the Transaction Analysis Workbench report and extract utility; CICS reports using CICS Performance Analyzer; and IMS reports using IMS Performance Analyzer.

Using forms to exclude fields when creating formatted record reports
When using the report and extract utility to create a formatted record report, you can optionally use forms to exclude fields. Formatted record reports reproduce the formatting that the ISPF dialog displays when you browse an individual record in the log browser.

Related information

FUW0055W

CICS/IMS-DBCTL request parameter is invalid; PARM(*invalid parameter*

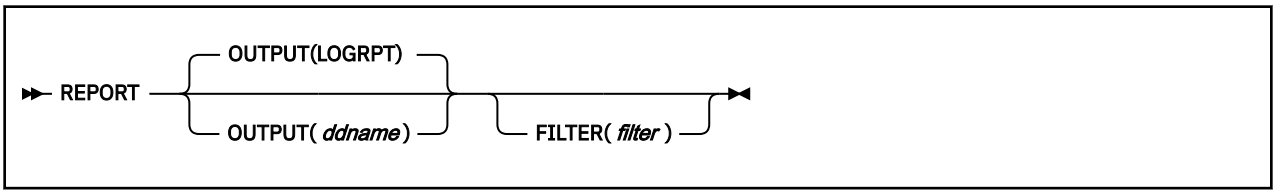
REPORT command for formatted record reports

Requests a formatted record report. Formatted record reports reproduce how the ISPF dialog log browser displays log records.

By default, these reports have the same format as a record that you select in the log browser, and then display using the FORM format, using forms in the current filter to determine which fields to show; if the filter does not refer to forms, all fields are shown. This default behavior is equivalent to specifying a `FORMAT(FORM)` command after the **REPORT** command.

To specify a different type of record formatting, or to request additional reports in other formats, include a **FORMAT** command after the **REPORT** command.

Format



Parameters

ddname
The ddname of the data set where you want to write the report. If you do not specify a ddname, the report is written to the default ddname LOGRPT.

filter
The name of a filter that selects the log records you want to report.

The filter is optional, but is recommended to reduce the size of the report. If you do not specify a filter, all log records are selected.

If the filter refers to forms, then formatted record reports requested by the `FORMAT(FORM)` command contain only the fields specified in the forms.

The filter must be defined in the control repository specified by the FUWCDS ddname.

Related tasks

Creating formatted record reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports that reproduce the formatting that the ISPF dialog displays when you browse an individual log record. These batch reports are known as *formatted record reports*.

Related reference

FORMAT command
Specifies options for presenting data in formatted record reports, matching the options that you can specify when browsing records in the ISPF dialog. Ignored for extracts, CSV, and JSON output.

REPORT CICS-DBCTL command: CICS reports

Requests a list or summary report that uses CMF records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

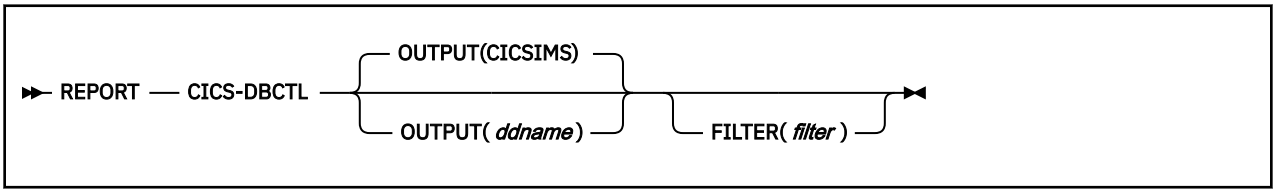
A **REPORT CICS-DBCTL** command that is *not* followed by `PARM(COMBINED)` processes a single input file containing CMF records (ddname SMFIN) to create one or more of the following outputs:

- A CICS-DBCTL list report
- A CICS-DBCTL summary report

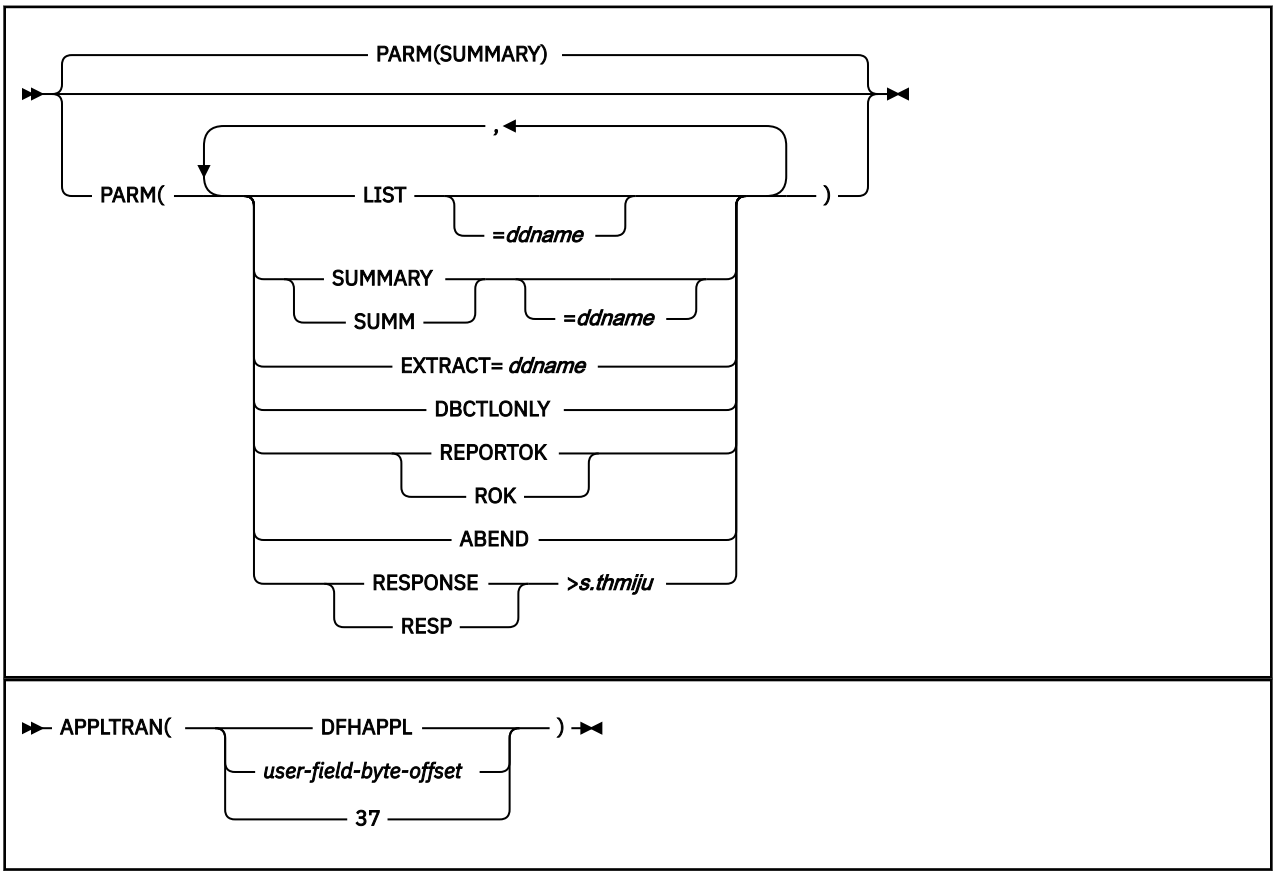
- An extract that you can then sort by CICS transaction ID. You can use the sorted extract, known as a CICS transaction index, as input to a later **REPORT CICS-DBCTL** command.

Typically, the input file is an SMF file containing many types of SMF record, but you can also use extracts created by an earlier **REPORT CICS-DBCTL** command.

Format



You can optionally follow the **REPORT CICS-DBCTL** command with a **PARM** command and an **APPLTRAN** command.



REPORT CICS-DBCTL command parameters

OUTPUT(*ddname*)
The ddname of the data set where you want to write reports. If you do not specify a ddname, reports are written to the default ddname CICSIMS.

You can override this value using the **LIST**, **SUMMARY**, and **COMBINED** parameters of the **PARM** command.

filter
The name of a filter that selects the CMF records you want to report or extract. If the filter refers to forms, the forms are ignored. The filter must be defined in the control repository specified by the FUWCDS ddname.

Recommendation: Instead of using a filter to select CMF records, use exception criteria (specified by the **ABEND** and **RESPONSE** parameters of the **PARM** command). **REPORT CICS-DBCTL** command processing is optimized to use exception criteria to filter out transactions that do not require investigation. Exception criteria, if specified, will still be applied to records selected by a filter.

The filter must specify code CMF 6E13 to include the CMF performance class records for reporting and extract, and can specify conditions. You can also use the **CODE** and **COND** commands following the **REPORT CICS-DBCTL** command to specify the filter. For example, the following filter selects only transactions with IDs that start with MY and that took longer than 2 seconds to process DL/I requests.

```
REPORT CICS-DBCTL PARM(...
CODE(CMF,6E13)
COND TRAN    EQ 'MY*'
COND IMSWAIT GT  2.0
```

PARM command parameters

The **PARM** command specifies options for the **REPORT CICS-DBCTL** command.

If you omit the **PARM** command, the **REPORT CICS-DBCTL** command creates a summary report only.

LIST
Create a CICS-DBCTL list report that shows details of each transaction.

SUMMARY
SUMM
Create a CICS-DBCTL summary report that summarizes transactions by transaction ID and APPLID.

SUMM is an alias for **SUMMARY**.

LIST=*ddname*, SUMMARY=*ddname*
By default, the summary report and the list report are written to the same output file, specified by the **OUTPUT** parameter of the **REPORT CICS-DBCTL** command.

To avoid the inconvenience of locating the summary report at the end of the list report, specify separate output files. For example, `LIST=LIST1,SUMMARY=SUMM1`.

EXTRACT=*ddname*
Copy exceptions (transactions that match the exception criteria) from the input file to the extract file specified by *ddname*. If no exception criteria are specified, copy all transactions to the extract file.

DBCTLONLY
Only report or extract CICS transactions that issued DL/I requests. Ignore all other transactions.

REPORTOK
ROK
Report all transactions, regardless of whether or not they are exceptions. This parameter does not affect extract processing.

ROK is an alias for **REPORTOK**.

Parameters that specify exception criteria:

ABEND
A transaction is an exception if it abends.

RESPONSE
RESP
A transaction is an exception if it exceeds this response time threshold. Allowed values are 0.000001 to 999999 seconds.

RESP is an alias for **RESPONSE**.

APPLTRAN command parameters

The **APPLTRAN** command replaces the actual CICS transaction ID in the input CMF records with an application (or umbrella) transaction ID. If you specify an **APPLTRAN** command, reports will refer to the application transaction ID, and the TRAN field value in extract records will be the application transaction ID rather than the TRAN field value (the "actual" CICS transaction ID) in the original CMF records.

You can use the application transaction ID specified by a field in a CMF record: either DFHAPPL, or a user field written at a user-defined event monitoring point (EMP) such as the Tivoli OMEGAMON XE for CICS OMEGBSC ("Basic") EMP.

Reporting an application transaction ID is useful in situations where different applications use the same actual CICS transaction ID, and you want to identify which instances of the transaction belong to each application.

The application transaction ID will only be reported when both of the following conditions are true:

- The application transaction ID has a non-blank value.
- The CMF record contains DFHAPPL or user fields.

Otherwise, the actual CICS transaction ID will be reported.

APPLTRAN(*user-field-byte-offset*)
Use the application transaction ID located in the CMF user fields (written by EMPs) at the specified offset. The offset must be in the range 1 to 999, where 1 is the first character of the first user field.

The offset must be from the start of the first user field, regardless of whether the ID is located in a subsequent user field. In the following example, where the transaction ID is located in the third user field, you would need to specify APPLTRAN(19):

```
User EMP1 length=8  value='xxxxxxxx'
User EMP2 length=6  value='xxxxxx'
User EMP3 length=12 value='xxxxTRANxxxx'
```

If there are no user fields recorded by CMF or the name is blank, then the actual CICS transaction name is retained and reported.

APPLTRAN(37)
Use the umbrella transaction ID defined by the Tivoli OMEGAMON XE for CICS OMEGBSC ("Basic") EMP in the MCT:

```
DFHMCT TYPE=EMP,CLASS=PERFORM,
       ID=(OMEGBSC.1),FIELD=(1,OMEGBSC),
       PERFORM=(MOVE(0,132))
```

If OMEGBSC is not the first user EMP, then you will need to adjust the offset by adding the length of all preceding user fields.

DD statements

The following DD statements are specific to the **REPORT CICS-DBCTL** command:

//EXTRACT1 DD
Required only if you request an extract. To create an extract, you must specify EXTRACT=*ddname*, where *ddname* refers to a DD statement that specified the extract output file. EXTRACT1 is an example ddname; there is no default value.

//SORTCTIM
Specifies the output file to contain a **SORT** statement that you can use in a subsequent job step to sort the extract into transaction start time sequence. The sorted extract is known as a CICS transaction index. A CICS transaction index is required for combined CICS and IMS reporting.

Example: Default summary report

The following example JCL requests a CICS-DBCTL summary report:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW     EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=CICS.SMF
//SYSIN    DD  *
REPORT CICS-DBCTL
/*
```

The summary report is written to the default ddname CICSIMS. The report contains information for all CMF performance class records in the input SMF file (CIMS.SMF), including transactions that did not issue any DL/I requests.

Example: list and summary reports of exceptions from yesterday

The following example JCL requests CICS-DBCTL list and summary reports:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW     EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=CICS.SMF
//SYSIN    DD  *
START -1-08.00.00.00
STOP  -1-17.00.00.00
REPORT CICS-DBCTL
PARM(LIST=RPTLIST,SUMM=RPTSUMM,DBCTLONLY,ABEND,RESPONSE>2.0)
APPLTRAN(37)
LINECNT(0)
/*
```

//SMFIN DD
Specifies the input SMF file containing CMF records, CICS.SMF.

START and STOP
Specify the report interval as yesterday, between 8 a.m. and 5 p.m.

LIST=RPTLIST
Requests a CICS-DBCTL list report, to be written to the ddname RPTLIST.

SUMM=RPTSUMM
Requests a CICS-DBCTL summary report, to be written to the ddname RPTSUMM.

DBCTLONLY
Restricts the reports to transactions that issued DL/I requests.

ABEND,RESPONSE>2.0
Restricts the reports to transactions that match these exception criteria: transactions that abended or transactions that had a response time greater than 2 seconds.

APPLTRAN(37)
Reports the umbrella CICS transaction IDs specified by the Tivoli OMEGAMON XE for CICS OMEGBSC user EMP, rather than the original CICS transaction IDs.

Related tasks

Creating CICS-DBCTL reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

Related reference

CICS-DBCTL report JCL
To create CICS-DBCTL reports, use the **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands of the report and extract utility. The multi-step JCL presented here generates all CICS-DBCTL reports.

CICS-DBCTL list report

The CICS-DBCTL list report uses data in CMF performance class (SMF type 110) records to show the performance characteristics of individual instances of CICS transactions, with a focus on DBCTL.

CICS-DBCTL summary report
The CICS-DBCTL summary report uses data in CMF performance class (SMF type 110) records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

CICS-DBCTL combined summary report
The CICS-DBCTL combined summary report combines data from CMF performance class (SMF type 110) records with data from IMS log records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

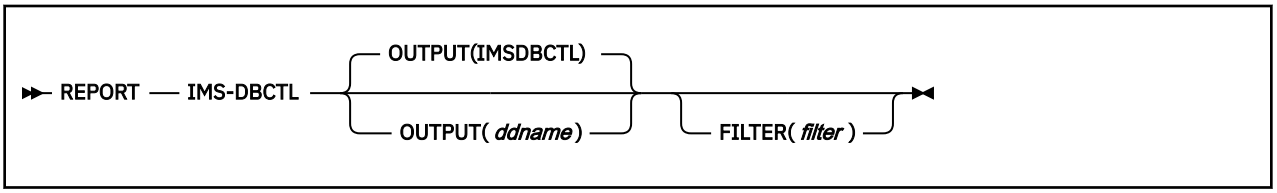
Related information

FUW0243E
APPLTRAN must specify (DFHAPPL or 1-999)

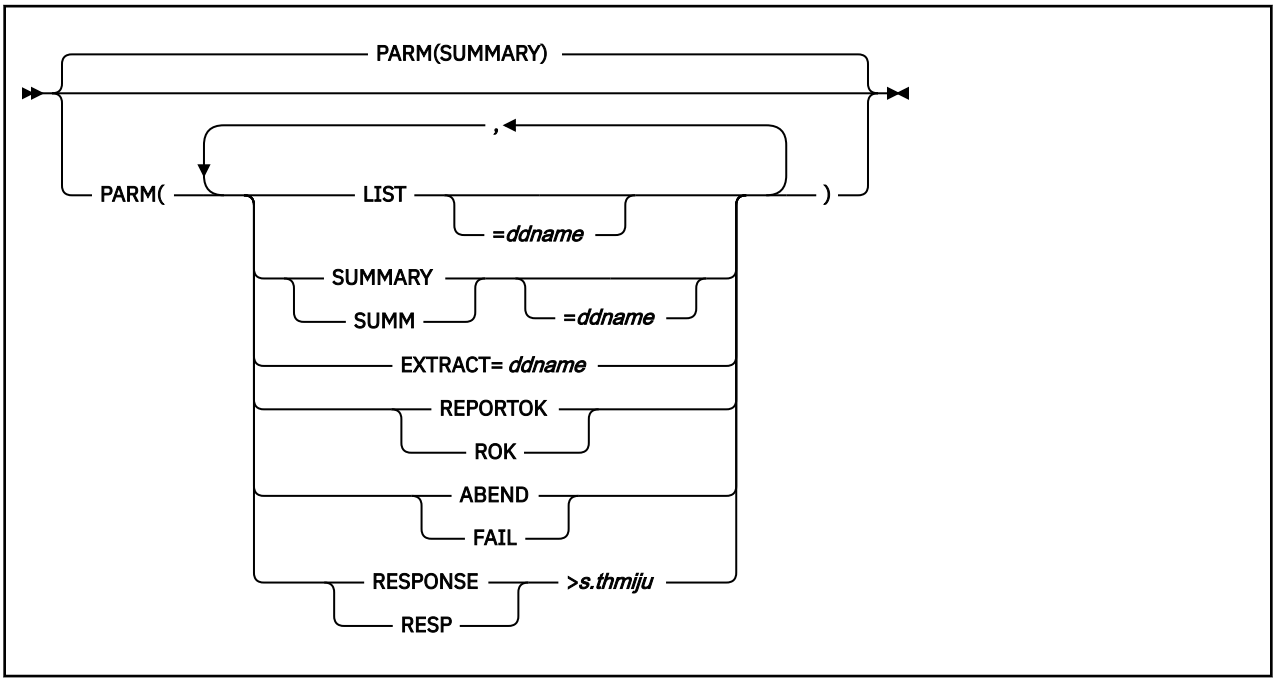
REPORT IMS-DBCTL command: IMS reporting

Requests a list or summary report that uses IMS log records to show response time problems in CICS-DBCTL transactions, or creates an extract that can be used for combined CICS and IMS reporting.

Format



You can optionally follow the **REPORT IMS-DBCTL** command with a **PARM** command.



REPORT IMS-DBCTL command parameters

OUTPUT(*ddname*)
The ddname of the data set where you want to write reports. If you do not specify a ddname, reports are written to the default ddname IMSDBCTL.

You can override this value using the **LIST** and **SUMMARY** parameters of the **PARM** command.

filter
The name of a filter that selects the IMS log records you want to report or extract. If the filter refers to forms, the forms are ignored. The filter must be defined in the control repository specified by the FUWCDS ddname.

Recommendation: Instead of using a filter to select IMS log records, use exception criteria (specified by the **ABEND** and **RESPONSE** parameters of the **PARM** command). **REPORT IMS-DBCTL** command processing is optimized to use exception criteria to filter out transactions that do not require investigation. By specifying a filter, you might exclude record types that are crucial to creating a complete index record. Exception criteria, if specified, will still be applied to records selected by a filter.

PARM command parameters

The **PARM** command specifies options for the **REPORT IMS-DBCTL** command.

If you omit the **PARM** command, the **REPORT IMS-DBCTL** command creates a summary report only.

LIST
Create an IMS-DBCTL list report that shows details of each transaction.

SUMMARY
SUMM
Create an IMS-DBCTL summary report that summarizes transactions by CICS transaction ID, IMS PSB name, and CICS generic APPLID.

SUMM is an alias for **SUMMARY**.

LIST=*ddname*, SUMMARY=*ddname*
By default, the summary report and the list report are written to the same output file, specified by the **OUTPUT** parameter of the **REPORT IMS-DBCTL** command.

To avoid the inconvenience of locating the summary report at the end of the list report, specify separate output files. For example, `LIST=LIST1,SUMMARY=SUMM1`.

EXTRACT=*ddname*
Copy exceptions (transactions that match the exception criteria) from the input file to the extract file specified by *ddname*. If no exception criteria are specified, copy all transactions to the extract file.

REPORTOK
Report all transactions, regardless of whether or not they are exceptions. This parameter does not affect extract processing.

Parameters that specify exception criteria:

ABEND
A transaction is an exception if it abends.

FAIL
A transaction is an exception if it abends or generates a Fast Path failure (type 5938 log record). If you specify **FAIL**, you do not need to specify **ABEND**.

RESPONSE
RESP
A transaction is an exception if it exceeds this response time threshold. Allowed values are 0.000001 to 999999 seconds.

RESP is an alias for **RESPONSE**.

DD statements

The following DD statements are specific to the **REPORT IMS-DBCTL** command:

//EXTRACT2 DD
Required only if you request an extract. To create an extract, you must specify EXTRACT=*ddname*, where *ddname* refers to a DD statement that specified the extract output file. EXTRACT2 is an example ddname; there is no default value.

//SORTITIM
Specifies the output file to contain a **SORT** statement that you can use in a subsequent job step to sort the extract into transaction start time sequence. The sorted extract is known as an IMS transaction index. An IMS transaction index is required for combined CICS and IMS reporting.

Example: Default summary report

The following example JCL requests an IMS-DBCTL summary report:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW     EXEC PGM=FUWBATCH,PARM=V121
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=CICS.SMF
//SYSIN    DD  *
REPORT IMS-DBCTL
/*
```

The summary report is written to the default ddname IMSDBCTL.

Example: list and summary reports of exceptions from yesterday

The following example JCL requests IMS-DBCTL list and summary reports:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUW     EXEC PGM=FUWBATCH,PARM=V121
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//LOGIN    DD  DISP=SHR,DSN=IMSA.SLDS
//SYSIN    DD  *
START -1-08.00.00.00
STOP  -1-17.00.00.00
REPORT IMS-DBCTL
PARM(LIST=RPTLIST,SUMM=RPTSUMM,FAIL,RESPONSE>0.25)
LINECNT(0)
/*
```

//LOGIN DD
Specifies the input IMS log, IMSA.SLDS.

START and STOP
Specify the report interval as yesterday, between 8 a.m. and 5 p.m.

LIST=RPTLIST
Requests an IMS-DBCTL list report, to be written to the ddname RPTLIST.

SUMM=RPTSUMM
Requests an IMS-DBCTL summary report, to be written to the ddname RPTSUMM.

FAIL,RESPONSE>0.25
Restricts the reports to transactions that match these exception criteria: transactions that failed or transactions that had a response time greater than 0.25 seconds.

Related tasks

Creating CICS-DBCTL reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

Related reference

CICS-DBCTL report JCL
To create CICS-DBCTL reports, use the **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands of the report and extract utility. The multi-step JCL presented here generates all CICS-DBCTL reports.

IMS-DBCTL list report

The IMS-DBCTL list report uses data in IMS log records to show the performance characteristics of individual instances of DBCTL threads started by CICS transactions.

IMS-DBCTL summary report
The IMS-DBCTL summary report uses data in IMS log records to show the performance characteristics of DBCTL threads started by, summarized by CICS transaction ID, program, and APPLID.

CICS-DBCTL combined summary report
The CICS-DBCTL combined summary report combines data from CMF performance class (SMF type 110) records with data from IMS log records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

REPORT CICS-DBCTL command: combined CICS and IMS reporting

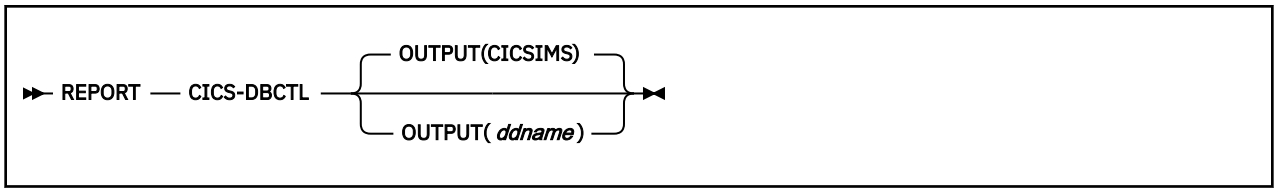
Requests a report that combines output from earlier **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands to show CICS DBCTL transaction response times across CICS and IMS.

A **REPORT CICS-DBCTL** command that is followed by PARM(COMBINED) processes the following two input files to create a CICS-DBCTL combined summary report:

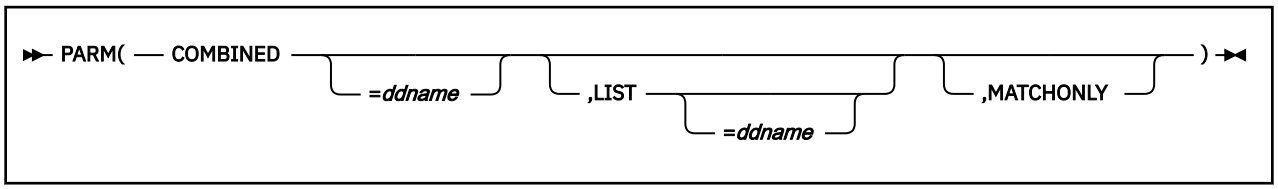
CICS transaction index
A copy of the extract created by an earlier **REPORT CICS-DBCTL** command, sorted by CICS transaction start time (ddname SMFIN).

IMS transaction index
A copy of the extract created by an earlier **REPORT IMS-DBCTL** command, sorted by IMS transaction start time (ddname LOGIN).

Format



To request a CICS-DBCTL combined summary report, you must follow the **REPORT CICS-DBCTL** command with a **PARM** command that specifies the **COMBINED** parameter.



REPORT CICS-DBCTL command parameters

OUTPUT(*ddname*)
The ddname of the data set where you want to write reports. If you do not specify a ddname, reports are written to the default ddname CICSIMS.

You can override this value using the **COMBINED** parameter of the **PARM** command.

PARM command parameters

COMBINED
Create a CICS-DBCTL combined summary report that summarizes transactions by transaction ID and APPLID.

COMBINED=*ddname*
By default, the combined report is written to the output file specified by the **OUTPUT** parameter of the **REPORT CICS-DBCTL** command.

You can also request a CICS-DBCTL list report with the combined report. To avoid the inconvenience of locating the combined report at the end of the list report, specify separate output files. For example, `LIST=LIST1,COMBINED=COMB1`.

MATCHONLY
Only report CICS transactions that match an IMS transaction index record.

MATCHONLY ensures that only transactions with both CICS and IMS metrics are reported; that the CICS and IMS report sections are for the same transactions. This ensures that the report is not distorted by CICS transactions that did not have an associated IMS exception.

If you do not specify **MATCHONLY**, then all CICS transactions are reported whether or not they match an IMS transaction index record; this can cause the number of CICS transactions reported to be higher than the number of IMS threads reported.

Example: Using existing CICS and IMS transaction indexes

The following JCL uses existing CICS and IMS transaction indexes from earlier **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands.

```
//FUWBATCH JOB ,CLASS=A,NOTIFY=&SYSUID
//*
//S1      EXEC PGM=FUWBATCH
//STEPLIB  DD  DISP=SHR,DSN=FUW.SFUWLINK
//SYSPRINT DD  SYSOUT=*
//SMFIN    DD  DISP=SHR,DSN=CICS.CMF.EXCEPT.EXTRACT
//LOGIN    DD  DISP=SHR,DSN=IMS.DBCTL.EXCEPT.EXTRACT
//SYSIN    DD  *
REPORT CICS-DBCTL OUTPUT(CICSIMS)
PARM(COMBINED,MATCHONLY)
LINECNT(0)
/*
```

Related tasks

Creating CICS-DBCTL reports
You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of CICS-DBCTL transactions.

Related reference

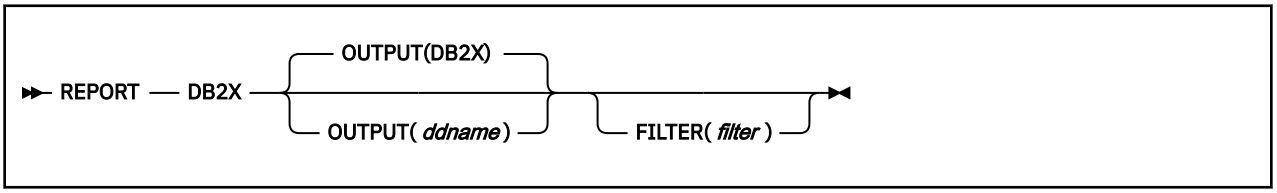
CICS-DBCTL report JCL
To create CICS-DBCTL reports, use the **REPORT CICS-DBCTL** and **REPORT IMS-DBCTL** commands of the report and extract utility. The multi-step JCL presented here generates all CICS-DBCTL reports.

CICS-DBCTL combined summary report
The CICS-DBCTL combined summary report combines data from CMF performance class (SMF type 110) records with data from IMS log records to show the performance characteristics of CICS transactions, summarized by CICS transaction ID and APPLID, with a focus on DBCTL.

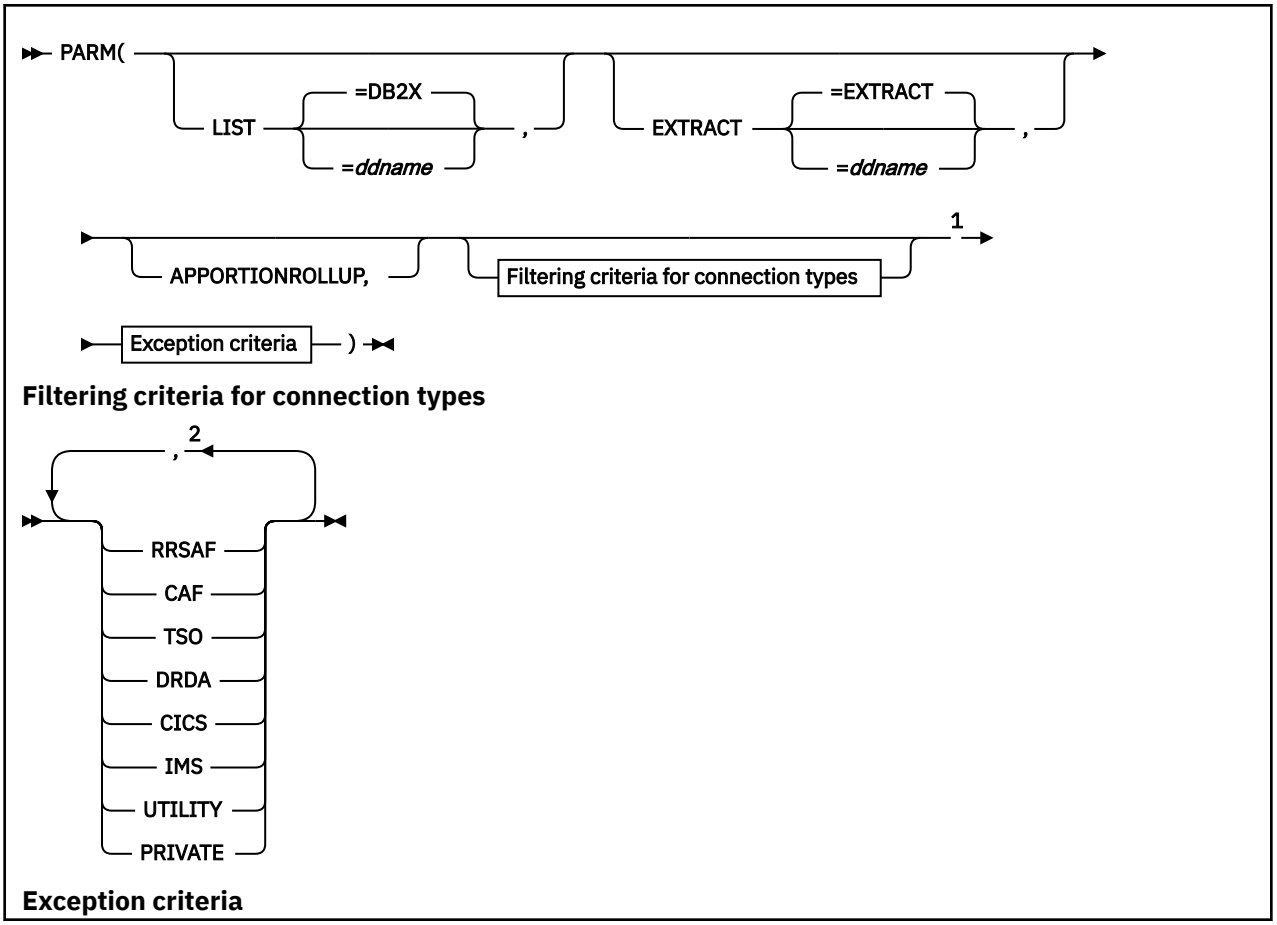
REPORT DB2X command: DB2 accounting exception report and extract

Requests a report or an extract, or both, of DB2 accounting (SMF type 101) records that match the specified exception criteria.

Format

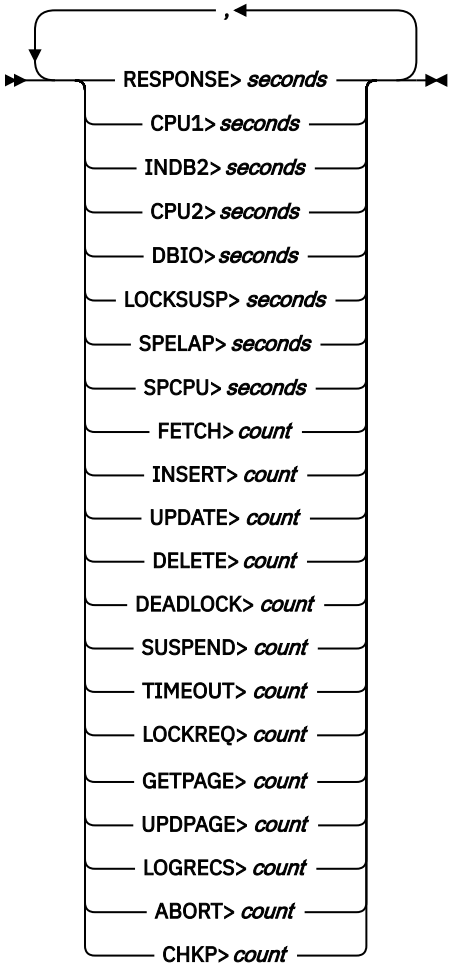


The **REPORT DB2X** command must be followed by one or more **PARM** commands that specify exception criteria, among other options.



Filtering criteria for connection types

Exception criteria



Notes:

[1] If you omit filtering criteria for connection types, records for all connection types are selected.
[2] **RRFSAF** and **CAF** are synonyms. **UTILITY** and **PRIVATE** are not generated by the ISPF dialog.

REPORT DB2X command parameters

OUTPUT(*ddname*)
The ddname of the data set where you want to write the report. If you do not specify a ddname, the report is written to the default ddname DB2X.

You can override this value using the **LIST** parameter of the **PARM** command.

filter
The name of a filter that selects the DB2 accounting records you want to report or extract. If the filter refers to forms, the forms are ignored. The filter must be defined in the control repository specified by the FUWCDS ddname.

Recommendation: Where possible, instead of using a filter to select records, use exception criteria (specified by the **PARM** command). **REPORT DB2X** command processing is optimized to use exception criteria to select records.

PARM command parameters

The **PARM** command specifies options for the **REPORT DB2X** command.

LIST
Create a report of the exceptions: the records that match the exception criteria.

EXTRACT
Copy exceptions from the input file to the extract file specified by *ddname*.

APPORTIONROLLUP
Divide accounting values by the rollup count (number of threads accumulated) in the accounting record before comparison with exception criteria threshold values. Apportion the accounting values equally among all of the threads.

Response (thread elapsed) time is treated differently because it is recorded for each thread being rolled up; the value for each thread is checked individually.

If you omit exception criteria, there will be no exceptions; the report and extract will be empty.

Related reference

DB2 accounting exception reports and extracts
DB2 accounting exception reports and extracts are generated from DB2 accounting (SMF type 101) records. DB2 accounting records from the specified input files are checked against optional *filtering criteria* for the DB2 subsystem ID (SSID), plan, and connection type. Records that match the filtering criteria are checked against *exception criteria* that specify thresholds for various performance-related values, such as CPU time.

DB2 accounting exception report and extract JCL
To create DB2 accounting exception reports and extracts, use the **REPORT DB2X** command of the report and extract utility.

DB2 accounting exception criteria
Exception criteria specify thresholds for various performance-related accounting values, such as CPU time. Only records that trigger an exception (match the exception criteria) are included in the DB2 accounting exception extracts and reports.

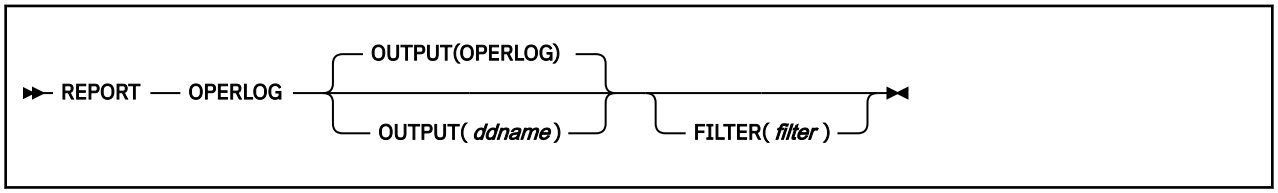
REPORT command for OPERLOG reports

Prints selected records from OPERLOG, the z/OS operations log (log stream).

The **REPORT OPERLOG** command must be preceded by the following command:

```
LOGSTREAM OPERLOG:SYSPLEX.OPERLOG
```

Format



Parameters

ddname
The ddname of the data set where you want to write the report. If you do not specify a ddname, the report is written to the default ddname OPERLOG.

filter
The name of a filter that selects the log records you want to report.

The filter is optional, but is recommended to reduce the size of the report. If you do not specify a filter, all log records are selected.

If the filter refers to forms, the forms are ignored.

The filter must be defined in the control repository specified by the FUWCDS ddname.

Related tasks

Creating OPERLOG reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch printouts of OPERLOG, the z/OS operations log (log stream).

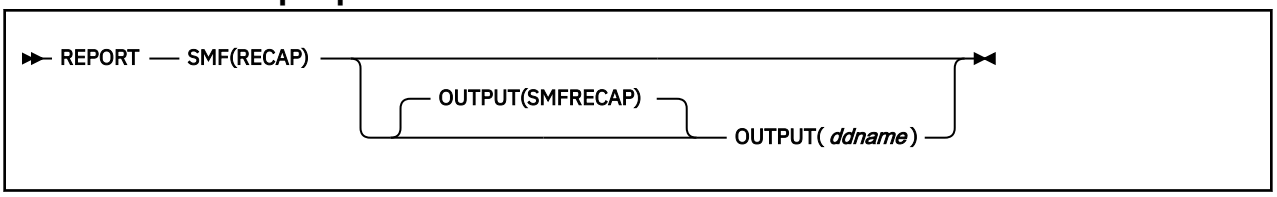
Related reference

LOGSTREAM command
Specifies that the input log file is a log stream rather than a data set.

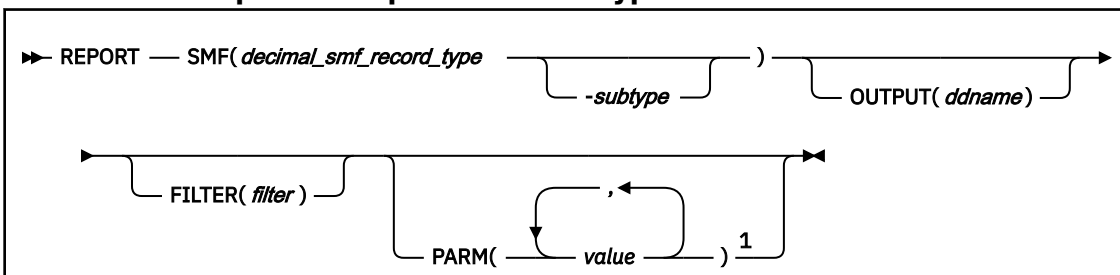
REPORT command for SMF reports

Requests either an SMF Recap report that gives an overview of the contents of an SMF file or, for some SMF record types only, a report that processes a particular type of SMF record; these reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

Format: SMF Recap report



Format: SMF reports for specific record types



Notes:

[1] The **PARM** parameter is relevant only to SMF(116-1) reports.

Parameters

decimal_smf_record_type-subtype
A decimal SMF record type and, if applicable, subtype. For a list of supported values, see the table in the description of *ddname*.

ddname
The ddname of the data set where you want to write the report. If you do not specify a ddname, the report is written to the default ddname shown in the following table.

Table 31. Default output ddnames for SMF reports

REPORT command parameter	Default output ddname	Report description
SMF(30)	JOBSTATS	“SMF type 30: Address Space Activity report” on page 435
SMF(33-2)	APPCCONV	“SMF type 33-2: APPC/MVS Conversation List report” on page 437
SMF(42-6)	DASDSTAT	“SMF type 42-6: DASD Data Set I/O report” on page 438
SMF(64)	VSAMSTAT	“SMF type 64: VSAM Data Set report” on page 442

Table 31. Default output ddnames for SMF reports (continued)		
REPORT command parameter	**Default output ddname**	**Report description**
SMF(70-1)	RMFPROC	"SMF type 70-1: RMF Processor Activity report" on page 443
SMF(75)	RMFPGDS	"SMF type 75: RMF Page Data Set Activity report" on page 445
SMF(78-2)	RMFVSUSE	"SMF type 78-2: RMF Virtual Storage Activity report" on page 447
SMF(79-15)	IRLMLOCK	"SMF type 79-15: IRLM Long Lock Detection report" on page 449
SMF(88-1)	LOGGER	"SMF type 88-1: System Logger Log Stream Summary report" on page 450
SMF(101)	DB2ACCT	"SMF type 101: DB2 Thread Accounting Summary report" on page 453
SMF(116-0)	MQACCT1	"SMF type 116-0: IBM MQ Accounting Class 1 reports" on page 456
SMF(116-1)	MQACCT3	"SMF type 116-1: IBM MQ Accounting Class 3 reports" on page 458
SMF(RECAP)	SMFRECAP	"SMF Recap report" on page 431

filter

The name of a filter that selects the log records you want to report.

The filter is optional, but is recommended to reduce the size of the report. If you do not specify a filter, all log records are selected.

If the filter refers to forms, the forms are ignored.

The filter must be defined in the control repository specified by the FUWCDS ddname.

Related tasks

Creating SMF reports

You can use the Transaction Analysis Workbench ISPF dialog to create batch reports of SMF records.

Related reference

SMF reports

Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

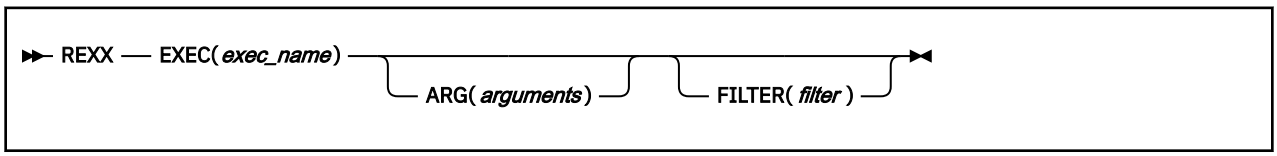
REXX command

Runs a REXX exec that enables you to read sequentially forward through the selected log records.

A SYSIN data set can contain only one **REXX** command. A SYSIN data set that contains a **REXX** command cannot contain any **REPORT**, **EXTRACT**, or **CSV** commands.

To filter the input log records that are processed by this command, either use the **FILTER** parameter to refer to a filter in the control repository, or specify filter conditions in **CODE** commands and related **COND** statements immediately after this command.

Format



Parameters

exec_name
The name of the member in the SYSEXEC library concatenation that contains the REXX exec.

arguments
An argument, or list of arguments, to pass to the REXX exec. To pass a list of arguments, separate the arguments with spaces and enclose the list in quotes. For example:

```
ARG('A1 A2 A3')
```

filter
The name of a filter that selects the log records you want to process. The filter must be defined in the control repository specified by the FUWCDS ddname.

A filter is optional, but is recommended to reduce the size of the extract data set. If you do not specify a filter, all log records are selected.

Related concepts
REXX API for formatting and analyzing logs
Transaction Analysis Workbench provides a REXX application programming interface (API) that you can use to format and analyze logs.

Qualifying commands

restrict or add conditions to an action command. Qualifying commands affect only the preceding action command in the SYSIN data set.

CODE command and COND statement

The **CODE** command filters log records by including or excluding them based on their log type and code. To refine the filter based on field values in a log record, follow the **CODE** command with one or more **COND** statements.

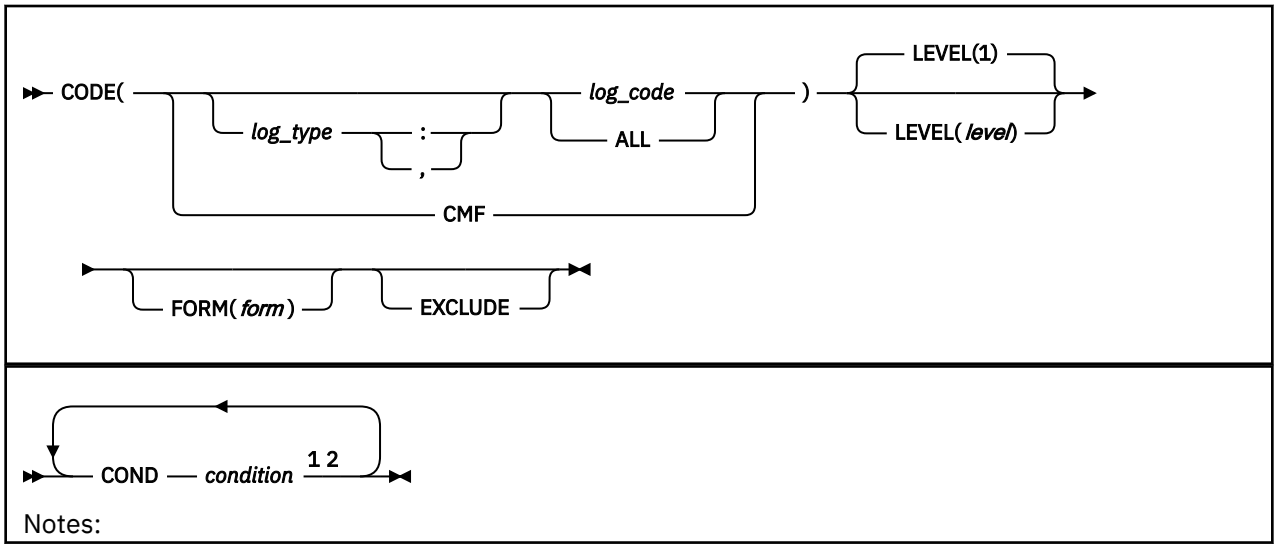
The **CODE** command applies only to the preceding action command in the SYSIN data set.

Each action command can be followed by one or more **CODE** commands.

Each **CODE** command can be followed by one or more **COND** statements. There can be no intervening commands between a **CODE** command and its **COND** statements.

The **FILTER** parameter on action commands and the **CODE** command are mutually exclusive; you cannot specify **CODE** commands and the **FILTER** parameter for the same action command.

Format



Notes:

[1] For the format of a condition, see "Filter conditions" on page 398.
[2] Conditions for `CODE(ALL)` or `CODE(log_type:ALL)` commands can use global field names only, not field names that are specific to a particular log code. **COND** statements for other **CODE** commands can use field names of the specified log code.

`CODE(CMF)` is equivalent to `CODE(CMF:6E13)`.

Parameters

CODE command parameters:

ALL
Selects all log records from all log types. If you do not specify any **CODE** commands, this is the default behavior.

log_type
Specifies one of the following log types supported by Transaction Analysis Workbench:

ATF
Tivoli OMEGAMON XE for IMS Application Trace Facility journal records

CMF
SMF records written by the CICS monitoring facility (CMF)

CON
IMS Connect event data, generated by IMS Connect Extensions

CQS
CQS Shared Queues log records

CTR
CICS trace entries

DB2
DB2 log records

DTR
DB2 trace records

IMS
IMS system and user log records

ITR
IMS trace table entries, extracted from IMS log record types 67FA and 67FF

MON
IMS monitor records

MVS
OPERLOG records, or records from log streams for which Transaction Analysis Workbench does not provide record formatting; these "other" log streams are indicated in Transaction Analysis Workbench by the prefix OTHER:

MQ
IBM MQ log extract records

SMF
SMF records, except for records that belong to the more specific log types CMF and DTR

VSR
CICS VSAM forward recovery and autojournaling codes

Specifying a log type is optional but recommended, and is particularly useful in the following cases:

- To avoid ambiguity, in cases where log types have overlapping log codes.

  For example, some DB2 and MQ records have the same log code, but they are different records. If you specify a log code without a log type, such as `CODE(0002)`, and your input includes both DB2

and MQ records, then the results will include records with this log code from both log types. To avoid ambiguity, specify the log type and the log code. For example, `CODE(DB2:0002)`.

- To select all log codes from a particular log type. For example, `CODE(DB2:ALL)`.

log_code
Specifies a log record type and, optionally, subtype, in one of the following formats:

- 2 or 4 hexadecimal digits. The first two digits identify the record type. The second two digits, if specified, identify the subtype.
- A decimal integer (1 - 255) followed by a hyphen (-) or a period (.), and then, optionally, another decimal integer (1 - 255). The first integer identifies the record type. The second integer, if specified, identifies the subtype. The hyphen or period indicates that the digits are decimal. The hyphen or period is required, whether or not you specify a subtype.
- DTR log type only: 3 decimal digits with leading zeros. For example, for DB2 accounting records: `CODE(DTR:003)`.

For example, the following **CODE** commands are equivalent:

```
CODE(SMF:78.2)
CODE(SMF:78-2)
CODE(SMF:4E02)
```

Only some log record types have subtypes. To select all subtypes for a record type, omit the subtype. When using the decimal format, remember to specify a hyphen after the record type. For example, the following **CODE** commands are equivalent:

```
CODE(SMF:78.)
CODE(SMF:78-)
CODE(SMF:4E)
```

level
For complex filters that involve multiple **CODE** commands. Specifies an integer, 1 - 255, that determines how to combine this **CODE** command with other **CODE** commands.

CODE commands with the same level are combined by a logical OR.

Different levels are combined by a logical AND.

form
Relevant only to **REPORT** commands for formatted record reports. Ignored for all other types of **REPORT** command, **CSV** commands, and **EXTRACT** commands. Specifies the name of a form. The form must be for the log code specified by this **CODE** command; otherwise, it is ignored. The form controls the fields that are included in the formatted record report.

The form must be defined in the control repository specified by the FUWCDS ddname.

EXCLUDE
Specifies that this **CODE** command is to *exclude* matching log records from further processing.

By default, the **CODE** command *includes* matching log records for further processing.

COND statement

The **COND** statement specifies a filter condition. Filter conditions use field values to refine a filter.

Multiple **COND** statements for a **CODE** command are combined by a logical AND, with the following exception: consecutive **COND** statements that specify the same field name (or offset) and operator, where the operator is something other than **NE**, are combined by a logical OR.

For example:

Logical AND

The following consecutive **COND** statements select a log record if its transaction code is MENU *and* its user ID is GXH01:

```
COND TRANCODE EQ 'MENU'
COND USERID EQ 'GXH01'
```

The following consecutive **COND** statements select a log record if its transaction code is not MENU *and* not LOGO (neither MENU nor LOGO):

```
COND TRANCODE NE 'MENU'
COND TRANCODE NE 'LOGO'
```

(Same field name and operator; operator is **NE**.)

Logical OR

The following consecutive **COND** statements select log records if their transaction code is MENU *or* LOGO:

```
COND TRANCODE EQ 'MENU'
COND TRANCODE EQ 'LOGO'
```

(Same field name and operator; operator is something other than **NE**.)

Example

```
REPORT
  CODE(CMF:6E13)
    COND TRANCODE EQ 'CCV*'
EXTRACT
  CODE(CMF:6E13)
    COND TRANCODE EQ 'CCV*'
```

Related concepts

Filters: Log record selection criteria

Filters enable you to select the log records that you want to analyze and exclude others. For example, you can define a filter to select only those records of a particular log type and code that are associated with a particular transaction code or user ID. You can use a filter when browsing logs in the ISPF dialog or when writing JCL for the report and extract utility.

Related reference

Filter conditions

A filter condition is an expression that selects log records based on a field value. Conditions refine filters. Without conditions, filters select log records based only on log types and codes.

Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

ELAPSED command

Displays elapsed time since the previous log record in either microsecond or nanosecond precision. Applies only to reports that display elapsed time between log records.

The **ELAPSED** command applies only if the preceding action command is **REPORT**. Otherwise, **ELAPSED** is ignored.

Format

Parameters

MICRO
Shows seconds to microsecond precision: six decimal places. This is the default value.

NANO
Shows seconds to nanosecond precision: nine decimal places. A period separator appears before the additional three decimal places.

Example

```
+0004  Code... 43    Log Data Set Control
+011E  STCK... C07471C201606C40      LSN.... 0000000000000B3B      Record... 1
       Date... 2007-04-16 Monday     Time... 16.41.56.465158.765

+0004  Code... 42    Log Buffer Control
+0248  STCK... C07471C201609300      LSN.... 0000000000000B3C      Record... 2
       Date... 2007-04-16 Monday     Time... 16.41.56.465161.187   Elapsed.. 0.000002.421

+0004  Code... 03    Output Message
+01D4  STCK... C07471C201636762      LSN.... 0000000000000B3D      Record... 3
       Date... 2007-04-16 Monday     Time... 16.41.56.465206.461   Elapsed.. 0.000045.273
```

*Figure 140. Report output from **ELAPSED** NANO command*

FIELDS command

Specifies which fields to include in the output of the **CSV** and **JSON** commands.

The **FIELDS** command applies only if the preceding action command is **CSV** or **JSON**. Otherwise, **FIELDS** is ignored.

FIELDS is only for use with a **CSV** or **JSON** command that specifies a **CODE** parameter, not a **FORM** parameter.

By default, if you do not specify **FIELDS**, the output includes all fields.

To include all fields from a particular section of a log record, use section parameters to specify the section instead of using **FIELDS** to specify each field.

Tip: To list all field names for a log code, create a form: on the primary option menu of the Transaction Analysis Workbench ISPF dialog, select option 2 **Forms**. Enter **NEW** on the command line. Specify the log type and code, and then press Enter. A new form is displayed, containing all fields in that log code. To exit without saving the new form, press the Cancel function key (F12).

FIELDS command versus FIELDS parameter

You can specify **FIELDS** either as a parameter of a **CSV** or **JSON** command, or as this qualifying command immediately following a **CSV** or **JSON** command.

The format and behavior of the parameter and command are identical, with one exception: with the parameter format, if the list of fields spans multiple lines, you must specify a continuation character (+) at the end of each line. For this reason, the command format is typically easier to use.

Order of fields in the output

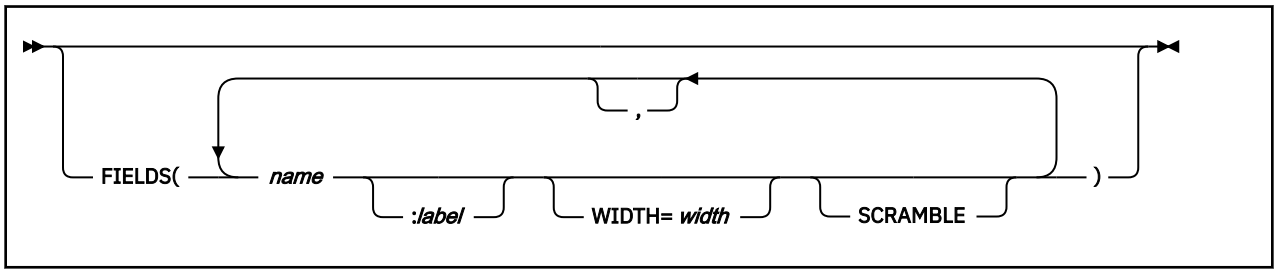
The **FIELDS** command does not affect the order of fields in the output CSV or JSON. The order of fields in the output CSV or JSON matches the order in the corresponding knowledge module. Typically, the order of fields in a knowledge module matches the order in the original log record..

The **METADATA** parameter of the **CSV** or **JSON** command creates JSON metadata that contains a **sequence** property that reflects the order of fields in the **FIELDS** command.

Representing bit flags as separate fields in the output

Fields that represent a single bit are known as *bit flag* fields. The value of an individual bit flag field is output to JSON as the JavaScript Boolean value `true` or `false`; to CSV, as the number 1 or 0. The accumulated value of multiple instances of a bit flag field from repeating sections is output to both JSON and CSV as an integer that represents the number of instances where the bit flag is on. For details on when field values from repeating sections are output individually or accumulated, see the section parameters.

Format



name
Field names must refer to fields that belong to the log type and code specified by the **CODE** parameter of the **CSV** or **JSON** command.

Field names can be delimited by a blank or a comma.

The last character of a field name can be a wildcard (*) character. For example, `FIELDS(ABC*,XYZ*)` selects all fields that start with either "ABC" or "XYZ".

label
By default, CSV column labels and JSON property names match the field names. To use a different label (or "alias"), append a colon to the field name, followed by the label. For example, `SMF30CPT:CPUTime`.

If the label includes blanks, enclose the label in single or double quotation marks. For example, `SMF30CPT:"CPU time"`.

To control the case of labels in the output CSV or JSON, use the **FIELDCASE** parameter of the **CSV** or **JSON** command. The default is all lowercase, `FIELDCASE(LOWER)`. To output labels in the same case that you specify them, use `FIELDCASE(ASIS)`.

WIDTH
Sets the maximum width of a character field in the output CSV or JSON. Only applies to character fields; ignored if specified for a non-character field.

The effect of **WIDTH** depends on whether the **FIXEDWIDTH** parameter of the **CSV** or **JSON** command is specified:

- If **FIXEDWIDTH** is specified, then **WIDTH** specifies the *fixed* output width. Longer values will be truncated, shorter values will be right-padded with blanks.
- Otherwise, **WIDTH** specifies the *maximum* output width. Longer values will be truncated, but shorter values will not be padded.

SCRAMBLE
Scrambles the value of a character field in the output CSV or JSON, making the original value unrecognizable. Only applies to character fields; ignored if specified for a non-character field.

Examples

The following example selects three fields for output, and scrambles one of them:

```
CSV TYPE(SMF:30.) LABELS
FIELDS(
  SMF30RUD SCRAMBLE
  SMF30JBN
  SMF30CPT
)
```

The column labels in the CSV header row are the field names, in all lowercase.

The following example is the same as the previous example, except that the column labels in the CSV header row are mixed-case aliases:

```
CSV TYPE(SMF:30.) LABELS FIELDCASE(ASIS)
FIELDS(
  SMF30RUD:"User ID" SCRAMBLE
  SMF30JBN:"Job name"
  SMF30CPT:"CPU time"
)
```

Related tasks

Scrambling character field values in CSV and JSON output
When converting logs to CSV or JSON, you can specify which character fields to scramble. Scrambling a field makes its original value unrecognizable.

Related reference

CSV and JSON commands
The **CSV** command converts logs to comma-separated values (CSV) format. The **JSON** command converts logs to JavaScript Object Notation (JSON) format. The **CSV** and **JSON** commands have similar syntax with many common parameters. Both commands can write to a ddname, a z/OS UNIX file path, or a TCP socket.

Section parameters
The section parameters of the **CSV** and **JSON** commands specify which sections of a log record to include in the output, and how to handle repeating sections.

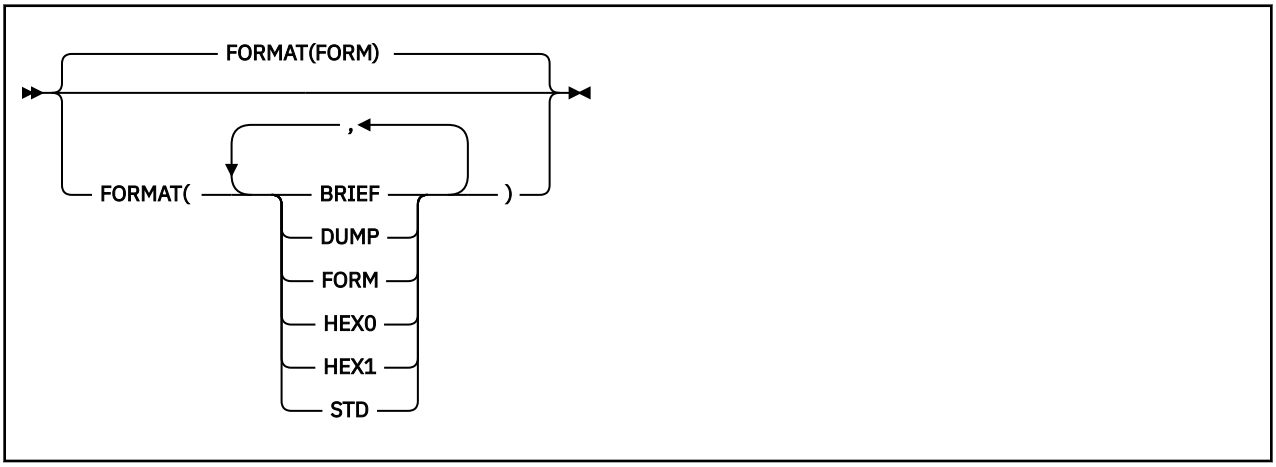
FORMAT command

Specifies options for presenting data in formatted record reports, matching the options that you can specify when browsing records in the ISPF dialog. Ignored for extracts, CSV, and JSON output.

The **FORMAT** command applies only if the preceding action command is a **REPORT** command for a formatted record report. Otherwise, **FORMAT** is ignored.

Each value that you specify generates a report of that format. For example, specifying `FORMAT(STD,DUMP)` generates two formatted record reports: one in standard format, one in dump format.

Format



Parameters

BRIEF
Reproduces the format of the ISPF dialog log browser list panel, where each log record occupies a few lines at most, showing only general and high-level information such as the log record code, description, and global fields.

FORM
Shows the fields selected by the specified form. You can specify a form in the following ways:

- Using the **FILTER** parameter of the **REPORT** command.
- Using the **FORM** parameter of the **CODE** command.

FORM is the default value for the **FORMAT** command. However, if you do not specify a form, then the report uses standard (STD) formatting.

The form must be defined in the control repository specified by the FUWCDS ddname.

For details of the other formats, see Chapter 25, “Formatting log records,” on page 151.

Related tasks

Formatting log records
You can display log records in several formats. The standard Transaction Analysis Workbench format uses knowledge modules to display log records organized into segments, with field names next to field values. Dump formats show raw log record data without using knowledge modules.

Related reference

REPORT command for formatted record reports
Requests a formatted record report. Formatted record reports reproduce how the ISPF dialog log browser displays log records.

PARM command

Specifies options for the current action. The available options depend on the action.

The **PARM** command applies to the current action. Furthermore, the **PARM** command applies only to actions that require input in addition to the action command.

For details on whether or not the **PARM** command applies to a particular action, and the options that you can specify in the **PARM** command, see the documentation for each action command.

Usage

All of the following example **PARM** commands are equivalent.

If you specify multiple parameters on the same line in a **PARM** command, separate the parameters with commas, with no spaces between the parameters. For example:

```
PARM(LIST=DB2X,EXTRACT=EXTRACT,APPORTIONROLLUP,RESPONSE>1)
```

You can continue a **PARM** command over multiple lines. If you specify a single parameter per line, omit the separating commas. For example:

```
PARM(
  LIST=DB2X
  EXTRACT=EXTRACT
  APPORTIONROLLUP
  RESPONSE>1
)
```

If you specify multiple **PARM** commands for the same output command, the contents of the **PARM** commands are concatenated and treated as a single command. For example:

```
PARM(LIST=DB2X,EXTRACT=EXTRACT)
PARM(
  APPORTIONROLLUP
  RESPONSE>1
)
```

TRACK command

Activates tracking. Only log records associated with transactions specified by the selection filter (**COND** statements of **CODE** commands) are included in the output.

You can specify **TRACK** for each **REPORT** or **EXTRACT** command.

TRACK invokes tracking only if the first **CODE** command specified includes a **COND** statement.

To generate this command from the ISPF dialog, specify a filter with **Activate Tracking** selected.

Format

Parameters

None.

Example

The following example activates tracking for user ID DAVE. Only log records attributable to transactions run by DAVE will be reported.

```
REPORT
TRACK
CODE(ALL)
COND USERID EQ 'DAVE'
```

Administrative commands

Administrative commands copy filters, forms, or object lists (collectively known as *controls*) between Transaction Analysis Workbench control repositories.

Related concepts

Repositories

Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

EXPORT command

Copies selected filters, forms, and object lists (known collectively as *controls*) from a control repository to a tersed file.

After using **EXPORT**, you can then use the **IMPORT** command to copy the controls from the tersed file to another control repository. If the two control repositories are on different systems, you can use binary FTP to transfer the exported tersed file between systems.

The **EXPORT** command can occur only once in a SYSIN data set; if specified, **EXPORT** must be the only command in a SYSIN data set.

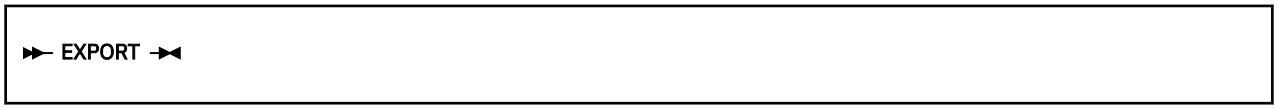
The **EXPORT** command can be followed by statements that specify which controls to copy: **FILTER**, **FORM**, and **OBJECTLIST**. Each statement can occur only once. If you do not specify any statements, **EXPORT** copies all controls. If you specify one or more statements, **EXPORT** copies only the specified controls.

Each **FILTER**, **FORM**, and **OBJECTLIST** statement must begin on a new line of the SYSIN data set, and can continue across multiple lines. Each line of a statement, except for the last line, must end with a comma that separates the last name mask on the current line with the first name mask on the next line. For example:

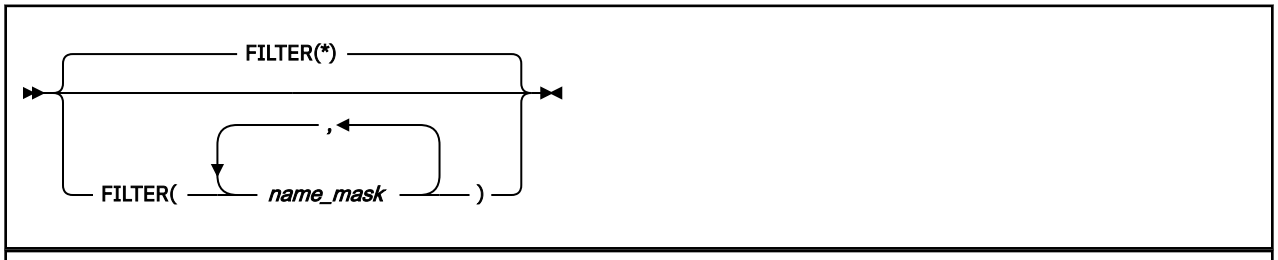
```
EXPORT
FILTER(abc01,abc02,abc03,
def*,ghi*,
jkl01,jkl02)
FORM(*)
```

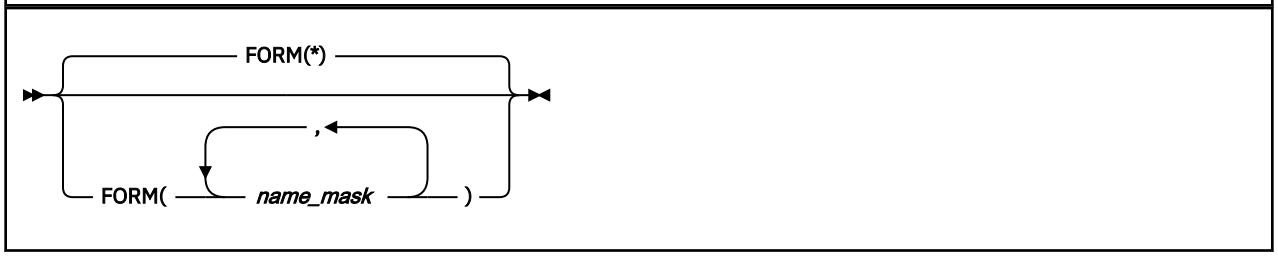
If a filter that is to be copied refers to forms or object lists, then, to maintain referential integrity, **EXPORT** also copies those forms and object lists, regardless of any **FORM** or **OBJECTLIST** statements.

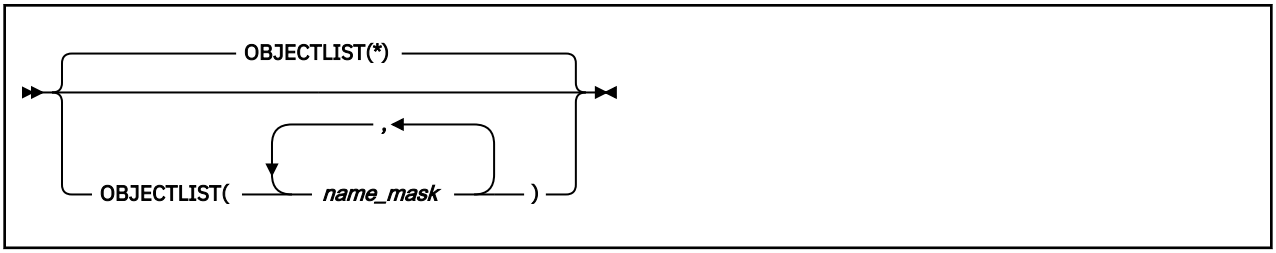
Format



Statements that can follow an **EXPORT** command:







Parameters

name_mask

The name of a filter, form, or object list that you want to copy. You can specify names explicitly or use the masking character * to represent any number of characters or % for a single character.

The following example copies filters whose names begin with MY (and any forms and object list to which these filters refer):

```
EXPORT
FILTER(MY*)
```

The following example copies three explicitly named forms:

```
EXPORT
FORM(ABC01,ABC02,ABC03)
```

The following example copies all object lists and all forms:

```
EXPORT
FORM(*)
OBJECTLIST(*)
```

Example

The following JCL exports all filters, forms, and object lists from the control repository MY.FUW.CONTROLS to the tersed file MY.FUW.EXPORT:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//FUWCDS   DD DISP=SHR,DSN=MY.FUW.CONTROLS
//AMAWORK1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//AMAPRINT DD SYSOUT=*
//FUWTERSE DD DISP=(,CATLG),
//            DSN=MY.FUW.EXPORT,
//            UNIT=SYSDA,SPACE=(CYL,(1,1)),
//            DCB=(RECFM=FB,LRECL=1024,BLKSIZE=6144)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
EXPORT
/*
```

IMPORT command

Copies filters, forms, and object lists from a tersed file, that was created by the **EXPORT** command, to a control repository.

The **IMPORT** command can occur only once in a SYSIN data set; if specified, **IMPORT** must be the only command in a SYSIN data set.

Format

Parameters

REPLACE|<u>NOREPLACE</u>

Specify REPLACE to overwrite existing items in the control repository that have the same name as items of that type (filter, form, or object list) in the tersed file that you are importing. Specify NOREPLACE to preserve existing items. If neither is specified, NOREPLACE is the default.

Example

The following JCL imports filters, forms, and object lists in the tersed file MY.FUW.EXPORT (created by an **EXPORT** command) to the control repository MY.FUW.CONTROLS. The import does not overwrite existing items in the control repository.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWBATCH EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//FUWCDS   DD DISP=SHR,DSN=MY.FUW.CONTROLS
//AMAWORK1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//AMAPRINT DD SYSOUT=*
//FUWTERSE DD DISP=OLD,DSN=MY.FUW.EXPORT
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
IMPORT
/*
```

Chapter 76. System take-up utility

The system take-up utility reads a plain-text file of system definitions, and then defines the corresponding systems in Transaction Analysis Workbench system definition repositories.

If you have many systems to define, consider using this batch utility instead of the interactive Transaction Analysis Workbench ISPF dialog (option 3 **Systems**).

Optionally, you can use IBM Tivoli Discovery Library Adapter for z/OS (z/OS DLA) to discover your systems, and then use the output from the z/OS DLA discovery job as a starting point for input to the Transaction Analysis Workbench system take-up utility. To streamline this optional step, Transaction Analysis Workbench supplies a REXX exec that reads ZOSTASK IdML books (XML files) created by z/OS DLA, and then writes a file of system definitions in the plain-text syntax required by the system take-up utility. Before running the system take-up utility, you can edit the output from the REXX exec and annotate it with additional system attributes used by Transaction Analysis Workbench; for example, RECON data sets for IMS, or bootstrap data sets for DB2. z/OS DLA is a component of various IBM products, such as IBM Tivoli Application Dependency Discovery Manager (TADDM). For details of z/OS DLA availability, contact IBM Software Support.

If you do not have z/OS DLA, you can create the input to the system take-up utility in a text editor.

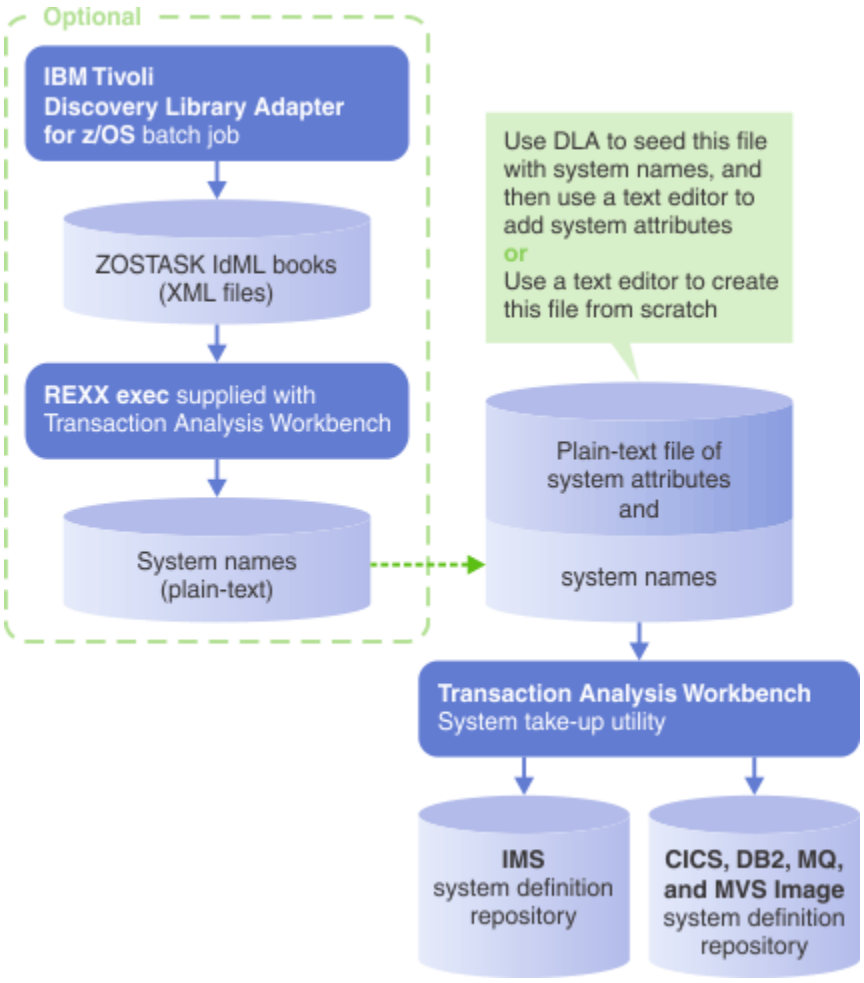


Figure 141. Overview of the Transaction Analysis Workbench system take-up utility

Limitations

The system take-up utility only takes up the systems and attributes that are used by Transaction Analysis Workbench. This is a subset of the information that can be stored in system definition repositories. Transaction Analysis Workbench can share repositories with other applications. Some of those applications might use systems or attributes that Transaction Analysis Workbench does not use.

The system take-up utility takes up cyclic SMF file and SMF log stream attributes for MVS Image systems only, even though CICS, DB2, and MQ system definitions can also contain those attributes, and even though the Transaction Analysis Workbench automated file selection utility can use those attributes in those system definitions. This limitation encourages the practice of associating CICS, DB2, and MQ system definitions with an MVS Image name, rather than specifying cyclic SMF file and SMF log stream attributes separately for each subsystem. If you use the Transaction Analysis Workbench automated file selection utility to locate the SMF records for CICS, DB2, or MQ system definitions, then, if those definitions do not contain their own cyclic SMF file and SMF log stream attributes, the automated file selection utility uses the attributes of the associated MVS Image.

For details on the systems and attributes supported by the system take-up utility, see the syntax for defining systems.

Some update capabilities

The primary function of the system take-up utility is to take up (insert) *new* system definitions into the system definition repositories. If the input to the system take-up utility identifies existing system definitions, then, by default, the utility leaves the existing definitions unchanged. However, you can choose instead to update existing definitions; to insert, replace, or delete attributes. For details, see the **DUPLICATE** parameter of the utility.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Related reference

Syntax for defining systems
The SYSUT1 input data set to the system take-up utility can contain the following types of records: system attributes, system definition statements, comments, and blank lines.

System take-up utility parameter: DUPLICATE
The system take-up utility can read an optional SYSIN data set that contains a single parameter, **DUPLICATE**. The **DUPLICATE** parameter controls what the utility does if the *key* of a system definition statement in the SYSUT1 input data set duplicates the key of an existing system definition.

REXX exec to extract system names from ZOSTASK IdML books

If you have IBM Tivoli Discovery Library Adapter for z/OS (z/OS DLA), you can use the ZOSTASK IdML books from z/OS DLA discovery jobs as a starting point for system take-up. Transaction Analysis Workbench supplies a REXX exec that reads ZOSTASK books, and then writes a plain-text file of system names in the syntax required by the Transaction Analysis Workbench system take-up utility.

The REXX exec is supplied in member FUWIDMLP of library SFUWEXEC.

JCL

The following JCL runs the REXX exec:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//IDMLP    EXEC PGM=IRXJCL,PARM=FUWIDMLP
//SYSEXEC  DD  DISP=SHR,DSN=FUW.SFUWEXEC
//* Input
```

```
//SYSUT1   DD  DISP=SHR,DSN=IZD.IDML.MYZ1(ZOSTASK)
//         DD  DISP=SHR,DSN=IZD.IDML.MYZ2(ZOSTASK)
//         DD  DISP=SHR,DSN=IZD.IDML.MYZ3(ZOSTASK)
//* Output
//SYSUT2   DD  DSN=MY.FUW.TAKEUP,DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE)
//* Messages
//SYSTSPRT DD  SYSOUT=*
//
```

DD statements:

SYSEXEC
Specifies the data set name of the Transaction Analysis Workbench SFUWEXEC library that contains the REXX exec. The member name of the REXX exec, FUWIDMLP, is specified in the **PARM** parameter of the preceding **EXEC** statement.

SYSUT1
The input: a ZOSTASK book, or a concatenation of ZOSTASK books, created by z/OS DLA.

For details on creating ZOSTASK books, see the z/OS DLA documentation.

SYSUT2
The output: a list of systems in the plain-text syntax required by the Transaction Analysis Workbench system take-up utility.

SYSTSPRT
Messages from the REXX exec.

For general information about running REXX execs, see the z/OS TSO/E REXX documentation.

Output

Example snippet of system definitions in the SYSUT2 output data set:

```
IMAGE=MYZ1
IMS=MYIA VERSION=154 TYPE=DBCTL
IMS=MYIB VERSION=154 TYPE=DBDC
CICS=MYCICSA
CICS=MYCICSB
DB2=MYDA DSG=MYGRPA
MQ=MYMA

IMAGE=MYZ2
⋮
```

Before using the SYSUT2 output data set as input to the Transaction Analysis Workbench system take-up utility:

1. Check that the list of systems matches your expectations. Delete the entries for any systems that you do not want to take up.
2. Annotate the definitions with system attributes.

Related reference

Syntax for defining systems
The SYSUT1 input data set to the system take-up utility can contain the following types of records: system attributes, system definition statements, comments, and blank lines.

System take-up utility JCL

The JCL to run the system take-up utility consists of a single step that runs the program FUWSLOAD.

Attention: Before using the system take-up utility to update existing repositories, back up the repositories.

To update existing repositories

```
//UIDFUW   JOB NOTIFY=&SYSUID
//* System take-up utility
//TAKEUP   EXEC PGM=FUWSLOAD
//STEPLIB  DD  DISP=SHR,DSN=FUW.SFUWLINK
//* Input parameter
//SYSIN    DD  *
DUPLICATE=IGNORE /* or UPDATE
/*
//* Input system definitions (plain text)
//SYSUT1   DD  DISP=SHR,DSN=MY.FUW.TAKEUP
//* Output system definition repository: CICS and others
//SYSDEFC  DD  DISP=SHR,DSN=MY.FUW.CICS.REPOSTRY
//* Output system definition repository: IMS
//SYSDEFI  DD  DISP=SHR,DSN=MY.FUW.IMS.REPOSTRY
//* Messages and take-up summary
//SYSPRINT DD  SYSOUT=*
//
```

To create new repositories

To create new repositories, make the following changes to the previous JCL listing:

1. Insert a new first step that uses the IDCAMS utility to allocate the repository for CICS and other (non-IMS) system definitions.
2. In the SYSDEFI DD statement, replace the DISP=SHR parameter with parameters that allocate a new partitioned data set (PDS).

The following JCL snippet shows the changes required:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//* Define VSAM KSDS repository for CICS and other (non-IMS) system definitions
//IDCAMS   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DEFINE CLUSTER( -
           NAME(MY.FUW.CICS.REPOSTRY) -
           INDEXED  -
           KEYS(64 0)-
           SHAREOPTIONS(3 3) -
           FREESPACE(50 50) -
           RECORDSIZE(1024 32756) -
           CYLINDERS(5 5) -
           ) -
         DATA(NAME(MY.FUW.CICS.REPOSTRY.DATA)) -
         INDEX(NAME(MY.FUW.CICS.REPOSTRY.INDEX))
/*
⋮
//SYSDEFI  DD  DSN=MY.FUW.IMS.REPOSTRY,
//             DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(1,1,20))
⋮
```

Input DD statements

SYSIN
Optional. Contains a **DUPLICATE** parameter that controls the behavior of the system take-up utility.

SYSUT1
Contains system definitions in the plain-text syntax required by the system take-up utility.

Output DD statements

SYSDEFI
IMS system definition repository where you want to store taken-up system definitions. Only required if SYSUT1 contains IMS system definitions.

The IMS system definition repository is a PDS. You can allocate this data set in the same step that runs the system take-up utility.

SYSDEFC
CICS, DB2, IBM MQ, and MVS Image system definition repository where you want to store taken-up system definitions. Only required if SYSUT1 contains any of these types of system definitions.

The repository for CICS and other (non-IMS) system definitions is a VSAM key-sequenced data set (KSDS). You must allocate this data set before running the system take-up utility.

SYSPRINT
Messages and take-up summary.

Examples

The following members of the Transaction Analysis Workbench sample library SFUWSAMP contain example JCL that runs the system take-up utility:

FUWTAKE1
For users who have z/OS DLA.

Performs the following steps:

1. Runs the IDCAMS utility to delete and then define "sandbox" system definition repositories under your personal high-level qualifier (HLQ), &SYSUID.
2. Runs the REXX exec FUWIDMLP to extract system names from one or more ZOSTASK IdML books previously created by z/OS DLA.
3. Runs the system take-up utility using a SYSUT1 input data set that is a concatenation of the following two data sets:
   a. An in-stream data set containing system attributes
   b. System definition statements created by the REXX exec in the previous step
4. Writes the output from the REXX exec to the output spool.

FUWTAKE2
An abridged copy of FUWTAKE1 for users who do not have z/OS DLA, or who want to run the REXX exec FUWIDMLP and the system take-up utility in separate jobs.

Performs the following steps:

1. Runs IDCAMS to delete and then define "sandbox" system definition repositories under your personal HLQ.
2. Runs the system take-up utility using a SYSUT1 input data set that you provide. You can use sample member FUWTAKE3 as a starting point for your own input.

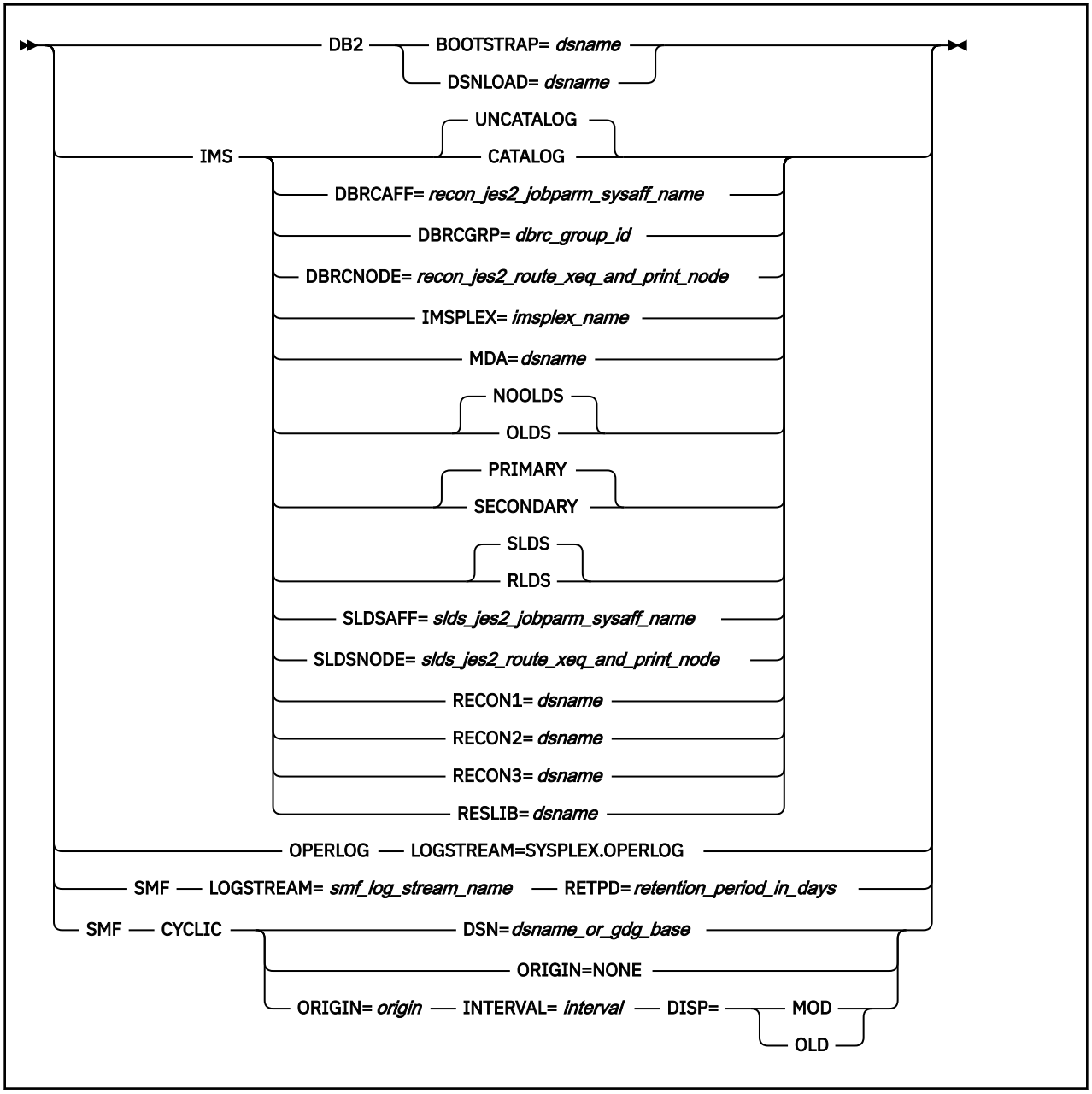
Syntax for defining systems

The SYSUT1 input data set to the system take-up utility can contain the following types of records: system attributes, system definition statements, comments, and blank lines.

System attributes

System attributes specify values for system definitions, such as RECON data set names for IMS systems, and the bootstrap data set name for DB2 systems. System attributes do not, by themselves, define systems. System attribute values are used by subsequent system definition statements.

Here is a summary of the syntax of all system attributes, for all system types:



For details, see the topics on the syntax for defining each system type.

Typically, the first keyword in a system attribute, such as DB2 or IMS, indicates the type of system to which the attribute applies. However, there are exceptions: system attributes that begin with the keywords SMF or OPERLOG apply only to MVS Images (IMAGE system type).

System attributes that specify data set names or log stream names, represented in the syntax diagram by *dsname*, must specify fully qualified data set names without enclosing quotation marks. After take-up, the corresponding Transaction Analysis Workbench ISPF dialog panels show the data set names in quotation marks.

Substitution variables in system attributes

System attributes that specify data set names or log stream names can refer to substitution variables.

Table 32. Substitution variables		
Variable name	**Description**	**Applicable system types**
DB2	DB2 subsystem ID, from the current **DB2** system definition statement	DB2
IMAGE	MVS image name (SMF ID), from either: • The current **IMAGE** system definition statement • For DB2 and IMS systems, the *most recent* **IMAGE** system definition statement	DB2, IMAGE, IMS
IMS	IMS subsystem ID, from the current **IMS** system definition statement	IMS

The rules for coding substitution variables follow the rules for coding JCL symbols:

- Prefix the variable name with an ampersand (&).
- Where required by the rules for coding JCL symbols, follow the variable name with a period.
- If a period follows the variable, follow the variable name with two consecutive periods.

For example:

- The following system attributes and system definition statement:

```
IMS RECON1=&IMS..RECON1
IMS RECON2=&IMS..RECON2
IMS RECON3=&IMS..RECON3
IMS=IMSA VERSION=154
```

  define an IMS system named IMSA with RECON data sets `'IMSA.RECON`*n*`'`.
- The following system attributes and system definition statements:

```
SMF LOGSTREAM=IFASMF.&IMAGE RETPD=14
DB2 DSNLOAD=DSN.&IMAGE..&DB2..SDSNLOAD
IMAGE=MYZ1
DB2=MYDB
```

  define two systems:

  – An MVS Image named MYZ1 with SMF log stream `IFASMF.MYZ1`
  – A DB2 system named MYDB with DSNLOAD library `'DSN.MYZ1.MYDB.SDSNLOAD'`

If you specify SYSUT1 as an in-stream data set, then you can mix substitution variables with JCL symbols. For example, the following JCL snippet and SYSUT1 input records:

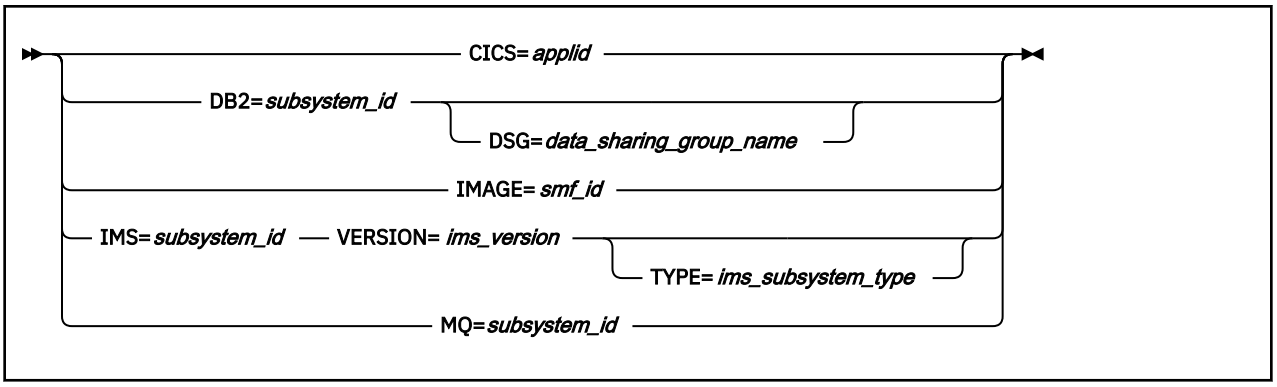
```
//         EXPORT SYMLIST=*
//         SET  IMSV=154
//SYSUT1   DD   *,SYMBOLS=JCLONLY
IMS RESLIB=&IMAGE..&IMS..V&IMSV..SDFSRESL
IMAGE=MYZ1
IMS=IMSA VERSION=&IMSV
```

define an MVS Image named MYZ1, and an IMS system named IMSA with the RESLIB data set name `'MYZ1.IMSA.V154.SDFSRESL'`.

System definition statements

System definition statements insert a new definition or update an existing definition in a system definition repository. Whether the utility updates existing definitions or leaves them unchanged depends on the **DUPLICATE** input parameter.

Here is a summary of the syntax of all system definition statements:



For details, see the topics on the syntax for defining each system type.

Tip: To distinguish system attributes from system definition statements, look at the character following the first keyword. System attributes have a space after the first keyword (DB2 B00TSTRAP=...). System definition statements have an equal sign after the first keyword (DB2=...).

System definition statements use system attributes from preceding input records. More specifically, system definitions use *the most recent* preceding system attributes. More recent system attributes take precedence over system attributes on earlier input records.

The following example SYSUT1 snippet demonstrates the precedence of system attributes:

```
IMS=IMSA VERSION=154

IMS RECON1=&IMS..PROD.RECON1
IMS RECON2=&IMS..PROD.RECON2
IMS RECON3=&IMS..PROD.RECON3

IMS=IMSB VERSION=154
IMS=IMSC VERSION=154

IMS RECON1=&IMS..TEST.RECON1
IMS RECON2=&IMS..TEST.RECON2
IMS RECON3=&IMS..TEST.RECON3

IMS=IMSD
```

- IMSA (assuming that IMSA does not already exist) does not refer to any RECON data sets
- IMSB and IMSC refer to RECON data sets with the mid-level qualifier PROD
- IMSD refers to RECON data sets with the mid-level qualifier TEST

System attributes with a blank value

System attributes with a blank value (no value after the equal sign) cause subsequent system definition statements to define systems without that attribute.

For example, given the following input:

```
SMF LOGSTREAM=IFASMF.&IMAGE RETPD=14
IMAGE=MYZ1
IMAGE=MYZ2
SMF LOGSTREAM=
IMAGE=MYZ3
```

- MYZ1 refers to the SMF log stream `IFASMF.MYZ1`
- MYZ2 refers to the SMF log stream `IFASMF.MYZ2`
- MYZ3 does not refer to an SMF log stream

If a system attribute normally requires more than one parameter, specify the first parameter only, with a blank value. For example, for the SMF LOGSTREAM= system attribute, omit the second **RETPD** parameter that is normally required.

If you specify DUPLICATE=UPDATE in the SYSIN data set to the system take-up utility, then system attributes with a blank value cause subsequent system definition statements for existing systems to *delete* those attributes from those existing systems.

For example, to delete the RECON data set names from an existing IMS system definition named IMSA, specify DUPLICATE=UPDATE in SYSIN and the following input records in SYSUT1:

```
IMS RECON1=
IMS RECON2=
IMS RECON3=
IMS=IMSA
```

System definition statements that occur before system attributes

For new system definitions, a system definition statement that occurs before a system attribute defines a system without that attribute.

For example, given the following input:

```
OPERLOG LOGSTREAM=SYSPLEX.OPERLOG
IMAGE=MYZ1
SMF LOGSTREAM=IFASMF.&IMAGE RETPD=14
IMAGE=MYZ2
SMF LOGSTREAM=
IMAGE=MYZ3
```

- MYZ1 does not refer to an SMF log stream, because there is no corresponding preceding system attribute
- MYZ2 refers to the SMF log stream IFASMF.MYZ2
- MYZ3 does not refer to an SMF log stream, because it follows the corresponding system attribute that specifies a blank value
- All three system definitions refer to the OPERLOG log stream

For existing system definitions, if you specify DUPLICATE=UPDATE in the SYSIN data set to the system take-up utility, a system definition statement that occurs before a system attribute has *no effect* on that attribute in the existing definition.

For example, given the same input as the previous listing, but this time with DUPLICATE=UPDATE in SYSIN, and where MYZ1, MYZ2, and MYZ3 already exist in the system definition repository:

- The system definition statement for MYZ1 has the following effect:
  - Updates the OPERLOG attribute
- The system definition statement for MYZ2 has the following effect:
  - Updates the OPERLOG attribute
  - Updates the SMF log stream attribute (consisting of the log stream name and retention period)
- The system definition statement for MYZ3 has the following effect:
  - Updates the OPERLOG attribute
  - Deletes the SMF log stream attribute
- All other attributes of the existing system definitions are unchanged.

CICS, DB2, and MQ system definitions refer to the last specified MVS image

CICS, DB2, and MQ system definitions can optionally refer to an MVS image.

When defining CICS, DB2, or MQ systems, the system take-up utility applies the MVS image name of the most recent **IMAGE** system definition statement, if any.

For example, given the following input:

```
CICS=MYCICSA

IMAGE=MYZ1
DB2=MYDA
MQ=MYMA

IMAGE=MYZ2
CICS=MYCICSB
DB2=MYDB
```

- MYCICSA does not refer to an MVS image, because there is no preceding **IMAGE** statement
- MYDA and MYMA refer to the MVS image MYZ1
- MYCICSB and MYDB refer to the MVS image MYZ2

Note:

- When creating the SYSUT1 input data set, group system definition statements under their "parent" **IMAGE** definition statements, as shown in the preceding example.
- To define a CICS, DB2, or MQ system that does not refer to an MVS image, insert the system definition statement before the first **IMAGE** statement.
- You cannot use the system take-up utility to delete the MVS Image name from an existing CICS, DB2, or MQ system definition.

Comments

A comment line begins with an asterisk (*) in column one. An inline comment begins after column one, and after any command on the line, with a slash followed by an asterisk (/*).

```
//SYSUT1 DD *
* Comment line
IMAGE=MYZ1 /* Inline comment

/*
```

Related reference

System take-up utility parameter: DUPLICATE

The system take-up utility can read an optional SYSIN data set that contains a single parameter, **DUPLICATE**. The **DUPLICATE** parameter controls what the utility does if the *key* of a system definition statement in the SYSUT1 input data set duplicates the key of an existing system definition.

Syntax for defining CICS systems

To define CICS systems, insert **CICS** system definition statements in the SYSUT1 input data set to the system take-up utility.

System attributes

The system take-up utility does not support any CICS system attributes.

System definition statement

Example

Given the following input:

```
CICS=MYCICSA

IMAGE=MYZ1
CICS=MYCICSB
CICS=MYCICSC

IMAGE=MYZ2
CICS=MYCICSD
```

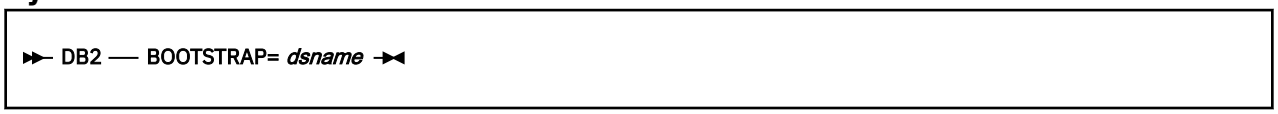
The system take-up utility defines the following CICS systems:

- MYCICSA, with no MVS image name, because there was no **IMAGE** statement preceding the corresponding **CICS** statement
- MYCICSB and MYCICSC, with MVS image name MYZ1
- MYCICSD, with MVS image name MYZ2

Syntax for defining DB2 systems

To define DB2 systems, insert **DB2** system definition statements in the SYSUT1 input data set to the system take-up utility. Optionally, you can precede the system definition statements with system attributes.

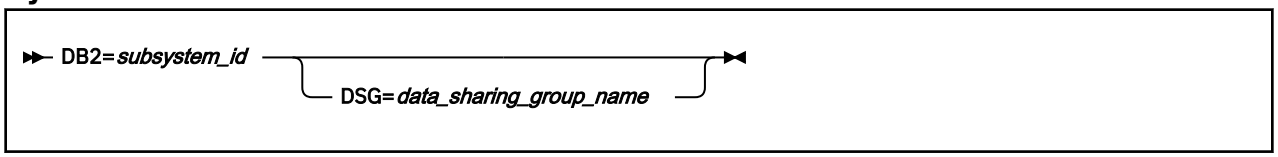
System attributes



BOOTSTRAP
The DB2 bootstrap data set (BSDS). Contains log file details such as data set names and time stamps.

DSNLOAD
The DB2 load module library. For example: DSN.SDSNLOAD

System definition statement



DSG
Optional. Sets the **Data sharing** field of the DB2 system definition to "Yes", "No", or leaves the field blank, according to the following rules:

- If a DSNLOAD library or a DB2 bootstrap data set has been specified for the DB2 system:
  - If the **DSG** parameter specifies any nonblank *data_sharing_group_name* value, then the system take-up utility sets **Data sharing** to "Yes".
  - Otherwise, if the **DSG** parameter is omitted or specifies a blank value, then the system take-up utility sets **Data sharing** to "Yes".
- If neither a DSNLOAD library nor a DB2 bootstrap data set has been specified for the DB2 system, the system take-up utility ignores any **DSG** parameter, and leaves **Data sharing** blank.

While the **DSG** parameter value is characterized in the syntax diagram as a data sharing group name, the actual value is not significant for take-up; the value does not appear in the system definition.

Tip: If you know that a DB2 system belongs to a data sharing group, but you do not know the name of the group, then, for the purposes of take-up, you can specify any nonblank value for the group name, such as DSG=YES.

To set the **Data sharing** field to "No" in an existing DB2 system definition (that specifies either a DSNLOAD library or a DB2 bootstrap data set), specify DUPLICATE=UPDATE in the SYSIN input data set to the system take-up utility and, in SYSUT1, specify a **DB2** system definition statement either without a **DSG** parameter or with a **DSG** parameter that specifies a blank value (no value after the equal sign). For example, given an existing DB2 system definition MYDA on MVS image MYZ1, the following two SYSUT1 snippets are equivalent:

```
IMAGE=MYZ1
DB2=MYDA
```

```
IMAGE=MYZ1
DB2=MYDA DSG=
```

Example

Given the following input:

```
DB2 DSNLOAD=DSN.PROD.SDSNLOAD
DB2 BOOTSTRAP=&DB2..BSDS01

DB2=MYDA

IMAGE=MYZ1
DB2=MYDB DSG=DSG1
DB2=MYDC DSG=DSG2

IMAGE=MYZ2
DB2=MYDD DSG=DSG1
DB2 DSNLOAD=DSN.TEST.SDSNLOAD
DB2=MYDE
```

The system take-up utility defines the following DB2 systems:

- MYDA, and no MVS image name, because there was no **IMAGE** statement preceding the corresponding **DB2** statement
- MYDB and MYDC, with MVS image name MYZ1
- MYDD and MYDE, with MVS image name MYZ2

MYDA and MYDE do not belong to a data sharing group. MYDB and MYDD belong to the data sharing group DSG1. MYDC belongs to the data sharing group DSG2.

MYDA, MYDB, MYDC, and MYDD refer to the DB2 load module library DSN.PROD.SDSNLOAD. MYDE refers to DSN.TEST.SDSNLOAD.

Each DB2 system refers to its own BSDS, such as MYDA.BSDS01.

Syntax for defining IMS systems

To define IMS systems, insert **IMS** system definition statements in the SYSUT1 input data set to the system take-up utility. Optionally, you can precede the system definition statements with system attributes.

System attributes

UNCATALOG
SLDS data sets are not cataloged. Transaction Analysis Workbench specifies **UNIT**, **VOLSER**, and **LABEL** parameters in generated JCL. This is the default behavior.

CATALOG
SLDS data sets are cataloged. Transaction Analysis Workbench omits **UNIT**, **VOLSER**, and **LABEL** parameters from generated JCL.

PRIMARY
Use primary SLDS. This is the default behavior.

SECONDARY
Use secondary SLDS if they are available. Otherwise, use primary SLDS.

JES2 options:

DBRCNODE
Specifies the network node where RECON data sets are available, where the DBRC log selection job must run. Transaction Analysis Workbench generates JCL that contains JES2 **XEQ** and **ROUTE PRINT** control statements:

```
/*XEQ node
/*ROUTE PRINT node
```

DBRCAFF
Specifies the name of the system where the DBRC log selection job must run. Transaction Analysis Workbench generates JCL that contains a JES2 **JOBPARM SYSAFF** control statement:

```
/*JOBPARM SYSAFF=name
```

SLDSNODE
Specifies the network node where SLDS are available, where run report jobs must run. Transaction Analysis Workbench generates JCL that contains JES2 **XEQ** and **ROUTE PRINT** control statements:

```
/*XEQ node
/*ROUTE PRINT node
```

SLDSAFF
Specifies the name of the system where report jobs must run. Transaction Analysis Workbench generates JCL that contains a JES2 **JOBPARM SYSAFF** control statement:

```
/*JOBPARM SYSAFF=name
```

For more information on how and when Transaction Analysis Workbench generates these JES2 control statements, see the online help for the **JES2 Control Statements** field in Transaction Analysis Workbench ISPF dialog option 0.1 **Workbench Personal Settings**.

DBRCGRP
For automated file selection, specifies the name of the DBRC sharing group if RECON loss notification is used and the RECON data sets are used by more than one IMSplex.

If you request a report or extract for a system with this attribute, the Transaction Analysis Workbench ISPF dialog generates JCL with **IMSPLEX** and **DBRCGRP** parameters:

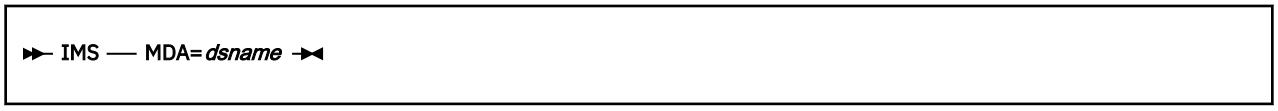
```
//FUWFILES EXEC PGM=FUWFILES,PARM='IMSPLEX=imsplex_name,DBRCGRP=dbrc_group_id'
```

The **DBRCGRP** system attribute is ignored when the system is part of a group report request.

IMSPLEX
For automated file selection, specifies the name of the IMSplex if the system uses RECON loss notification (RLN). If the RECON data sets are also used by other IMSplexes, also specify the DBRC sharing group ID (**DBCRGRP**).

If you request a report or extract for a system with this attribute, the Transaction Analysis Workbench ISPF dialog generates JCL with the **IMSPLEX** parameter:

```
//FUWFILES EXEC PGM=FUWFILES,PARM='IMSPLEX=imsplex_name'
```



MDA
The MVS Dynamic Allocation (MDA) data set for this IMS subsystem.

Transaction Analysis Workbench DBRC log selection uses the MDA data set to determine the RECON data set names if they are not specified, and in turn uses DBRC to select the log files for input to reporting.

If the RECON data set names are specified (see the **RECON** system attributes), the MDA data set is not required. However, if both are specified, Transaction Analysis Workbench validates that the RECON data sets match the MDA information.

NOOLDS
Do not use OLDS. This is the default behavior.

OLDS
If SLDS are not available for the requested reporting time period, use OLDS.

Be careful using the **OLDS** option. IMS reuses OLDS. The contents of OLDS might change before your batch job runs.

RECON*n*

The two or three RECON data sets used by this IMS subsystem.

Transaction Analysis Workbench DBRC log selection uses the RECON data sets to determine the log files to select for reporting.

If the MDA data set is specified (see the **MDA** system attribute), the RECON data sets can be omitted. However, if both are specified, Transaction Analysis Workbench validates that the RECON data sets match the MDA information.

You can update the RECON1, RECON2, and RECON3 data set names independently. For example, specifying system attributes for RECON1 and RECON2 does not affect the value of RECON3. In practice, it is likely that you will want to specify all three system attributes together. For example, suppose you have specified DUPLICATE=UPDATE in the SYSIN input data set to the system take-up utility, and you want to update an existing IMS subsystem, which currently refers to three RECON data sets, to refer to just two RECON data sets. In this case, remember to specify a blank value for RECON3 in SYSUT1:

```
IMS RECON1=&IMS..VNEW.RECON1
IMS RECON2=&IMS..VNEW.RECON2
IMS RECON3= /* Remove the obsolete value
IMS=MYIA VERSION=154
```

RESLIB

The RESLIB data set for this IMS subsystem. For DBRC Log Selection, the RESLIB data set must contain the DBRC API routine DSPAPI00.

The RESLIB data set can also be used to determine the IMS version. If you do not specify the VRM explicitly, Workbench sets the IMS Version from information in module DFSVC000.

You do not need to specify the RESLIB data set name if the modules reside in the system link list (LNKLST).

SLDS

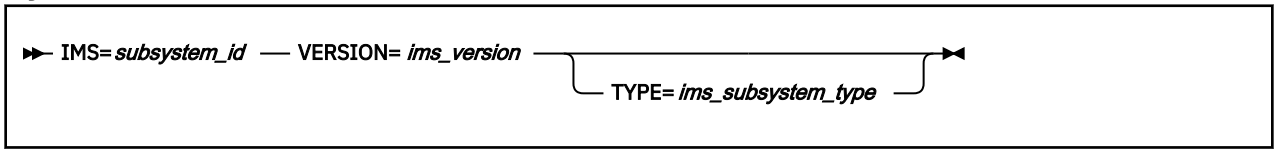
Use SLDS. Do not use recovery log data sets (RLDS) . This is the default behavior.

RLDS

Use RLDS if they are available. Otherwise, use SLDS.

Only a limited set of IMS reports can be run using RLDS.

System definition statement



VERSION
IMS version, in the format *vvr* (2-digit version number immediately followed by the single-digit release number, with no separator). For example: 154

TYPE
Not used by the system take-up utility; effectively, a comment. Specifies the IMS subsystem type:

DCCTL
Transaction Manager only.

DBCTL
Database Manager only.

DBDC
Both.

Example

Given the following input:

```
IMS RESLIB=IMS.V154.SDFSRESL
IMS RECON1=&IMS..RECON1
IMS RECON2=&IMS..RECON2
IMS RECON3=&IMS..RECON3

IMS=MYIA VERSION=154
IMS=MYIB VERSION=154

IMS RESLIB=IMS.V154.SDFSRESL
IMS RECON1=
IMS RECON2=
IMS RECON3=
IMS MDA=&IMS..MDALIB

IMS=MYIC VERSION=154
```

The system take-up utility defines the following IMS systems:

- MYIA, with RECON data sets `'MYIA.RECON`*n*`'`
- MYIB, with RECON data sets `'MYIB.RECON`*n*`'`
- MYIC, with MDA library `'MYIC.MDALIB'` and no RECON data sets

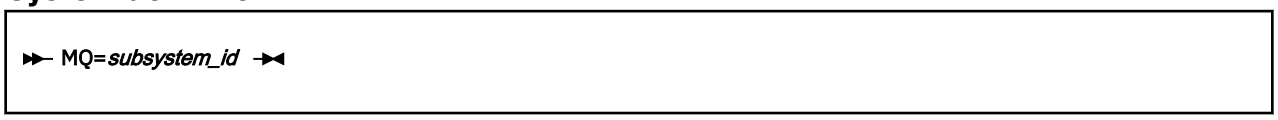
Syntax for defining MQ systems

To define MQ systems, insert **MQ** system definition statements in the SYSUT1 input data set to the system take-up utility.

System attributes

The system take-up utility does not support any MQ system attributes.

System definition



Example

Given the following input:

```
MQ=MYMA

IMAGE=MYZ1
MQ=MYMB
MQ=MYMC
```

```
IMAGE=MYZ2
MQ=MYMD
```

The system take-up utility defines the following MQ systems:

- MYMA, with no MVS image name, because there was no **IMAGE** statement preceding the corresponding **MQ** statement
- MYMB and MYMC, with MVS image name MYZ1
- MYMD, with MVS image name MYZ2

Syntax for defining MVS Image systems

To define MVS Image systems, insert **IMAGE** system definition statements in the SYSUT1 input data set to the system take-up utility. Optionally, you can precede the system definition statements with system attributes.

System attributes

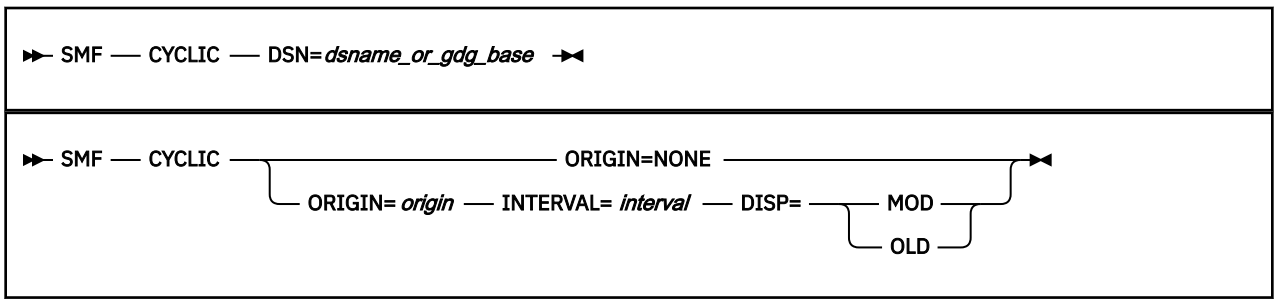
LOGSTREAM
The SMF log stream name.

RETPD
The SMF log stream retention period: a number of days in the range 0 - 65536, where 0 indicates that there is no expiry.

Automated file selection will select the log stream, ahead of cyclic SMF files, when the required time interval is within this retention period.

Set this retention period to less than or equal to the log stream's actual RETPD in the LOGR policy. One reason why you might set this retention period to less than the actual RETPD is to avoid a mass DFHSM recall of older offload data sets.

Cyclic SMF files:



You must specify system attributes for cyclic SMF files as a consecutive pair of `SMF CYCLIC DSN=...` and `SMF CYCLIC ORIGIN=...` input records.

There is one exception: a single `SMF CYCLIC DSN=` input record, with no value after the equal sign, causes subsequent **IMAGE** system definition statements to define MVS Image systems with no cyclic SMF files. If you specify `DUPLICATE=UPDATE` in the SYSIN data set, then a single `SMF CYCLIC DSN=` input record in SYSUT1, with no value after the equal sign, causes subsequent **IMAGE** system definition statements for existing systems to *delete* any cyclic SMF file attributes from those existing systems.

DSN
The data set name or the GDG (generation data group) base name of the SMF file. Cyclic SMF files are typically GDGs.

You can use the following symbolic variables in a data set name:

&YYYY
4-digit year

&YY
2-digit year (20*yy*)

&MM
Month (01 - 12)

&DD
Day of the month (01 - 31)

&DDD
Day of the year (001 - 366)

Examples:

```
SMF.MVS1.DAY
CICSPROD.SMF.WEEK
CICSPROD.SMF.D&YY&MM&DD
CICSPROD.SMF.D&YY.&MM.&DD
```

You can optionally terminate a variable name with a period. This period will not appear in the resolved data set name (the last two examples resolve to the same name). If you want a period to appear after a variable value in the resolved name, insert a second period:

```
CICSPROD.SMF.Y&YYYY..D&DDD
```

If you use symbolic variables:

- In the **ORIGIN** parameter, use asterisks to represent the digits of the origin date that are determined by symbolic variables.
- The origin date and the interval must be compatible with the symbolic variables. For example, if you use the variable &DDD, then the origin date must be in Julian format.

ORIGIN
The starting point of a new interval in the cycle, defining when the next (or first) SMF file in the cycle is created. Allowed values:

DAY
A new cycle starts every day, defining a daily cycle.

day_of_the_week
A new cycle starts on the specified day every week, defining the start of weekly cycle. Allowed values are the seven days of the week: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, and SUNDAY.

date
The first cycle starts on the specified date, and continues cycling forwards from that point in time. Cycles that start on a date are monthly, yearly or fixed number of days cycles. Allowed formats:

yyyy-mm-dd
*****-mm-dd*
yyyy-ddd
*****-ddd*

where **** (four consecutive asterisks) represents the current year, indicating that the cycle restarts from this point every year.

CDATE
A new cycle starts on the file creation date. The SMF file contains data starting from the date the file was created.

CDATE+*nnn*
A new cycle starts *nnn* number of days after the file creation date. That is, the SMF file contains data starting *nnn* number of days after the file was created.

For example, CDATE+1 specifies a file that is created before midnight to contain tomorrow's data.

CDATE-*nnn*
A new cycle starts *nnn* number of days before the file creation date. That is, the SMF file contains data starting *nnn* number of days before the file was created.

For example, CDATE-5 specifies a file that is created and then filled with data starting from five days ago.

NONE
No origin. Specify ORIGIN=NONE when you want to explicitly select a particular SMF file for reporting, rather than Transaction Analysis Workbench selecting appropriate SMF files for a requested reporting period.

If you specify ORIGIN=NONE, you cannot specify **INTERVAL** or **DISP**.

You cannot report on a mix of files with and without origins. If a system contains cyclic SMF file definitions with an origin of NONE and cyclic SMF file definitions with other origin values, then you must either exclude the files with an origin of NONE, or exclude all of the others.

INTERVAL
The time duration of one cycle of data. Allowed values:

- 0 (zero), to indicate an indefinite interval
- A number of days
- One of the following literal values: DAY, WEEK, MONTH, or YEAR

The allowed interval values can be restricted, depending on the origin:

ORIGIN	**INTERVAL**
DAY	Must be 1 (one day)
day_of_the_week	Must be WEEK
date	Can be any allowed value

DISP
Specifies whether the SMF file accumulates data (DISP=MOD) or does not accumulate data (DISP=OLD) over the interval.

DISP applies only to SMF files with one of the following origins:

- DAY
- *day_of_the_week*
- *date*

MOD
New cycles commence at the start of an interval, and continuously append new data to the SMF file until the end of the interval.

DISP=MOD cycles cover the current interval (up until today).

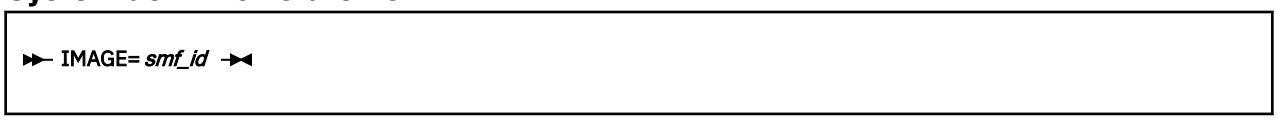
OLD
New cycles are created at the end of the interval.

DISP=OLD cycles do not cover the current interval. Other cyclic SMF files are required in this case.

To specify more than one cyclic SMF data set name or GDG base name for an MVS Image system, specify a consecutive block of **SMF CYCLIC** system attributes. You can use the system take-up utility to define up to 10 cyclic SMF data set names in an MVS Image system definition; up to 10 pairs of SMF CYCLIC DSN=… and SMF CYCLIC ORIGIN=… input records (a total of 20 records) in a consecutive block. A block of **SMF CYCLIC** system attributes is ended by a different system attribute or a system definition statement. A block is not broken by a blank line or a comment. A new block of **SMF CYCLIC** system attributes replaces any previous block.

OPERLOG
Specifies that the MVS Image refers to an OPERLOG log stream, which has the fixed name SYSPLEX.OPERLOG.

System definition statement



CICS, DB2, and MQ system definitions can optionally refer to an MVS image.

When defining CICS, DB2, or MQ systems, the system take-up utility applies the MVS image name of the most recent **IMAGE** system definition statement, if any.

To define a CICS, DB2, or MQ system that does not refer to an MVS image, insert the system definition statement before the first **IMAGE** statement.

Examples

Given the following input:

```
SMF CYCLIC DSN=&IMAGE..SMF.DAY
SMF CYCLIC ORIGIN=DAY INTERVAL=1 DISP=OLD
SMF CYCLIC DSN=&IMAGE..SMF.WEEK
SMF CYCLIC ORIGIN=MONDAY INTERVAL=WEEK DISP=OLD

CICS=MYCICSA

IMAGE=MYZ1
CICS=MYCICSB

SMF LOGSTREAM=IFASMF.&IMAGE RETPD=14

IMAGE=MYZ2
CICS=MYCICSC
```

- The MVS Image definitions MYZ1 and MYZ2 will contain per-day and per-week cyclic SMF file details.
- MYZ2 will refer to the SMF log stream IFASMF.MYZ2 with a retention period of 14 days.
- CICS system definition MYCICSA will not refer to an MVS Image.
- MYCICSB will refer to MYZ1, and MYCICSC will refer to MYZ2.

Given the following input:

```
SMF CYCLIC DSN=&IMAGE..DAY
SMF CYCLIC ORIGIN=DAY INTERVAL=1 DISP=OLD
SMF CYCLIC DSN=&IMAGE..WEEK
SMF CYCLIC ORIGIN=MONDAY INTERVAL=WEEK DISP=OLD

IMAGE=MYZ1

SMF CYCLIC DSN=&IMAGE..DAY
SMF CYCLIC ORIGIN=DAY INTERVAL=1 DISP=OLD

IMAGE=MYZ2

SMF CYCLIC DSN=

IMAGE=MYZ3
```

- MYZ1 will, as in the previous example, contain per-day and per-week cyclic SMF file details.
- MYZ2 will contain only per-day cyclic SMF file details.
- MYZ3 will not contain any cyclic SMF file details.

To delete all cyclic SMF file details from an existing MVS Image system definition, specify DUPLICATE=UPDATE in the SYSIN input data set to the system take-up utility and, in the SYSUT1 input data set, specify a single SMF CYCLIC DSN= system attribute before the corresponding **IMAGE** system definition statement:

```
SMF CYCLIC DSN=
IMAGE=MYZ1
```

You can only use the system take-up utility to add, replace, or delete the *complete set* of cyclic SMF file details for an existing MVS Image system definition.

System take-up utility parameter: DUPLICATE

The system take-up utility can read an optional SYSIN data set that contains a single parameter, **DUPLICATE**. The **DUPLICATE** parameter controls what the utility does if the *key* of a system definition statement in the SYSUT1 input data set duplicates the key of an existing system definition.

In this context, *existing* system definitions include: preceding system definition statements in SYSUT1, and system definitions that existed in the repository before the utility ran.

By default, the utility ignores such duplicates in the input. Alternatively, the utility can update the attributes of existing system definitions.

Keys

The key of a system definition depends on the system type:

- MVS Image and IMS systems have a single-part key: image name or IMS subsystem ID
- CICS, DB2, and MQ system definitions have a two-part key:
  1. (Sub)system identifier: CICS applid, DB2 or MQ subsystem ID
  2. MVS image name, which can be blank

  Two CICS system definitions, two DB2 system definitions, or two MQ system definitions have duplicate keys if their subsystem identifiers match *and* their MVS image names match. As a result of the two-part key of these system types, you cannot use the system take-up utility to update the MVS image name in definitions of these system types.

Syntax

IGNORE
Ignore the system definition in the input; leave the existing definition unchanged. This is the default behavior.

UPDATE
Update the existing system definition.

DUPLICATE=UPDATE does not replace the entire existing system definition; it only *inserts, replaces, or deletes attributes* that are specified in the SYSUT1 input data set. Any attributes of the existing system definition that are not specified in SYSUT1 are unaffected by the take-up.

To update some attributes of existing systems without affecting other attributes, only specify in SYSUT1 the attributes that you want to update.

Examples

Example SYSUT1 snippets for use with DUPLICATE=UPDATE:

- Insert or replace an SMF log stream attribute in an existing MVS Image system definition:

```
SMF LOGSTREAM=IFASMF.&IMAGE RETPD=14
IMAGE=MYZ1
```

- Delete the RECON data set names from an existing IMS system definition:

```
IMS RECON1=
IMS RECON2=
IMS RECON3=
IMS=IMSA
```

Related reference

Syntax for defining systems

The SYSUT1 input data set to the system take-up utility can contain the following types of records: system attributes, system definition statements, comments, and blank lines.

Chapter 77. Automated file selection utility

The JCL statements to invoke the automated file selection utility, FUWFILES, depend on the type of log file that you want to select.

If you want to use the Transaction Analysis Workbench ISPF dialog to generate JCL for the automated file selection utility, you must first create system definitions that contain the information the utility needs to locate the corresponding log files.

Related concepts

Automated file selection utility
The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

JCL for automated DB2 log file selection

To select DB2 log files, the Transaction Analysis Workbench automated file selection utility uses output from the DB2-supplied print log map utility, DSNJU004.

If you use the Transaction Analysis Workbench ISPF dialog to request automated file selection for DB2 log files, the dialog generates JCL that contains the following two steps.

Step 1: Run the DB2 print log map utility

The first step runs the DB2 print log map utility to generate a list of DB2 log files and their time spans, using information in the DB2 bootstrap data set (BSDS).

The JCL for this step depends on whether the DB2 system is in a data sharing group. You specify whether the DB2 system is in a data sharing group when you define the DB2 system to Transaction Analysis Workbench.

The following JCL listings include the JOB statement that precedes the first step. When generating this JCL, the Transaction Analysis Workbench ISPF dialog uses the value of the **Job Statement Information** field under option 0.1 **Workbench Personal Settings**.

If the DB2 system is in a data sharing group, this step consists of the following JCL, where the SYSIN control statement MEMBER identifies the DB2 system:

```
//UIDFUW   JOB NOTIFY=&SYSUID
//DSNJU004 EXEC PGM=DSNJU004
//STEPLIB  DD DISP=SHR,DSN=<DB2 SDSNLOAD library>
//SYSPRINT DD DISP=(NEW,PASS),DSN=&&DB2,
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//GROUP    DD DISP=SHR,DSN=<DB2 bootstrap data set>
//SYSIN    DD *
  MEMBER (<DB2 system ID>)
/*
```

Figure 142. JCL for automated file selection of DB2 logs: job step 1 of 2, for a DB2 system in a data sharing group

If the DB2 system is not in a data sharing group, this step consists of the following JCL (note the single SYSUT1 DD statement instead of the GROUP and SYSIN DD statements in the previous example):

```
//UIDFUW   JOB NOTIFY=&SYSUID
//DSNJU004 EXEC PGM=DSNJU004
//STEPLIB  DD DISP=SHR,DSN=<DB2 HLQ>.SDSNLOAD
//SYSPRINT DD DISP=(NEW,PASS),DSN=&&DB2,
//SYSUT1   DD DISP=SHR,DSN=<DB2 bootstrap data set>
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Figure 143. JCL for automated file selection of DB2 logs: job step 1 of 2, for a DB2 system not in a data sharing group

Here is an example of the output from the DB2 print log map utility:

```
000AA94BB000     000AAB67AFF     2008.002  21:25  DSN=DB2P.ARCHLOG1.A0001645
2008.002  05:29:00.0  2008.002  12:17:07.4        PASSWORD=(NULL) VOL=DATA08 UNIT=SYSALLDA
                                                  CATALOGUED
```

Figure 144. Automated file selection of DB2 logs: list of files generated by the DB2 print log map utility

For more information on the DB2 print log map utility, see the DB2 documentation.

Step 2: Run the Transaction Analysis Workbench automated file selection utility

The second step uses the output from the previous step as input to the Transaction Analysis Workbench automated file selection utility.

The automated file selection utility compares the time spans of the DB2 log files in the output from the previous step with the time span specified by the **FROM** and **TO** commands in this step. If the time span of a DB2 log file falls either partially or completely within the time span, the utility adds that file to the list of files for the specified problem session, stored in the session repository.

This step consists of the following JCL:

```
//FUWFILES EXEC PGM=FUWFILES,COND=(0,NE,DSNJU004)
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//FUWPROBR DD DISP=SHR,DSN=<session repository>
//FUWPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DB2LOGIN DD DISP=(OLD,DELETE),DSN=&&DB2OUT
//FUWWRK1  DD UNIT=SYSDA,SPACE=(TRK,(1,1,1))
//FUWWRK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN    DD UNIT=SYSDA,SPACE=(TRK,(3,1))
//FUWPARM  DD *
ZONE=<time zone>
PROBLEM=<session key>
FROM=(2010-06-14,12:00:00.00)
  TO=(2010-06-14,15:00:00.00)
DB2ID=<DB2 system ID>
```

Figure 145. JCL for automated file selection of DB2 logs: job step 2 of 2

The JCL for this step consists of statements with the following names:

FUWFILES
Runs the Transaction Analysis Workbench automated file selection utility, on the condition that the previous step completed successfully.

STEPLIB
Specifies the library containing the Transaction Analysis Workbench executable load modules, including the automated file selection utility program, FUWFILES.

If these modules reside in the system LNKLST, you do not need to specify this library.

To generate this JCL statement, the Transaction Analysis Workbench ISPF dialog uses the value of the **Workbench Load Library** field under option 0.1 **Workbench Personal Settings**.

FUWPROBR
Specifies the data set name of a Transaction Analysis Workbench session repository. This session repository must contain the problem session whose list of files you want to update.

DB2LOGIN
Specifies the output data set that the DB2 print log map utility created in the previous job step.

FUWWRK1, FUWWRK2, SYSIN
Specify work file data sets.

FUWPARM
Specifies input parameters for the automated file selection utility:

ZONE
The time zone of the system that create the log files. If you are running the automated file selection utility on the same system that created the log files, omit this parameter.

LOCAL
(Default.) Specifies that the log files were created on a system with the same time zone setting as the system on which you are running the automated file selection utility.

GMT
Specifies that the log files were created on a system whose time zone is set to GMT.

+hh*:*mm* or *-hh*:*mm
Specifies that the log files were created on a system whose time zone is set to *hh* hours and *mm* minutes east (+) or west (-) of GMT. For example, specify -08:00 for US Pacific time and +10:00 for Sydney Australia time.

PROBLEM
The key of the session whose list of files you want to update. This session must already exist in the session repository specified by ddname FUWPROBR.

FROM, TO
Specify when the problem occurred. The automated file selection utility selects the log files that are relevant to this period.

For details of the syntax of these parameters, see "START, STOP (FROM, TO) commands" on page 497.

Performance consideration: This is a useful way to reduce the amount of data processed by Transaction Analysis Workbench.

DB2ID
The identifier of the DB2 system for which you are performing automated file selection.

FUWPRINT
Specifies the data set to which the automated file selection utility write messages and other run-time information. You should check this data set for error messages. For an explanation of these error messages, see Chapter 68, "Messages," on page 325.

Related tasks

Tutorial: Automating selection of DB2 logs

This tutorial shows you how to define a DB2 system to Transaction Analysis Workbench, and then use that system definition to locate the related DB2 log files for a particular time interval.

JCL for automated IMS log file selection (using DBRC)

To select IMS log files, the Transaction Analysis Workbench automated file selection utility uses the IMS database recovery (DBRC) API to read RECON data sets.

```
//FUWFILES EXEC PGM=FUWFILES,PARM='IMSPLEX=<name>,DBRCGRP=<group>'
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//FUWPROBR DD DISP=SHR,DSN=<session repository>
//FUWPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//FUWWRK1  DD UNIT=SYSDA,SPACE=(TRK,(1,1,1))
//FUWWRK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN    DD UNIT=SYSDA,SPACE=(TRK,(3,1))
//FUWPARM  DD *
ZONE=<time zone>
PROBLEM=<session key>
FROM=(2010-06-14,12:00:00.00)
  TO=(2010-06-14,15:00:00.00)
IMSID=<IMS subsystem ID>
  VRM=<IMS release>
  SSID=<DBRC SSID>
  CAT=YES|NO
  OLDS=YES|NO
  SLDS=PRI|SEC
  MDA=<IMS MDA data set>
  RESLIB=<IMS RESLIB data set>
  RECON1=<IMS RECON data set 1>
  RECON2=<IMS RECON data set 2>
  RECON3=<IMS RECON data set 3>
```

Figure 146. JCL for automated file selection of IMS logs

JCL statements

FUWFILES
Runs the automated file selection utility program.

If the RECON data sets that you want to use belong to a running IMSplex:

- Specify the name of the IMSplex in the **PARM** parameter.
- The structured call interface (SCI) address space must be running on the system on which you want to run the automated file selection utility.

If the RECON data sets are used by more than one IMSplex, also specify the DBRC sharing group ID (DBRCGRP) in the **PARM** parameter.

STEPLIB
Specifies the library containing the Transaction Analysis Workbench executable load modules, including the automated file selection utility program, FUWFILES.

If these modules reside in the system LNKLST, you do not need to specify this library.

To generate this JCL statement, the Transaction Analysis Workbench ISPF dialog uses the value of the **Workbench Load Library** field under option 0.1 **Workbench Personal Settings**.

FUWPROBR
Specifies the data set name of a Transaction Analysis Workbench session repository. This session repository must contain the problem session whose list of files you want to update.

FUWPRINT
Specifies the data set to which the automated file selection utility write messages and other run-time information. You should check this data set for error messages. For an explanation of these error messages, see Chapter 68, “Messages,” on page 325.

FUWWRK1, FUWWRK2, SYSIN
Specify work file data sets.

FUWPARM
Specifies input parameters for the automated file selection utility:

ZONE
The time zone of the system that create the log files. If you are running the automated file selection utility on the same system that created the log files, omit this parameter.

LOCAL
(Default.) Specifies that the log files were created on a system with the same time zone setting as the system on which you are running the automated file selection utility.

GMT
Specifies that the log files were created on a system whose time zone is set to GMT.

+hh*:*mm* or *-hh*:*mm
Specifies that the log files were created on a system whose time zone is set to *hh* hours and *mm* minutes east (+) or west (-) of GMT. For example, specify -08:00 for US Pacific time and +10:00 for Sydney Australia time.

PROBLEM
The key of the session whose list of files you want to update. This session must already exist in the session repository specified by ddname FUWPROBR.

FROM, TO
Specify when the problem occurred. The automated file selection utility selects the log files that are relevant to this period.

For details of the syntax of these parameters, see "START, STOP (FROM, TO) commands" on page 497.

Performance consideration: This is a useful way to reduce the amount of data processed by Transaction Analysis Workbench.

IMSID
The IMS subsystem identifier (IMSID).

For sysplex processing, you can specify multiple subsystems. For example:

```
IMSID=IMS1
  CAT=YES
  VRM=154
  RECON1=IMS1.RECON1
  RECON2=IMS1.RECON2
  RECON3=IMS1.RECON3
IMSID=IMS2
  CAT=YES
  VRM=154
  RECON1=IMS2.RECON1
  RECON2=IMS2.RECON2
  RECON3=IMS2.RECON3
```

For each subsystem, specify the following parameters immediately after the IMSID specification:

SSID
The DBRC subsystem ID. This is required only if it is longer than 4 characters or different to the IMS subsystem ID.

VRM
The IMS version of the IMS subsystem. For example, V15.4 is 154; V15.4 is 154.

CAT=YES|NO
Whether the log data sets are cataloged.

OLDS=NO|YES
Whether to use OLDS data sets if SLDS files are not available for the requested time interval.

SLDS=SEC|PRI
Specify SEC to use secondary SLDS data sets if they are available, instead of primary SLDS data sets. Specify PRI to only use primary SLDS data sets.

RESLIB
The RESLIB data set for this IMS subsystem. The DBRC API routine must reside in this data set. If the modules reside in the LNKLST concatenation, you can omit this parameter.

The RESLIB data set can also be used to determine the IMS version. If you do not specify the VRM explicitly, Transaction Analysis Workbench sets the IMS version from information in module DFSVC000.

RECON1,2,3
The two or three RECON data sets used by this IMS subsystem. DBRC requires at least two RECON data sets to be specified. Transaction Analysis Workbench calls the DBRC API to determine the relevant log files from the RECON data sets. These are ignored if the MDA data set is specified.

MDA
The MVS dynamic allocation (MDA) data set for this IMS subsystem. Transaction Analysis Workbench uses the MDA data set to determine the RECON data set names, and in turn DBRC determines the relevant log files from the RECON data sets. If you specify the MDA data set, you do not need to explicitly specify the RECON data set names.

Related tasks

Tutorial: Automating selection of IMS logs
This tutorial shows you how to define an IMS system to Transaction Analysis Workbench, and then use that system definition to locate the related IMS log files (SLDS or OLDS) for a particular time interval.

JCL for automated IMS Connect Extensions journal selection

To use the automated file selection utility to select IMS Connect Extensions journals, you need to specify the location of the IMS Connect Extensions definition repository, and an IMS Connect system ID.

```
//FUWFILES EXEC PGM=FUWFILES
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//CEXDEF   DD <definition repository>,DISP=SHR
//FUWPROBR DD DISP=SHR,DSN=<session repository>
//FUWPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//FUWWRK1  DD UNIT=SYSDA,SPACE=(TRK,(1,1,1))
//FUWWRK2  DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN    DD UNIT=SYSDA,SPACE=(TRK,(3,1))
//FUWPARM  DD *
PROBLEM=<session key>
FROM=(2010-06-14,12:00:00.00)
  TO=(2010-06-14,15:00:00.00)
HWSID=<IMS Connect system ID>
```

Figure 147. JCL for automated file selection of IMS Connect Extensions journals

JCL statements

FUWFILES
Runs the Transaction Analysis Workbench automated file selection utility.

STEPLIB
Specifies the library containing the Transaction Analysis Workbench executable load modules, including the automated file selection utility program, FUWFILES.

If these modules reside in the system LNKLST, you do not need to specify this library.

To generate this JCL statement, the Transaction Analysis Workbench ISPF dialog uses the value of the **Workbench Load Library** field under option 0.1 **Workbench Personal Settings**.

CEXDEF
Specifies the data set name of the IMS Connect Extensions definition repository.

FUWPROBR
Specifies the data set name of a Transaction Analysis Workbench session repository. This session repository must contain the problem session whose list of files you want to update.

FUWWRK1, FUWWRK2, SYSIN
Specify work file data sets.

FUWPRINT
Specifies the data set to which the automated file selection utility write messages and other run-time information. You should check this data set for error messages. For an explanation of these error messages, see Chapter 68, “Messages,” on page 325.

FUWPARM
Specifies input parameters for the automated file selection utility. For IMS Connect Extensions journal selection, specify the following parameters:

ZONE
The time zone of the system that create the log files. If you are running the automated file selection utility on the same system that created the log files, omit this parameter.

LOCAL
(Default.) Specifies that the log files were created on a system with the same time zone setting as the system on which you are running the automated file selection utility.

GMT
Specifies that the log files were created on a system whose time zone is set to GMT.

+hh:mm* or *-hh:mm
Specifies that the log files were created on a system whose time zone is set to *hh* hours and *mm* minutes east (+) or west (-) of GMT. For example, specify -08:00 for US Pacific time and +10:00 for Sydney Australia time.

PROBLEM
The key of the session whose list of files you want to update. This session must already exist in the session repository specified by ddname FUWPROBR.

FROM, TO
Specify when the problem occurred. The automated file selection utility selects the log files that are relevant to this period.

For details of the syntax of these parameters, see “START, STOP (FROM, TO) commands” on page 497.

Performance consideration: This is a useful way to reduce the amount of data processed by Transaction Analysis Workbench.

HWSID
The ID of the IMS Connect system. If you specify HWSID, then you need to specify the data set name of the IMS Connect Extensions definition repository in the CEXDEF DD statement.

JCL for automated SMF and DB2 near-term history file selection

To use the automated file selection utility to select SMF files, you need to specify the name of a system definition that refers to SMF files. The following types of system definition can refer to SMF files: CICS, DB2, MVS image, and IBM MQ. To select DB2 near-term history files, you need to specify the name of a DB2 system definition that refers to a Tivoli OMEGAMON XE for DB2 Performance Expert near-term history file (SEQDATASET).

```
//UIDFUW   JOB NOTIFY=&SYSUID
//FUWFILES EXEC PGM=FUWFILES
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//FUWPROBR DD DISP=SHR,DSN=<session repository>
//FUWSYSDF DD  DISP=SHR,DSN=<system definition repository>
//FUWPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//FUWPARM  DD *
PROBLEM=<session key>
FROM=(2010-06-14,12:00:00.00)
  TO=(2010-06-14,15:00:00.00)
SMF
  CICS=<CICS system name>
  DB2=<DB2 system name>
  IMAGE=<MVS image system name>
  MQ=<IBM MQ system name>
/*
```

Figure 148. JCL for automated file selection of SMF files and DB2 near-term history files

JCL statements

FUWFILES
Runs the Transaction Analysis Workbench automated file selection utility.

STEPLIB
Specifies the library containing the Transaction Analysis Workbench executable load modules, including the automated file selection utility program, FUWFILES.

If these modules reside in the system LNKLST, you do not need to specify this library.

To generate this JCL statement, the Transaction Analysis Workbench ISPF dialog uses the value of the **Workbench Load Library** field under option 0.1 **Workbench Personal Settings**.

FUWPROBR
Specifies the data set name of a Transaction Analysis Workbench session repository. This session repository must contain the problem session whose list of files you want to update.

FUWSYSDF
System definition repository that contains the system definitions referred to by the FUWPARM data set.

FUWPRINT
Specifies the data set to which the automated file selection utility write messages and other run-time information. You should check this data set for error messages. For an explanation of these error messages, see Chapter 68, “Messages,” on page 325.

FUWPARM
Specifies input parameters for the automated file selection utility. For SMF and DB2 near-term history file selection, specify the following parameters:

ZONE
The time zone of the system that create the log files. If you are running the automated file selection utility on the same system that created the log files, omit this parameter.

LOCAL
(Default.) Specifies that the log files were created on a system with the same time zone setting as the system on which you are running the automated file selection utility.

GMT
Specifies that the log files were created on a system whose time zone is set to GMT.

+hh*:*mm* or *-hh*:*mm
Specifies that the log files were created on a system whose time zone is set to *hh* hours and *mm* minutes east (+) or west (-) of GMT. For example, specify -08:00 for US Pacific time and +10:00 for Sydney Australia time.

PROBLEM
The key of the session whose list of files you want to update. This session must already exist in the session repository specified by ddname FUWPROBR.

FROM, TO
Specify when the problem occurred. The automated file selection utility selects the log files that are relevant to this period.

For details of the syntax of these parameters, see "START, STOP (FROM, TO) commands" on page 497.

Performance consideration: This is a useful way to reduce the amount of data processed by Transaction Analysis Workbench.

SMF
Requests SMF file and, for DB2 systems, DB2 near-term history file selection for the specified systems. You can specify multiple systems in any order. The systems must be defined in the system definition repository.

If a DB2 system definition specifies a near-term history file pattern, then the automated file selection utility selects near-term history files in preference to SMF files. If a DB2 system definition does not specify a near-term history file pattern, or the near-term history files do not cover the specified time interval, the utility reverts to selecting SMF files.

Chapter 78. Task scheduler

The task scheduler manages the execution of some or all of the tasks in a session workflow, allowing you to perform a workflow without user intervention. The scheduler runs as a batch job, submitting the tasks in order, stopping if a task fails.

```
//UIDFUW   JOB NOTIFY=&SYSUID
//S1     EXEC PGM=FUWBATCH
//STEPLIB  DD DISP=SHR,DSN=<FUW HLQ>.SFUWLINK
//SYSPRINT DD SYSOUT=*
//FUWPROBR DD DISP=SHR,DSN=<session repository>
//FUWVARS  DD DISP=SHR,DSN=<workflow task JCL substitution variables member>
//SYSIN    DD *
REQUEST=SCHEDULE
SESSION=00000023
TASK=CCDEDB224967EF86 MAXRC=0
TASK=CCE8AB34FA57CA84 MAXRC=0
TASK=CCF0688187143706 MAXRC=0
/*
//JOBCARD  DD DATA,DLM=$$
//UIDFUW   JOB NOTIFY=&SYSUID
$$
/*
```

Figure 149. JCL for the task scheduler

JCL statements

S1
Runs the task scheduler program, FUWBATCH, which is the same program as the Transaction Analysis Workbench report and extract utility.

FUWPROBR
Specifies the data set name of a Transaction Analysis Workbench session repository. This session repository must contain the problem session whose list of files you want to update.

FUWVARS
Defines values for *user profile* workflow task JCL substitution variables. FUWVARS can refer to the same member that you use for the plug-in.

SYSIN
Contains task scheduler commands:

REQUEST=SCHEDULE
Runs the task scheduler.

SESSION
Specifies the session that contains the workflow tasks that you want to run.

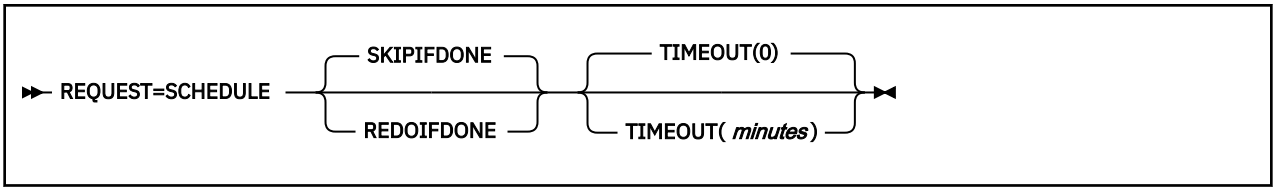
TASK
Specifies a workflow task to run.

A SYSIN data set can specify only one request, for one session, for one or more tasks in that session.

JOBCARD
Specifies the job statement that the task scheduler uses for each of the task jobs that it submits.

REQUEST=SCHEDULE commmand

The **REQUEST=SCHEDULE** command runs the task scheduler.



SKIPIFDONE
Skip tasks that have any of the following status values:

DONE
CC *nnnn*, where *nnnn* is less than or equal to the **MAXRC** parameter for the task
IGNORE

REDOIFDONE
Always perform tasks, except if their status is IGNORE.

TIMEOUT
If the task does not complete within the specified integer number of minutes, stop scheduling: do not submit subsequent tasks.

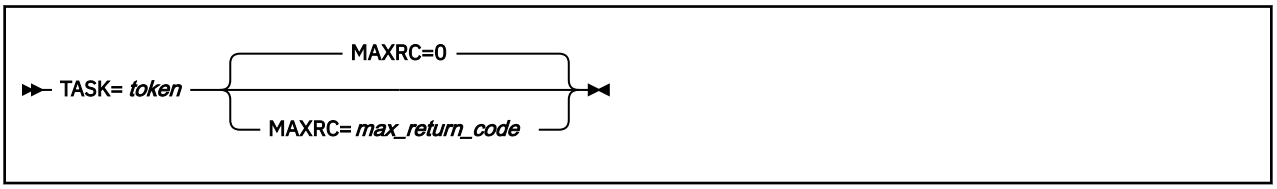
The default, **TIMEOUT(0)**, means no time limit: wait indefinitely for each task to complete.

SESSION command

The **SESSION** command specifies the session that contains the workflow tasks that you want to run.

TASK command

The **TASK** command specifies a workflow task to run. The task scheduler runs tasks in the order of the **TASK** commands in the SYSIN data set.



token
A string of 16 hexadecimal digits that identifies the task.

Tip: To display these tokens in the Transaction Analysis Workbench ISPF dialog, go to the session **Tasks** panel, and then press the Right function key (F11) to scroll the **Token** column into view.

MAXRC
Specifies the maximum allowable completion code from the task's batch job. If the job for a task fails to submit or its completion code exceeds **MAXRC**, then scheduling stops and remaining tasks are not performed. The default is `MAXRC=0`; each task job must end with completion code zero (`CC 0000`).

Related tasks

Performing workflow tasks
You can perform tasks in a session workflow either by selecting individual tasks or by scheduling multiple tasks to run in sequence.

Related reference

Workflow task JCL substitution variables
The JCL for workflow tasks can contain substitution variables that are specific to Transaction Analysis Workbench.

FUWVARS: Workflow task JCL substitution variables

The FUWVARS member of the Transaction Analysis Workbench control library defines substitution variables that the plug-in uses to resolve workflow task JCL.

Chapter 79. REXX API for formatting and analyzing logs

Transaction Analysis Workbench provides a REXX application programming interface (API) that you can use to format and analyze logs.

The REXX API allows you to analyze any of the file types that Transaction Analysis Workbench supports. You can display the output of your REXX exec in the Transaction Analysis Workbench ISPF dialog or write it to a data set.

You can run a REXX exec in Transaction Analysis Workbench in the following situations:

In the ISPF dialog
You can specify a REXX exec name in the following locations in the Transaction Analysis Workbench ISPF dialog. The exec must be in one of the libraries in your SYSEXEC concatenation.

In a filter
You can specify a REXX exec name in a filter, next to a particular log code. The exec runs in the following situations:

- When a user browses a record that matches the filter log code and condition
- When a user zooms on a field in a matching record

If you use a filter in a batch job, any REXX execs named in the filter are ignored. To run a REXX exec in batch, use the REXX command of the report and extract utility.

On the Process Log Files panel (your personal ad hoc list of log files)
You can enter a REXX exec name as a line action next to a log file on the **Process Log Files** panel: on the Transaction Analysis Workbench Primary Option Menu, select option 4 **Process**.

Browsing logs
You can enter a REXX exec name on the command line when browsing logs, browsing an individual log record, or zoomed on a field in a log record. The exec runs immediately.

In a batch job
You can specify the name of a REXX exec using the **REXX** command of the report and extract utility. Instead of creating a report or an extract for the specified log files, the utility runs the REXX exec.

Related concepts

Filters: Log record selection criteria
Filters enable you to select the log records that you want to analyze and exclude others. For example, you can define a filter to select only those records of a particular log type and code that are associated with a particular transaction code or user ID. You can use a filter when browsing logs in the ISPF dialog or when writing JCL for the report and extract utility.

Related tasks

Defining filters
To select only the log records that are of interest to you, define and then use a filter. A filter specifies which log record codes to select or exclude and, optionally, more detailed conditions based on field values.

Related reference

REXX command

Runs a REXX exec that enables you to read sequentially forward through the selected log records.

Parts of a typical REXX exec

Typically, REXX execs for Transaction Analysis Workbench contain parts that detect how the exec was invoked (its *invocation mode*), read records and fields, and output data to a panel or a data set.

The following example shows the structure of a typical ALZEXEC REXX exec:

```
/*ALZEXEC REXX example */
  address ALZEXEC 1
  address MVS 'NEWSTACK'
  select 2
    when ALZ.MODE = 'F' then do /* Actions for file mode */
      "READ STEM(REC.) CODE(01)" /* Find next 01 record */ 3
      if REC.0 = 0 then do /* Actions if no matches found */ 4
        queue "No 01 records found"
        ⋮
    else
      /* Actions when record found */
    end
    when ALZ.MODE = 'R' then do /* Actions for record mode */
      "FETCH STEM(FLD.) FIELD(*)" /* Fetch fields in record */ 5
      ⋮
    end
    when ALZ.MODE = 'Z' then do /* Actions for zoom mode */
      if  value(ALZ.FIELD) = "MYVALUE" /* Actions for field value */ 6
     ⋮
    end
  queue "Execution completed"
  "DISPLAY REPLACE" /* Show queued values in place of record */ 7
  address MVS 'DELSTACK' /* Clear the stack */
  exit 8
```

1

The instruction `address ALZEXEC` sets the host command environment to the environment of Transaction Analysis Workbench.

Note: This statement is not required when your REXX is initialized from the Transaction Analysis Workbench ISPF dialog because the ALZEXEC environment is already set for you. While not required, it is recommended.

2

The **`select`** instruction evaluates whether the user invoking the exec is browsing a file (F), a record within a file (R), or zooming on a field within a record (Z). Depending on how a user invokes the exec, you may want to initiate different actions. For details, see “Detecting how the exec was invoked” on page 607.

3

In file mode, this exec reads the first record with the log code 01 and returns values to the stem `'REC.'`.

4

The **READ** command sets the stem (compound) variable to REC. The variable `REC.0` contains the number of records matching a filter or code.

5

In record mode, this exec fetches each field in the record.

6

In zoom mode, this exec evaluates the value inside the field.

7

The **DISPLAY** command outputs the content of the stack or stem variable. Depending on its parameters, it can output to a data set and the Transaction Analysis Workbench ISPF dialog.

8

The REXX exec clears the stack created at the start of the exec and exits. When the REXX exec exits, the output of the last **DISPLAY** command appears in the ISPF dialog.

Detecting how the exec was invoked

You can invoke REXX execs in Transaction Analysis Workbench while browsing a log file, a record within a file, or a field with the record. You can detect these modes in the exec and adjust the behavior of the exec accordingly.

The following table shows how an exec was invoked determines its *invocation mode*.

Table 33. REXX exec invocation modes

How the exec was invoked			**Invocation mode**
In the ISPF dialog,	by a filter condition,	when the user browses a record that matches the filter log code and condition	Record
		when the user browses a field in a matching record	Zoom (field)
	when the user enter the exec name as a line action on either: • The **Investigate** panel for a session (under option 1 **Sessions**) • The **Process** panel (option 4 **Process**)		File
	when the user enters the exec name on the command line of the log browser,	while browsing a log file	File
		while browsing a record	Record
		while zoomed on a field	Zoom (field)
In a batch job, by the REXX command of the report and extract utility			File

The value of the predefined variable ALZ.MODE specifies the invocation mode:

F
File

R
Record

Z
Zoom (field)

The value of the predefined variable ALZ.ENV specifies the *invocation environment*; whether the exec was invoked from batch or from the ISPF dialog:

B
Batch

I
ISPF

You can adjust the behavior of a REXX exec according to its invocation mode and environment. In particular, you can choose the appropriate method for presenting output from your REXX exec.

- In record and zoom (field) modes, you can use the **DISPLAY** command to display data from your REXX exec in either of the following ways:
  - Insert the data into the panel that the Transaction Analysis Workbench ISPF dialog would normally display (a record browse panel or a field zoom pop-up window, depending on the invocation mode)
  - Display the data in an ISPF Edit, View, or Browse panel.

If you set the exec exit return code to 0, the normal Transaction Analysis Workbench ISPF dialog panel displays after the user exits the ISPF Edit, View, or Browse panel. To suppress the display of the normal panel, set the exec exit return code to 4.

- In file mode from the ISPF dialog, there is no Transaction Analysis Workbench panel to customize, to display output from your REXX exec, use the **DISPLAY** command to display the data in an ISPF Edit, View, or Browse panel.
- From a batch job (always file mode):
  - You cannot use the direct-access form of the **READ** command.
  - You cannot read backwards (**READ PREV**).
  - You cannot use the DISPLAY command. Instead, use REXX instructions such as **SAY** to write to SYSTSPRT or the TSO/E REXX command **EXECIO** to write to another file.

For example, the following figure shows an exec that counts the number of records with a given log code. If the user invokes your exec while browsing the log file, not having selected a record, your exec prompts the user to enter which log code to count. If the user is browsing a record within the log file, on the other hand, your exec gets the log code of that record and counts all other records with the same code.

```
⋮
select
  when ALZ.MODE = 'F' then do /* File mode */
    say "Enter the log code you want to count:"
    pull usercode .
    "READ COUNT(9999) STEM(REC.) CODE("usercode")"
  end
  when ALZ.MODE = 'R' then do /* Record mode */
    "READ COUNT(9999) STEM(REC.) CODE("ALZ.LOGCODE")"
⋮
```

Figure 150. Example of using the invocation mode in a REXX exec

Reading a record

Use the **READ** command to position on a record, find a record, or fetch global fields.

- Position on the first record in the file:

```
"READ RRN(1) STEM(stem.)"
```

- Position on the next record:

```
"READ STEM(stem.)"
```

- Position on the next record with a given log code:

```
"READ STEM(stem.) CODE(01)"
```

- Position on the record from which the exec was initialized:

```
"READ RRN("ALZ.RRN") STEM(stem.)"
```

- Position on the record that returned data to `stem1`:

```
"READ RRN("stem1.RRN") STEM(stem.)"
```

- Position on the end of the file:

```
"READ RRN(99999999) STEM(stem.)"
```

- Position on the last record in the file with a given log code:

```
"READ PREV STEM(stem.) CODE(01)"
```

- Get global fields from the record from which the exec was initialized:

```
"READ RRN("ALZ.RRN") STEM(stem.) GLOBALS"
```

Fetching fields from a record

Use the **FETCH** command to read fields in the current record.

Fetch all fields

Fetch all fields in the currently positioned record:

```
"FETCH"
```

Test flag fields

Fetch a flag field and test its flag bits:

```
"FETCH FIELD(MSGFLAGS)" /*gets flag bits in field*/
 if MSGFFRST & MSGFLAST then do /*both flag bits on*/
    say "Message spans single record"
 end
 else do
    say "Message spans multiple records"
 end
```

Note: Flag bit values are returned as either:

0
: Flag bit off

1
: Flag bit on

Fetch fields from a particular segment

Fetch fields in from a repeating segment in a record. In the example, fetching all fields in the third SMB segment:

```
"FETCH POSITION(SMB,3) FIELD(*)"
```

Fetch fields from another record

Fetch fields from another record (for example, an 01 record) and return to the current record:

```
/* Positions on an 01 record*/
"READ CODE(01) STEM(stem.)
/* Retrieves 01 message text to MSG01.*/
"FETCH FIELD(MSGXSTXT) STEM(MSG01.)"
/* Returns to the current record */
"READ RRN("ALZ.RRN") STEM(stem.)"
```

Exiting from a REXX exec

If a REXX exec is invoked from the Transaction Analysis Workbench ISPF dialog, then, when the REXX exec exits, the ISPF dialog displays data from the REXX exec in a panel according to the options specified by the **DISPLAY** command.

- Accumulate information in the stack or stem variable first, and only then issue the **DISPLAY** command. The information is displayed on exit.
- If you are using a stack, it is recommended that you get a new stack with:

  ```
  address MVS 'NEWSTACK'
  ```

  and then, after issuing the **DISPLAY** command, delete the stack with:

  ```
  address MVS 'DELSTACK'
  ```

REXX API reference

The Transaction Analysis Workbench REXX API, also known as the ALZEXEC REXX environment, consists of several commands and predefined REXX variables.

At the start of a Transaction Analysis Workbench REXX exec, insert the following **address** instruction to set the host command environment:

```
address ALZEXEC
```

This **address** instruction is required only for REXX execs invoked from a batch job. REXX execs invoked from the Transaction Analysis Workbench ISPF dialog implicitly run in the ALZEXEC REXX environment, and so do not require this instruction. However, it is good practice to specify this instruction and explicitly identify the host command environment.

Predefined REXX variables

The ALZEXEC environment predefines variables that you can use in your REXX exec.

The following table describes the predefined variables, their values, and the invocation modes in which the variables are set.

Table 34. REXX variables set on initialization

Variable	Description	Possible values	Invocation modes
ALZ.ENV	The environment in which the exec was invoked.	**B** Batch **I** ISPF	All
ALZ.MODE	The mode in which the exec was invoked (*exec invocation mode*).	**F** File **R** Record **Z** Zoom (field)	All
ALZ.IMSVRM	The IMS release of the log file, if applicable.	4 digits. For example, 1540 represents IMS V15.4.	All
ALZ.FILTER	The filter name. Blank if there is no named filter.	0 - 8 characters	All
ALZ.RRN	The relative record number. The position of the record in the merged file.	8 decimal digits	All
ALZ.LOGCODE	The log code of the record.	2-byte hexadecimal code	All
ALZ.LOGTYPE	The log type of the record.	Up to 4 characters	All
ALZ.POSITION	IMS log records only: the position of the current segment in relation to other segments in the record. See POSITION.	Variable-length string formatted as: *segment_name*, *n*, *segment_name*, *n*, ...	Zoom
ALZ.SEGMENT	IMS log records only: the name of the segment.	Variable-length string	Zoom

Table 34. REXX variables set on initialization (continued)			
Variable	**Description**	**Possible values**	**Invocation modes**
ALZ.FIELD	The name of the field currently zoomed. To get the value stored in the field, use the REXX function call `value(ALZ.FIELD)`	Variable-length string: *field_name*	Zoom
field_name	A variable with the name of the zoomed field, holding the value of that field.	Variable-length string	Zoom

REXX API commands

The REXX API provides three commands: **READ**, for reading records; **FETCH**, for reading fields; and **DISPLAY**, for displaying data in the ISPF dialog.

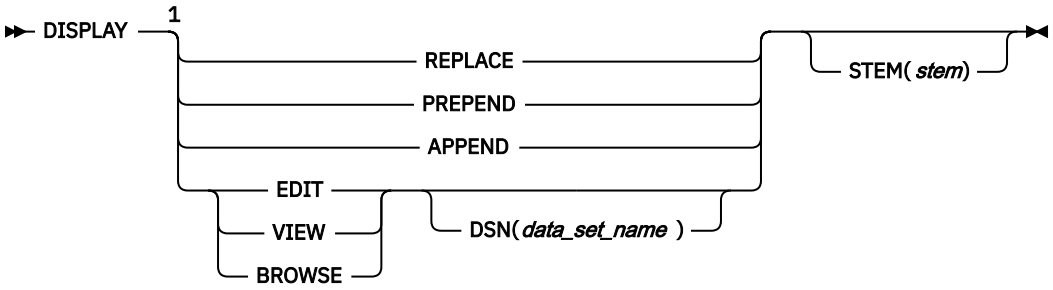
DISPLAY

Displays the contents of the stack or a stem variable in an ISPF panel, after the REXX exec has completed. You can choose to display the data on either: the panel that the Transaction Analysis Workbench ISPF dialog normally displays (that is, the record browse panel or the field zoom pop-up window, depending on how the exec was invoked); or an ISPF Edit, View, or Browse panel.

The **DISPLAY** command clears the stack.

You can only use the **DISPLAY** command when the REXX exec is invoked from the ISPF dialog. You cannot use the **DISPLAY** command when the REXX exec is invoked from a batch job.

Format



Notes:

[1] REPLACE, PREPEND, and APPEND are applicable only to the record and zoom (field) invocation modes.

Parameters

REPLACE | PREPEND | APPEND
Record and zoom (field) invocation modes only: Displays the data (from the stack or the specified stem variable) in the Transaction Analysis Workbench record browse panel or field zoom pop-up window, depending on the invocation mode of the exec.

REPLACE
The data is displayed instead of the normal contents of the panel.

PREPEND
The data precedes the normal contents of the panel.

APPEND
The data follows the normal contents of the panel.

Note:

- Do not use more than one **DISPLAY** command with **PREPEND**, **APPEND**, or **REPLACE**: if the exec is run from a filter, each **DISPLAY** command overwrites the output of the previous command.
- If you intend to use the exec in filters, then using **REPLACE**, **PREPEND**, or **APPEND** provides a more consistent experience to users than using **EDIT**, **VIEW**, or **BROWSE**.

EDIT | VIEW | BROWSE
Writes the data (from the stack or the specified stem variable) to a data set, and then displays the data in an ISPF Edit, View, or Browse panel.

Note:

- Record and zoom (field) invocation modes only: If the REXX exec exits with return code 0, the normal Transaction Analysis Workbench record browse panel or field zoom pop-up window (depending on the invocation mode of the exec) is displayed after the user exits the ISPF Edit, View, or Browse panel. To suppress the display of the normal Transaction Analysis Workbench panel, set the REXX exec return code to 4.
- If you use more than one **DISPLAY** command with **EDIT**, **VIEW**, or **BROWSE**, then each set of data (from the stack or the specified stem variable) is displayed in sequence, in its own ISPF panel.

STEM(*variable_name*)
Displays data from a stem variable, where

***variable_name*.0**
Holds the number of lines to display.

variable_name.n
Holds the contents of the *n*th line.

If you omit **STEM**, the **DISPLAY** command pulls data from the stack, clearing it in the process.

DSN(*data_set_name*)
The name of the data set to write the data to. The name must be specified in uppercase. Only valid with **EDIT**, **VIEW**, or **BROWSE**.

Return codes

After the DISPLAY command runs it sets the REXX special variable RC to one of the following return codes:

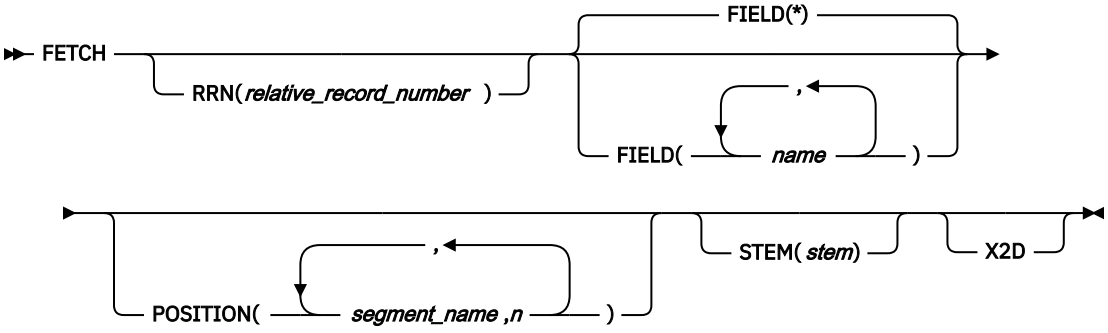
0
Successful.

20
Unrecoverable error. For example, a syntax error or an I/O error. Message in SYSTSPRT log.

FETCH

Fetches (reads) the contents of a field into a stem variable.

Format



Parameters

RRN(*relative_record_number*)
Retrieves fields from the specified record. Omit, to retrieve fields from the currently positioned record. You cannot use this parameter in batch.

FIELD(* | *name_list*)
Retrieves values for the fields you list, setting the values in variables with the names of the fields. If you do not specify the POSITION parameter fields in repeatable segments will be ignored (if FIELD(*) is specified) or generate an error condition if you try and fetch them directly.

POSITION(*segment_name*,*n*)
Sets the segment from which to retrieve field values. The variable specifies the nesting level and segment number within the record. Up to six levels of nesting can be specified. For example, if this is the structure of the record:

```
SegA
    SegB
    SegB
        <To position here>
    SegB
```

then this value for the POSITION parameter retrieves fields from the second SegB segment in the record:

```
"…POSITION(SegmentA,1,SegmentB,2)…"
```

Note:

- You must use the POSITION parameter to retrieve fields from repeatable segments.
- If in zoom mode, the parameter

  ```
  "…POSITION("ALZ.POSITION")…"
  ```

  Sets the position to the current segment.

STEM(*stem_variable_name*)
Specifies the name of the stem variable set by this function. All variables will have the form:

```
stem_variable_name.VARNAME
```

The *stem* parameter can be up to 176 characters.

X2D
Converts hexadecimal fields to decimal.

- Only fields that are 1 - 4 bytes long are converted to decimal.
- Flag fields are always returned as hexadecimal.
- Fields defined as integers are always returned as decimal.

Return codes

After the command runs it sets the REXX special variable RC to one of the following return codes:

0
Successful: field values fetched.

8
Match not found:

Record not found
The variable ALZ.ERRMSG is set to the message FUW8000E.

Segment not found
The variable ALZ.ERRMSG is set to the message FUW8001E.

Field not found
The variable ALZ.ERRMSG is set to the message FUW8002E.

Field values ambiguous
The variable ALZ.ERRMSG is set to the message FUW8003E.

20
Unrecoverable error. For example, a syntax error or an I/O error. Message in SYSTSPRT log.

READ

Reads the contents of a record into a stem variable.

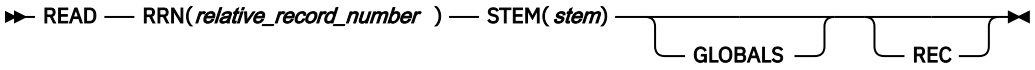
The **READ** command has the following two forms:

Direct access
Reads and positions on a record with a given relative record number (RRN).

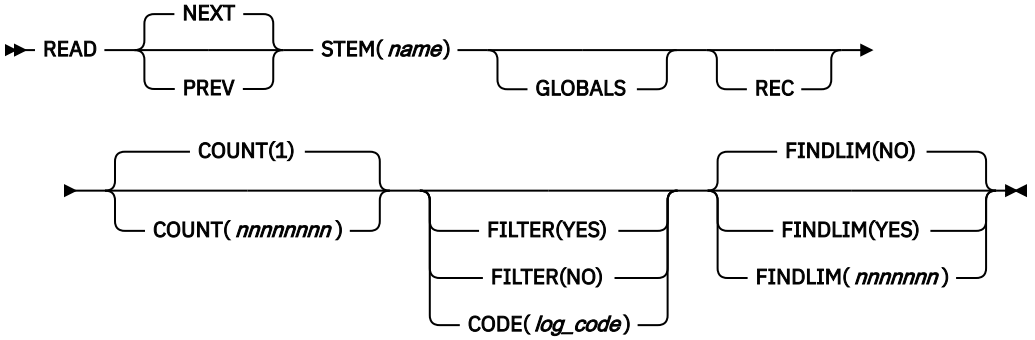
Sequential access
Reads and positions on the next or previous record; optionally, matching criteria.

Format

Direct access



Sequential access



Parameters

RRN(*relative_record_number*)
(Direct access only) Sets the record that is read and positions on that record. Subsequent sequential access **READ** commands will operate relative to this RRN.

If the record does not exist, then return code 8 is set and the position is set to the start of the file, `RRN(0)`.

NEXT | PREV
(Sequential access only) Selects the next (**NEXT**) or previous (**PREV**) record relative to the current position.

STEM(*stem*)
Specifies the name of the stem variable set by this function. All variables will have the form:

```
stem.variable
```

The *stem* parameter can be up to 176 characters. For the names and descriptions of variables returned, see "Results" on page 616.

GLOBALS
Sets variables with the global fields. The variables are in this format:

```
stem.global_field_name
```

Where:

stem
The name assigned by the **STEM** parameter.

global_field_name
The name of the global field. For example:

```
stem.IMSID
```

REC
Returns the data of the entire record in hexadecimal format. The **READ** command places the data in this variable:

```
stem.REC
```

Where *stem* is the name assigned in the **STEM** parameter.

COUNT(*nnnnnnnn*)
(Sequential access only) Specifies the number of records to read. The default count is one. If you specify a number greater than 1, then each variable this command returns will be suffixed by an index number, in the form:

```
stem.varname.i
```

where *i* is the record index number.

FILTER(YES | NO)
(Sequential access only) One of the following:

YES
Activates the filter.

NO
Deactivates the filter.

If you do not specify the **FILTER** parameter, the command uses the existing batch or ISPF filter settings.

CODE(*nn* | *nnnn*)
(Sequential access only) Specifies a log code number for filtering. The **READ** command returns only those records with the selected log code. If you specify a code it will override any other filtering option.

FINDLIM(NO | YES | *nnnnnnn*)
(Sequential access only) Specifies the maximum number of records that the command can parse to find a matching record. The options are:

NO
If required, parses until end of file (for **NEXT**), or start of file (for **PREV**).

YES
Uses the **FINDLIM** value set through the program ISPF dialog.

nnnnnnn
Parses up to *nnnnnnn* records. Specify 1000 - 9999999 records.

If the limit is reached, the variable RC equals 1.

Results

The **READ** command returns the following variables to the stem:

0
Number of matching records found.

RRN
The relative record number.

LOGCODE
The log code of the record.

LOGTYPE
The log type of the record.

DESCRIPTION
The description of the log code.

DATE
The date stamp of the record. For example:

```
2009-03-16 Monday
```

TIME
The time stamp of the record. For example:

```
17.17.34.544527
```

STCK
The store clock time. For example:

```
BE82FEEAE80A0320
```

LSN
The logical sequence number. For example:

```
1-000000000002D4
```

If you specified **GLOBALS**, the **READ** command also return global fields to the stem. The available global fields depend on the type of record that the **READ** command is processing.

Return codes

After the command runs it sets the REXX special variable RC to one of the following return codes:

0
Successful: field values retrieved and position set.

1
The limit set by the **FINDLIM** parameter reached.

2
End of file for **NEXT** parameter. Start of file for **PREV** parameter.

8
RRN not found. ALZ.ERRMSG set to FUW8000E.

20
Unrecoverable error. For example, a syntax error or an I/O error. Message in SYSTSPRT log.

Related reference

Global fields

Global fields are generic field names defined by Transaction Analysis Workbench that map to record-type specific field names in log records.

Chapter 80. Knowledge module source reference

Knowledge module source members are assembler source. They consist of your original mapping macro, calls to the knowledge module macro FUWUKMF, and exit routines that enable you to programmatically alter field attributes.

A sample knowledge module source member, FUWUKMS, is supplied in the SFUWSAMP sample library.

Related tasks

Analyzing IMS user log records

IMS applications can use the DL/I **LOG** call to write records to the IMS system log. Such records are known as IMS user log records; they have a log code of X'A0' or greater, which is outside the range of log codes used by IMS itself. To enable Transaction Analysis Workbench to process and format IMS user log records, you create knowledge modules for the corresponding log codes.

Reasons to edit knowledge module source

When Transaction Analysis Workbench creates a knowledge module source member from an assembler mapping macro, it displays the source member in an ISPF edit session, with messages identifying lines in the source that require editing.

Reasons to edit knowledge module source include the following:

Field attributes are incomplete

For example:

- Variable length fields require a field exit to calculate the field length.
- Flag fields require that their bit settings are correctly specified.

The record consists of more than one section or segment

- Additional sections in the log record that require their own addressability require a base exit to resolve the base address.
- Repeating sections (MULT=YES) require a special base exit to ensure that all segments in the record are processed.

The mapping DSECT includes fields not defined in the macro member

The scanning process cannot detect when the mapping macro uses other macros or copybooks to define additional log record fields. To include these fields you will need to manually insert the missing field definitions.

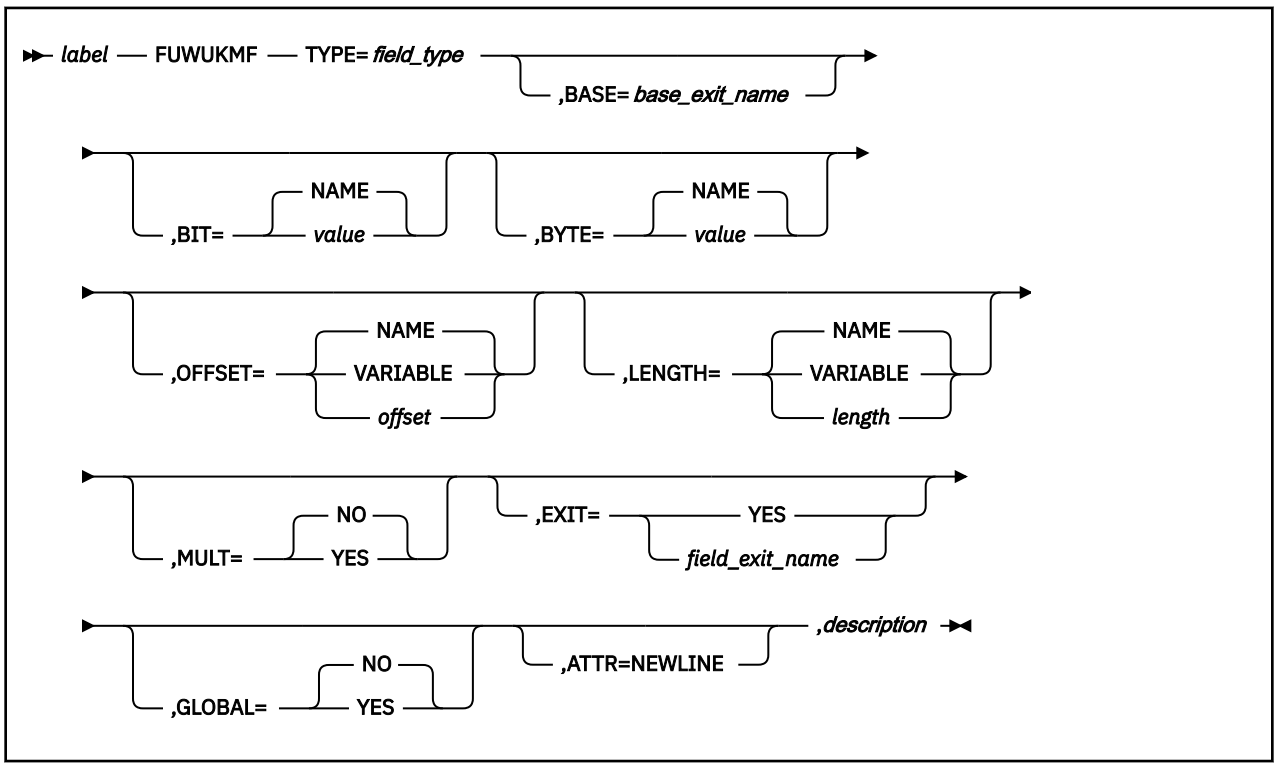
Customized processing

- You can change the format of reported field values or include additional information into the report, requiring changes to field definitions and field exits.
- Insert informative comments to improve the presentation of the formatted log record.

FUWUKMF macro: Define knowledge module fields

The FUWUKMF assembler macro defines how Transaction Analysis Workbench formats fields in a record.

Format



label

The name of the field as defined in the assembler mapping macro. This name is used to identify the field in the formatted report.

ATTR=

Specifies the field reporting attribute.

ATTR=NEWLINE

Specifies that the field is always reported at the start of a new line.

BASE=*base_exit_name*

Required for TYPE=SEGMENT only, specifies the name of the base exit. If not specified, label is used as the *base_exit_name*. The base exit has an entry point of BASE_*base_exit_name* and establishes the base address of the segment and all the fields that follow in the segment.

BIT=

Required for TYPE=BIT only, specifies the bit value of a flag byte. The default is BIT=NAME requesting that the name field of the EQU instruction is used. BIT specifications must be preceded by a TYPE=FLAG field, for example:

```
FLAG1  DS  X     - TYPE=FLAG
F1TRAN EQU X'80' - TYPE=BIT,BIT=NAME
F1TERM EQU X'40' - TYPE=BIT,BIT=NAME
```

BYTE=

Required for TYPE=BYTE only, specifies the whole byte value of a flag byte. The default is BYTE=NAME requesting that the name field of the EQU instruction is used. BYTE specifications must be preceded by a TYPE=FLAG field, for example:

```
STATUS DS  X     - TYPE=FLAG
OPEN   EQU 1     - TYPE=BYTE,BYTE=NAME
```

```
ACTIVE EQU 2     - TYPE=BYTE,BYTE=NAME
CLOSED EQU 3     - TYPE=BYTE,BYTE=NAME
```

EXIT=YES or *label*
Specifies that the field exit is to be invoked for each log record. Field exits can be used to change field attributes based on the current log record, for example a variable length field whose length can only be determined from the log record itself.

The field exit has an entry point of EXIT_ *label*. For EXIT=YES, the field name label is used.

GLOBAL=NO or YES
Optional for the first TYPE=SEGMENT only. Specify GLOBAL=YES when the log record has a global exit. Global exits are used to set the global field values, for example TRANCODE and USERID.

LENGTH=NAME or *length*
Specifies the length of the field. Specify LENGTH=NAME (the default) to request the implied length of the field from the DS instruction.

You can use a field exit (EXIT=) to change the field length.

MULT=NO or YES
Optional for TYPE=SEGMENT fields only. Specifies whether the segment can be repeated multiple times in the record. The base exit is invoked once for each repeating segment.

OFFSET=NAME or *offset*
Specifies the offset to the field from the start of its segment. Specify OFFSET=NAME (the default) to request the offset calculated as the difference between the field and its segment location. The offset is used to calculate the address of the field in the record.

You can use a field exit (EXIT=) to change the field address.

TYPE=*field_type*
Specifies the type of field and is used to determine how the field is processed and reported:

BIT
Defines the field as a bit setting and must follow a TYPE=FLAG field, for example:

```
FLAG1  FUWUKMF TYPE=FLAG,'Shutdown Status'
F1SHUT FUWUKMF TYPE=BIT,'Normal Shutdown'
F1DUMP FUWUKMF TYPE=BIT,'Shutdown was DUMPQ/PURGE'
F1STAE FUWUKMF TYPE=BIT,'STAE Exit Termination'
F1RSTA FUWUKMF TYPE=BIT,'Restart in Progress'
```

BYTE
Defines the field as a byte value and must follow a TYPE=FLAG field, for example:

```
SFLAG1 FUWUKMF TYPE=FLAG,'System Flag'
S1OS   FUWUKMF TYPE=BYTE,'System OS'
S1VSR  FUWUKMF TYPE=BYTE,'System VS and V=R'
S1VSV  FUWUKMF TYPE=BYTE,'System VS and V=Virtual'
```

CHAR
Defines the field as a character string.

DSECT
Generates the field definition DSECTs required by the knowledge module to support the field definitions and exits.

DUMP
Defines the field as a long string to be reported in dump format.

EOF
Indicates the end of the field definitions.

FLAG
Defines the field is a flag byte. The TYPE=BIT or TYPE=BYTE fields that follow are associated with this flag, allowing the flag to be broken down (using zoom) into its individual bit settings.

HEX
Defines the field as a hexadecimal string.

INFO
Inserts an informational comment (based on the specified description) into the formatted report. INFO field definitions do not relate to any field in the record.

INT
Defines the field as an integer. The length of an integer cannot exceed 4 bytes.

SEGMENT
Identifies the start of a new section or segment in the log record.

TYPE=SEGMENT must be the first field specified in the knowledge module to indicate the start of the record.

Each segment must have a base exit to establish the address of the segment in the log record.

All subsequent fields in the segment are located by adding their offset to this address.

Each segment definition must be paired with a TYPE=SEGMENTEND specification at the end of the fields for this segment to establish the segment boundary. Segments can be nested.

SEGMENTEND
Identifies the end of a section or segment in the log record.

STCK
Defines the field as an 8-byte store-clock value in timer-units. For STCK values that represent a date and time, the field value is reported in ISO date and time format. For small STCK values that represent an elapsed time, the field value is reported in ISO time format.

UTC
Defines the field as a coordinated universal time stamp (UTC) field.

Example

The following is an example of multiple invocations of the macro to perform field definitions within a knowledge module. It is supplied as member FUWUKMS in the SFUWSAMP sample library.

```
DBTRACE  FUWUKMF TYPE=SEGMENT,
               GLOBAL=YES,'Database trace event'
LL       FUWUKMF TYPE=HEX,'Log Record length'
ZZ       FUWUKMF TYPE=HEX,'QSAM ZZ, always zero'
TYPE     FUWUKMF TYPE=FLAG,'Log record type'
TYPEFA   FUWUKMF TYPE=BIT,'Type FA'
SUBT     FUWUKMF TYPE=HEX,'Log record subtype'
SUBT01   FUWUKMF TYPE=BIT,'FF Database trace'
SUBT02   FUWUKMF TYPE=BIT,'FP Database trace'
FLAG1    FUWUKMF TYPE=FLAG,'Flag Byte 1'
F1MPP    FUWUKMF TYPE=BIT,'MPP'
F1BMP    FUWUKMF TYPE=BIT,'BMP'
F1IFP    FUWUKMF TYPE=BIT,'IFP'
TRANCD   FUWUKMF TYPE=CHAR,'Transaction Code'
DBENT    FUWUKMF TYPE=SEGMENT,MULT=YES,'Database entry'
DBNAME   FUWUKMF TYPE=CHAR,'Database name'
DBINTENT FUWUKMF TYPE=FLAG,'Intent'
DBUPDATE FUWUKMF TYPE=BYTE,'Database was updated'
DBREAD   FUWUKMF TYPE=BYTE,'Database was read only'
DBNONE   FUWUKMF TYPE=BYTE,'Database was not used'
UPDATES  EQU   DBUPDCNT
UPDATES  FUWUKMF TYPE=INT,'Update DLI call count'
GETS     EQU   DBGETCNT
GETS     FUWUKMF TYPE=INT,'Get DLI call count'
DBENT    FUWUKMF TYPE=SEGMENTEND
DBTRACE  FUWUKMF TYPE=SEGMENTEND
⋮
```

Figure 151. Sample knowledge module source member

Knowledge module exit routines

You can write exit routines (often referred to simply as *exits*) to provide special processing not supported by the FUWUKMF macro; for example, for segments whose starting positions must be calculated programmatically, or for fields with unusual data types.

Knowledge modules can contain the following three types of exit:

Base exits
Used to calculate the starting address (base) of a segment.

Field exits
Used when the field requires special processing.

Global exits
Used when the record contains one or more global field values. For example, transaction code or database name.

Rules for coding exits

Observe the following rules for coding knowledge module exits:

- All exits are invoked in AMODE 31 and problem program state, and must exit the same.
- Return code (register 15) is preset upon entry to indicate that the segment (RC=8) or field (RC=4) does not exist and must always return a supported value otherwise Transaction Analysis Workbench will fail.
- Registers 3 to 14 must remain unchanged. Use the 32K work area (register 13) to save and restore the registers if necessary.

Performance considerations for coding exits

Knowledge module exits can affect the response time of the Transaction Analysis Workbench ISPF-based log browser and batch processing. To improve performance, consider the following recommendations:

- Keep the code path to a minimum
- Avoid using MVS services such as GETMAIN
- Use the allocated work area

Base exits

Knowledge module base exits calculate the starting address (base) of a segment in a record.

All segments (TYPE=SEGMENT) must have a base exit. Fields belonging to the segment are offset from the start of the segment. The address of the field in the record is the base address plus the field offset. A special base exit (MULT=YES) is required when the segment can occur multiple times, one segment after another.

The base exit entry point label is BASE_*segment_field_name*. At least one base exit is always required to identify the start of the log record.

Input registers

Base exits use the following input registers:

Register
Contents

3
Address of the log record.

4
Exit entry point BASE_*segment_field_name*.

5
Address of segment field descriptor control block mapped by DSECT FLDDEF.

6
Address of the parent segment if the segment is nested, otherwise zero.

13
32K work area.

14
Return address.

15
Return code initialized to 8, indicating the segment does not exist.

Output registers

Base exits use the following output registers:

Register
Contents

0 to 2
Can be used as work registers and need not be restored upon return.

3 to 14
Must remain unchanged.

15
Return code indicating how Transaction Analysis Workbench is to process the segment:

RC
Meaning

0
Segment exists in the log record and is to be reported. Register 1 points to the start of the segment.

4
Repeating segment (MULT=YES) exists in the log record and is to be reported. Register 1 points to the start of the segment. When the segment and all its fields have been reported, the base exit is re-invoked so that positioning can be set to the start of the next segment. RC=8 signifies that all segments have been processed. FDWPLACE is DSECT FLDDEF can be used as a place-holder for the current segment address, and FDWCOUNT as a counter of the number of segments remaining.

8
Segment does not exist or all repeating segments have been processed. Transaction Analysis Workbench skips the segment and all its fields and proceeds to the next segment or end of record.

All other return codes will cause unpredictable results.

In the following example, an exit routine is coded against an invocation of FUWUKMF with TYPE=BASE.

```
DBTRACE  FUWUKMF TYPE=SEGMENT,'Database trace event'
         . . .
         USING WorkArea,R13
         USING DBTRACE,R3
         USING FLDDEF,R5
         USING BASE_DBTRACE,R4
BASE_DBTRACE DS 0H
         LA    R1,DBTRACE
         SR    R15,R15
         BR    R14
         DROP  R3,R4,R5
         EJECT ,
```

Figure 152. Base exit example

In the following example, an exit routine for a recurring segment is coded against an invocation of FUWUKMF with TYPE=BASE and MULT=YES:

```
DBTRACE  FUWUKMF TYPE=SEGMENT,MULT=YES,'Database entry'
         . . .
          USING DBTRACE,R3
         USING FLDDEF,R5
         USING BASE_DBENT,R4
BASE_DBENT DS  0H
         LM    R1,R2,FDWDATA
         LTR   R1,R1
         BNZ   BASE_DBENT_1
         LA    R1,DBENT1
         LA    R2,0
         B     BASE_DBENT_2
BASE_DBENT_1 DS 0H
         USING DBENT,R1
         LA    R1,DBENT+DBENTLEN
         DROP  R1
BASE_DBENT_2 DS 0H
         CL    R2,COUNT
         BNLR  R14
         LA    R2,1(R2)
         STM   R1,R2,FDWDATA
         SR    R15,R15
         BR    R14
         DROP  R3,R4,R5
```

Figure 153. Base exit for recurring segment example

Field exits

Knowledge module field exits allow you to apply special processing to field attributes, such as programmatically calculating field length, or presenting an interpreted value of a field instead of its literal data value.

Field exits can change most field attributes by updating the following field descriptor control block mapped by DSECT FLDDEF in macro FUWUKMF:

FDTYPE
Field type, used for example to change from hexadecimal to character.

FDADDR
Address of the field in the log record, used when the field offset varies due to other factors in the record, for example preceding optional fields based on flag settings.

FDLENGTH
Field length, used for dump or other fields with variable length.

FDADDR with FD1EXTRN bit set
Field value external to the log record, used when the field value is difficult to interpret and can be better presented in an easy-to-understand format. Alternative values of length 8 bytes or less can be saved in FDXDATA. For longer field values, use working storage (register 13), however in this case you must ensure that no two fields use overlapping the same storage.

FDADESC
Alternative field description, used when the field description can change due to other factors. For example, a flag setting determines the contents of a character field.

The field exit entry point label is EXIT_*field_name*.

Input registers

Field exits use the following input registers:

Register
Contents

1
Address of owning segment.

3
Address of the log record.

4
Exit entry point EXIT_*field_name*.

5
Address of the field descriptor control block mapped by DSECT FLDDEF.

13
32K work area.

14
Return address.

15
Return code initialized to 4, indicating the field does not exist.

Output registers

Field exits use the following output registers:

Register
Contents

0 to 2
Can be used as work registers and need not be restored upon return.

3 to 14
Must remain unchanged.

15
Return code indicating how Transaction Analysis Workbench is to process the field:

RC
Meaning

0
Field exists in the log record and is to be reported.

4
Field does not exist and is not reported. Processing resumes at the next field in the segment.

8
Field does not exist and is not reported. Processing resumes at the next field in the segment.

All other return codes will cause unpredictable results.

In the following example, a field exit routine is coded against an invocation of FUWUKMF with TYPE=STCK. It shows how a new field, elapsed time, can be derived from start and stop time fields.

```
Elapsed  FUWUKMF TYPE=STCK,EXIT=YES,OFFSET=0,'Transaction elapsed time (STCK)'
         . . .
         USING DBTRACE,R3
         USING FLDDEF,R5
         USING EXIT_Elapsed,R4
EXIT_Elapsed DS 0H
         LG    R1,STOSTCK
         LG    R2,STASTCK
         SLGR  R1,R2
         STG   R1,FDXDATA
         LA    R0,FDXDATA
         ST    R0,FDADDR
         OI    FDADDR,X'80'
```

Figure 154. Field Exit example

Global exits

Use a knowledge module global exit when the record contains global field values that you want to see in the formatted list of log records. For example, transaction code or database name.

The global exit entry point label is EXIT_GLOBAL.

A user log record can participate in tracking by setting one or more of the global fields used for tracking.

Input registers

Global exits use the following input registers:

Register
Contents

3
Address of the log record.

4
Exit entry point GLOBAL_EXIT.

5
Address of global fields list mapped by DSECT FUWGF.

13
32K work area.

14
Return address.

Output registers

Global exits use the following output registers:

Register
Contents

0-2, 15
Can be used as work registers and need not be restored upon return.

3-14
Must remain unchanged.

In the following example, global fields are set against an invocation of a segment with GLOBAL=YES. The fields TRANCD, USID, LTRM, and TERM from the mapping macro are set to populate global variables.

```
         USING DBTRACE,R3          .R3=&gt;Log Record
         USING FUWGF,R5            .R5=&gt;KM Global Fields
         USING GLOBAL_EXIT,R4      .R4=&gt;Entry Point
GLOBAL_EXIT DS 0H                  .Set Global Variables
*Set the following Global variables if available in the record
         MVC   GF_TRAN,TRANCD      .Transaction Code
*        MVC   GF_PROG,...         .Program (PSB) Name
         MVC   GF_USID,USID        .User ID
         MVC   GF_LTERM,LTRM       .Logical Terminal
         MVC   GF_TERM,TERM        .VTAM Node/APPC NetID/OTMA Tpipe
*        MVC   GF_DBASE,...        .Database Name
*        MVC   GF_RBA,...          .VSAM RBA or OSAM RBN
         MVC   GF_PST,PST          .PST (Region) ID
         MVC   GF_ORGID+0(8),IMSID     .Originating
         MVC   GF_ORGID+8(8),ORGUOWID  .Tracking UOWID
         MVC   GF_IMSID,IMSID          .IMS System ID
         MVC   GF_RECTK+0(8),IMSID     .Recovery
         MVC   GF_RECTK+8(8),RECTOKEN  .Token
*        MVC   GF_INFO,INFO        .User Information
*        MVC   GF_DESC,...         .Alternative description
         SR    R15,R15             .RC=0=Global Fields are set
         BR    R14                 .Return to caller
         DROP  R3,R4,R5
```

Figure 155. Global exit example

Chapter 81. System and file types

System definitions are stored in one of two system definition repositories, depending on the system type. Each type of system can write to one or more type of file. Some file types can contain records from more than one system type. Some types of file are eligible for automated file selection.

System types

You can use Transaction Analysis Workbench to create and edit system definitions in the following two system definition repositories.

IMS system definition repository

Table 35. System definitions in the IMS system definition repository and their applicable file types

System type	System type description	Applicable file types	File type description	Automated file selection?
IMS	IMS subsystem	LOG	IMS log	Yes
		MON	IMS monitor	
		CQS	CQS, including the log stream	
		ATF	Tivoli OMEGAMON XE for IMS Application Trace Facility journal	
CONNECT	IMS Connect system	LOG	IMS Connect Extensions journal	Yes

CICS, DB2, IBM MQ, and MVS image system definition repository

Table 36. System definitions in the CICS, DB2, IBM MQ, and MVS image system definitions repository and their applicable file types

System type	System type description	Applicable file types	File type description	Automated file selection?
CICS	CICS APPLID	SMF	System management facilities (SMF) containing CICS monitoring facility (CMF) records	Yes
		TRACE	Either: • z/OS generalized trace facility (GTF) data set containing CICS trace entries • CICS auxiliary trace data set	

Table 36. System definitions in the CICS, DB2, IBM MQ, and MVS image system definitions repository and their applicable file types (continued)

System type	System type description	Applicable file types	File type description	Automated file selection?
DB2	DB2 subsystem	LOG	DB2 log	Yes
		SMF	SMF containing DB2 statistics, accounting, system, or performance trace records	Yes
		TRACE	GTF data set containing DB2 trace records	
		NTH	Tivoli OMEGAMON XE for DB2 Performance Expert near-term history	Yes
IMAGE	z/OS image	SMF	SMF (general)	Yes
		OPERLOG	OPERLOG (log stream)	
MQ	IBM MQ subsystem	LOG	IBM MQ log extract	
		SMF	SMF containing IBM MQ statistics and accounting records	Yes

File types

Table 37. File types

File type	Description
ATF	Tivoli OMEGAMON XE for IMS Application Trace Facility journal
CQS	IMS CQS log
INDEX	Depending on the system type: CICS transaction index, DB2 accounting index, IMS transaction index, IMS Connect transaction index, or IBM MQ accounting index
LOG	Depending on the system type: DB2 log, IMS log, IMS Connect Extensions journal, or IBM MQ log extract
MIXED	Extract of one or more file types
MON	IMS monitor
NTH	Tivoli OMEGAMON XE for DB2 Performance Expert near-term history
OPERLOG	OPERLOG (log stream)
SMF	SMF
TRACE	GTF data set or, for CICS systems, CICS auxiliary trace data set
UNKNOWN	Type is established when file is referenced

Related concepts

Transaction indexes

Transaction indexes are specialized extracts that contain a single record type, where each record contains information about a single transaction (or thread), sorted in time sequence.

Automated file selection utility

The automated file selection utility is a batch program that locates the log files for a session, based on a time period and a system definition.

Repositories
Repositories are the data sets where Transaction Analysis Workbench stores data related to transaction analysis. Users can share repositories. Some repositories can be shared with other products.

Related tasks

Defining systems for log file selection
Before you can use automated file selection, you must define to Transaction Analysis Workbench the systems that generated the log files you want to analyze. You might also wish to define systems for documentation purposes: when you create a session for a problem, you can specify the systems that are involved in the problem.

Chapter 82. Log stream types

Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

For example:

```
SMF:IFASMF.PROD
```

The prefix distinguishes log stream names from data set names, and enables Transaction Analysis Workbench to correctly format the records of each type of log stream.

Use the "OTHER" type for log streams that do not belong to one of the more specific types in the following table. If you prefix a log stream name with `OTHER:`, Transaction Analysis Workbench shows the records in "raw" dump format.

Table 38. Log stream types

Log stream type	Description
ATF	Tivoli OMEGAMON XE for IMS Application Trace Facility (ATF) log stream
CQS	IMS Common Queue Server (CQS) log streams: both message queue (MSGQ) primary (and optional overflow) structures and expedited message handler queue (EMHQ) primary (and optional overflow) structures
OPERLOG	The z/OS operations log
SMF	System management facilities (SMF) log stream
VSAM	CICS VSAM forward recovery and autojournaling log stream
OTHER	Other log streams (no record formatting)

Related tasks

Browsing logs in your personal ad hoc list
Using your personal ad hoc list of logs offers a private alternative to using sessions. There might be occasions when you want to browse logs without registering a session. Simply add logs to your personal ad hoc list, and then select them for browsing.

Related reference

LOGSTREAM command
Specifies that the input log file is a log stream rather than a data set.

EXTRACT command
Writes the selected log records to an extract data set.

MVS: OPERLOG and OTHER log stream codes
The MVS log type consists of log codes that identify records from OPERLOG and records from log streams specified using the prefix `OTHER:`.

Chapter 83. Log types and codes

Transaction Analysis Workbench uses two identifiers to classify log records: a general *log type* and a more specific *log code*. A log type is a short mnemonic that typically corresponds to the type of system that writes the log record or the type of file to which the log records were originally written. A log code identifies a specific type of log record within a log type.

Log type and code combinations are sometimes shown joined by a colon. For example, `IMS:01` refers to records of log type IMS, log code 01.

For details of the supported log codes for each log type, see the log type descriptions.

A log code is known as *supported* if Transaction Analysis Workbench supplies a specific knowledge module for those records. You can, for instance, use Transaction Analysis Workbench to browse formatted views of these records with their field names, and their field values displayed according to the appropriate data type.

You can use Transaction Analysis Workbench to create extracts that contain log records of any combination of log types and codes.

Related concepts

What does Transaction Analysis Workbench do?
Transaction Analysis Workbench interprets a wide range of logs from many z/OS subsystems, and consolidates them in a consistent, feature-rich environment.

Knowledge modules
A *knowledge module* is an executable load module that Transaction Analysis Workbench uses to interpret and process a particular type of log record.

Extracts
You can use Transaction Analysis Workbench to create extracts of original log files, containing the subset of records that you need to analyze a problem, within the relevant time period.

Related tasks

Highlighting log types in different colors
You can customize the color and highlighting of records according to their log type.

Related reference

Hardware and software prerequisites
Before you install and configure Transaction Analysis Workbench, make sure that your environment meets the following minimum hardware and software requirements.

Filter log types and codes
A filter specifies one or more combinations of log type and code, identifying the log records that the filter selects or excludes.

CODE command and COND statement
The **CODE** command filters log records by including or excluding them based on their log type and code. To refine the filter based on field values in a log record, follow the **CODE** command with one or more **COND** statements.

ATF: OMEGAMON for IMS ATF codes

The ATF log type consists of log codes for Tivoli OMEGAMON XE for IMS (OMEGAMON for IMS) Application Trace Facility (ATF) journal records.

Table 39. ATF log codes supported by Transaction Analysis Workbench

Log code	Description
01	DLI

Table 39. ATF log codes supported by Transaction Analysis Workbench (continued)

Log code	Description
03	Fast Path
04	Summary To process ATF summary records that have been written to an IMS log, you must specify the IMS log code that ATF used to write those records to the log. To specify the IMS log code in the Transaction Analysis Workbench ISPF dialog, go to option 0.5 **IMS Tools Settings** and set the **Log Code** field under the **OMEGAMON for IMS** heading. To specify the IMS log code while browsing logs, enter the following primary command: `ATC hh` where *hh* is the 2-digit hexadecimal log code. To specify the IMS log code in a batch job, use the **ATF** command of the Transaction Analysis Workbench report and extract utility.
05	DLI IOPCB
06	Monitor
82	DB2
87	Generic External Subsystem (ESS)
88	IBM MQ
F1	Transaction end

Related reference

ATF command
Specifies the IMS log record code that OMEGAMON for IMS on z/OS uses to write Application Trace Facility (ATF) summary records to IMS log data sets.

CMF: CICS monitoring facility codes

The CMF log type consists of log codes that identify MVS System Management Facilities (SMF) type 110, subtype 1 records written by the CICS monitoring facility.

Table 40. CICS monitoring facility (CMF) records supported by Transaction Analysis Workbench

Log code	Description
6E13	CICS monitoring facility (CMF) performance class data

Typically, the last two digits of a 4-digit hexadecimal log code identify the record subtype. However, CMF records have several classes, each with its own record format. The third digit of a CMF log code identifies the subtype, and the fourth digit identifies the class. Hence, the log code 6E13 identifies CMF performance class data (SMF type 110, subtype 1, class 3).

SMF type 110, subtype 1 records can contain information for multiple CICS transactions. Transaction Analysis Workbench splits each original SMF record into one CMF 6E13 record per transaction.

Related concepts

CICS performance records
Typically, the last two digits of a 4-digit hexadecimal log code identify the record subtype. However, CMF records have several classes, each with its own record format. The third digit of a CMF log code

identifies the subtype, and the fourth digit identifies the class. Hence, the log code 6E13 identifies CMF performance class data (SMF type 110, subtype 1, class 3).

CON: IMS Connect event codes

The CON log type consists of log codes for IMS Connect events that have been recorded by IMS Connect Extensions.

As IMS Connect processes incoming transaction requests, IMS Connect Extensions writes details of those requests to event records in journal data sets.

Which records are available depends on the types of IMS Connect events that occurred and the IMS Connect Extensions collection level that was set.

The log code of a CON record consists of four hexadecimal digits:

- The first two digits are the log record prefix that IMS Connect Extensions used when it wrote the event records.

  If the event record identifier for your IMS Connect Extensions installation is not A0, then you must specify your identifier to Transaction Analysis Workbench: select ISPF dialog option 0.5 **IMS Tools Settings**.
- The last two digits identify the type of event.

  For details, see the information about event record types in the *IMS Connect Extensions User's Guide*.

Related concepts

IMS Connect Extensions journal records
You can use Transaction Analysis Workbench to analyze IMS Connect events that have been recorded in an IMS Connect Extensions journal data set. IMS Connect Extensions is an IBM tool that provides instrumentation for IMS Connect. The tool collects real-time data about IMS Connect events, which it then regularly archives.

CQS record codes

The CQS log type consists of log codes for IMS Common Queue Server (CQS) records.

The log code of a CQS record consists of four hexadecimal digits: the first two digits represent the record type; the last two digits represent the record subtype. For details, see the information about CQS log records in *IMS Diagnosis*.

CTR: CICS trace entry codes

The CTR log type consists of log codes that identify CICS trace entries.

The log code of a CTR record is the two-character domain ID of the CICS trace entry. For example, the log code AP identifies a CICS trace entry from a trace point in the application domain. For a list of domain IDs, see the information about trace points in the CICS documentation.

Transaction Analysis Workbench can read CICS trace entries from CICS auxiliary trace data sets or z/OS generalized trace facility (GTF) data sets.

The first four characters of the log record description contain the hexadecimal code of the trace point in that domain.

DB2 log codes

The DB2 log type consists of log codes for DB2 log records.

Transaction Analysis Workbench uses *umbrella* log codes to refer to DB2 log records. Each umbrella log code represents a category of DB2 log record, spanning one or more actual DB2 log codes.

Umbrella codes are especially useful for filtering DB2 checkpoint records, which are many in both form and number. For example, you can choose to exclude from view all DB2 checkpoint records, creating a more transaction-centric view of the log data. The following table lists the Transaction Analysis Workbench umbrella log codes and the corresponding actual DB2 log codes. All codes are 2-byte hexadecimal values.

Transaction Analysis Workbench supplies knowledge modules for the following DB2 log records.

Table 41. DB2 umbrella log codes supported by Transaction Analysis Workbench

Umbrella log code	Description	Actual DB2 log codes included
0600	Unit-of-recovery (UR) undo/redo	*x*6*xx*, *x*2*xx*, *x*4*xx*
0020	UR control	*xx*2*x*
0008	Non-UR related lock	*xxx*8
0100	Checkpoint record	*x*1*xx*
0002	Page set control	*xxx*2
0010	System event	*xx*1x
0800	Archive log command	*x*8*xx*
0004	Syscopy utility	*xxx*4
1000	Data manager	1*xxx*
4000	Diagnosis	4*xxx*

Note:

- *x* represents any hexadecimal digit.
- The order of rows in this table is significant. Transaction Analysis Workbench assigns actual DB2 log codes to the first matching umbrella log code in the table. For example, Transaction Analysis Workbench assigns the actual DB2 log code 1400 to the umbrella log code 0600, not 1000.

Related concepts

DB2 log records

DB2 log records contain detailed recovery information about each change operation performed against DB2 database objects. These changes are recorded as database *undo* (backout) records, *redo* (reapply) records, or a combination of both.

DTR: DB2 trace codes

The DTR log type consists of log codes for DB2 trace records.

The log codes for DB2 trace records match the IFCIDs, formatted as 3-digit decimal numbers with leading zeros. For example, log type DTR, log code 003 represents IFCID 3 (DB2 accounting records).

Related concepts

Collecting DB2 trace data

To collect DB2 trace data, including accounting data, use the DB2 command **START TRACE**.

IMS log codes

The IMS log type consists of log codes for IMS log records. Not all log codes are applicable to all releases of IMS.

The log code of an IMS record matches the IMS log record type and, if applicable, subtype. For a list of IMS log record types and subtypes, see the information about IMS log records used to analyze IMS problems in *IMS Diagnosis*.

Notes for specific log codes:

67FA
Trace table. Transaction Analysis Workbench formats these records as individual trace entries (log type ITR).

67FF
Exception condition SNAP. Transaction Analysis Workbench formats 67FF IMS trace table records as individual trace entries (log type ITR).

CA01
IMS transaction index.

CA20
IMS Connect transaction index.

Related concepts

IMS instrumentation
IMS instrumentation is provided by IMS itself and also by other products.

Related reference

ITR: IMS trace table entry codes
The ITR log type consists of log codes that identify IMS trace table entries in IMS log records of type 67FA and 67FF.

ITR: IMS trace table entry codes

The ITR log type consists of log codes that identify IMS trace table entries in IMS log records of type 67FA and 67FF.

An ITR log code consists of either two or four hexadecimal digits.

The first two digits represent the first byte of the table entry, which identifies the table entry type.

Some table entries have a subtype, represented by the second two digits of the log code. Subtype values are derived differently depending on the table entry type.

For details, see the information about trace records in *IMS Diagnosis*.

Related reference

Traces
IMS log records related to traces.

IMS log codes
The IMS log type consists of log codes for IMS log records. Not all log codes are applicable to all releases of IMS.

MON: IMS monitor codes

The MON log type consists of log codes for IMS monitor records.

A MON log code consists of four hexadecimal digits. The first two digits are X'4E'. The last two digits represent the IMS Monitor SLOG code, which identifies the monitor event.

MQ: IBM MQ log codes

The MQ log type consists of log codes that identify IBM MQ log extract records.

Table 42. IBM MQ log extract records supported by Transaction Analysis Workbench

Log code	Description
0001	Put
0002	Get
0003	Object Alteration
0004	Object Definition
0005	Message Expiry
0006	Commit Phase 1
0007	Commit Phase 2
0008	Backout
00FF	Log Extract

Related concepts

IBM MQ log extracts

You can use Transaction Analysis Workbench to analyze log extracts from IBM MQ.

MVS: OPERLOG and OTHER log stream codes

The MVS log type consists of log codes that identify records from OPERLOG and records from log streams specified using the prefix 0THER:.

Use the 0THER: prefix to process log streams for which Transaction Analysis Workbench does not provide specific record formatting.

Table 43. Log codes for OPERLOG and OTHER log stream records

Log code	Description
CA52	OPERLOG record
CAFF	Record from a log stream specified using the prefix 0THER:

Related concepts

z/OS SMF and OPERLOG records

You can use Transaction Analysis Workbench to analyze OPERLOG records and various SMF records.

Related reference

Log stream types

Transaction Analysis Workbench can read log records in log streams. To refer to a log stream in Transaction Analysis Workbench, you prefix the log stream name with a log stream type followed by a colon.

SMF codes

The SMF log type consists of log codes that identify MVS System Management Facilities records.

In one sense, Transaction Analysis Workbench supports *all* SMF record types: you can use the Transaction Analysis Workbench ISPF dialog to browse any SMF record type. However, if Transaction Analysis Workbench does not have a specific knowledge module for an SMF record type, it reverts to using a

generic SMF knowledge module that only interprets the header fields that are common to all SMF records, and displays the remaining type-specific fields in an uninterpreted dump format.

The following table lists the SMF record types and subtypes for which Transaction Analysis Workbench supplies a specific knowledge module.

Table 44. Supported SMF record types and their log codes

SMF record type (decimal) "1" on page 642	Log code (hexadecimal)	Description	Contents relevant to analyzing transactions	Report available ? "2" on page 642
29	1D	IMS BPE statistics, subtype 1: ODBM accounting	ODBM request processing for charge back	
30	1E	Common address space work	Address space accounting: IMS and CICS address space resource usage, including CPU, I/O activity by ddname, paging activity, and virtual storage usage	Yes
33	21	APPC/MVS transaction program (TP) accounting	APPC transaction response time	Yes, subtype 2 (33-2) only
42-6	2A06	DFSMS	DASD data set level I/O statistics	Yes
64	40	VSAM cluster status	VSAM data set read and update activity, CI-CA splits	Yes
70-1	4601	RMF processor activity	System CPU usage	Yes
71	47	RMF paging activity	System paging and page data set activity	
72-3	4803	RMF workload activity	Service class response time analysis	
74-1	4A01	RMF device activity	DASD device I/O activity and performance	
74-2	4A02	RMF cross-system coupling facility (XCF) activity	Message traffic flow between systems in the sysplex	
75	4B	RMF page data set activity	Auxiliary storage usage and page data set I/O activity	Yes
77	4D	RMF enqueue activity	Identify resources where enqueue/dequeue contention occurred	
78-2	4E02	RMF virtual storage activity	Common storage (CSA, SQA) usage and Job private storage usage	Yes
79-15	4F0F	Internal resource lock manager (IRLM) long lock detection	Identify IMS database locks held for a long time	Yes
88-1	58	System logger data	Log stream activity and performance degradation events	Yes

Table 44. Supported SMF record types and their log codes (continued)

SMF record type (decimal) "1" on page 642	Log code (hexadecimal)	Description	Contents relevant to analyzing transactions	Report available ? "2" on page 642
100	64"3" on page 643	DB2 statistics	Transaction data collected at event monitoring points	
101	65"3" on page 643	DB2 accounting	Depending on which of the following accounting trace classes are active: **1** Thread (including IMS and CICS) call elapsed and CPU times **2** In-DB2 elapsed and CPU times **3** Suspend times, Buffer manager, locking, SQL DML statistics	Yes
102	66"3" on page 643	DB2 performance	DB2 response times	
110-1.3	6E13"4" on page 643	CICS monitoring facility (CMF) performance class data.	CICS transaction performance	
115-1	7301	IBM MQ statistics	Log manager	
115-2	7302	IBM MQ statistics	Message counts, buffer and paging information	
116-0	7401	IBM MQ class 1 accounting	Thread (including IMS and CICS) GET-PUT activity, CPU time for MQ calls	Yes
116-1 and 116-2	7401 and 7402	IBM MQ class 3 accounting	Thread (including IMS and CICS) GET-PUT activity, CPU time for MQ calls	Yes
120-9	7809	WebSphere Application Server for z/OS	Request activity performance statistics	
120-11	780B	z/OS Connect	Audit interceptor	
123-1	7B01	z/OS Connect Enterprise Edition (EE)	Audit interceptor	

Table notes:

1. Colloquially, and in the SMF documentation, SMF records are typically known by their decimal type values. A hyphen indicate a record subtype. A period indicates a class within a type or a subtype. For example, 116-1 refers to SMF type 116, subtype 1. Transaction Analysis Workbench refers to SMF records using a 2- or 4-digit hexadecimal log code.
2. For some SMF record types, you can request a report whose layout is specific to that record type, showing fields from the record that are relevant to analyzing transactions. You can request these reports using the ISPF dialog, or using the SMF parameter of the **REPORT** command in your own JCL.

3. The SMF log type does not contain the log codes 64, 65, or 66. Instead, these DB2 trace records belong to the more specific DTR log type.
4. The SMF log type does not contain the log code 6E13. Instead, CMF performance class records belong to the more specific CMF log type.

Related concepts

What does Transaction Analysis Workbench do?
Transaction Analysis Workbench interprets a wide range of logs from many z/OS subsystems, and consolidates them in a consistent, feature-rich environment.

z/OS SMF and OPERLOG records
You can use Transaction Analysis Workbench to analyze OPERLOG records and various SMF records.

Related reference

SMF reports
Transaction Analysis Workbench provides reports for several types of SMF record. The reports show selected fields from the specified SMF record type, and values calculated from those fields, to help you analyze transactions.

VSR: CICS VSAM forward recovery and autojournaling codes

The VSR log type consists of log codes that identify IBM CICS VSAM forward recovery and autojournaling records.

The VSR log type, or VSAM Space Recovery Log, is an essential component of CICS VSAM Recovery for z/OS (CICS VR). It is vital to ensure data integrity and recovery during system failure or corruption.

When CICS applications change VSAM data files, CICS VR records these changes in the VSR log. The log contains a record of all updates, inserts, deletes, and other operations performed on VSAM files and is essential for data recovery.

The Transaction Analysis Workbench has the ability to read records directly from a log stream or a data set containing records extracted from the log stream. To view these records through Transaction Analysis Workbench, navigate to ISPF dialog option **4 Process**, and then you must specify the log stream name with the prefix VSAM followed by a colon.

For example:

```
VSAM:RGNUSR.APPLA.GRP1
```

To add a log stream or data set of these records to a session, navigate to ISPF dialog option **1 Sessions**, select option **3 Files**, and then insert a file of type JOURNAL for the corresponding CICS system.

Chapter 84. Event time stamps versus log record time stamps

Transaction Analysis Workbench defines an *event time stamp* for each input log record. Transaction Analysis Workbench presents event time stamps in various contexts, such as the **Time** column of the ISPF dialog log browser and the `time` field of CSV and JSON output. What the event time stamp represents depends on the record type.

Log record time stamps

Every log record contains a *log record time stamp* that represents when the record was written (or "cut"). The phrase "when the record was written" is useful for general understanding, but imprecise. The details depend on the record type.

The log record time stamp is typically in a fixed location defined by the record type, at or near the start of each log record, and can be read quickly without process-intensive parsing.

Event time stamps and log record time stamps can be different

For most record types, Transaction Analysis Workbench uses the log record time stamp as the event time stamp. For example, for records of log type SMF, the event time stamp represents when the record was written *to the SMF buffer*.

There are exceptions. Some log records also contain other time stamps that represent the start or end time of an event, distinct from the time that the log record was written. Such time stamps might be embedded deeper in the log record structure and can require more complex parsing to read. For some record types, Transaction Analysis Workbench uses one of these other time stamps as the event time stamp. For example (this is not a comprehensive list):

CICS performance records (log type CMF): Event time stamp is the CICS transaction start time
: A single SMF type 110 subtype 1 class 3 record can contain CICS monitoring facility (CMF) performance data for multiple CICS transactions. Transaction Analysis Workbench breaks such "wrapper" SMF records into one CMF record per transaction. Rather than defining the event time stamp of each CMF record as the log record time stamp of the SMF "wrapper" record, Transaction Analysis Workbench uses the CICS transaction start time.

IMS transaction index records (log type IMS, log code CA01): Event time stamp is the arrival time of the first related IMS log record
: Each IMS transaction index record is created from multiple IMS log records. The event time stamp of an IMS transaction index record is the arrival time stamp of the first IMS log record that is related to the IMS transaction. This time is sometimes referred to colloquially as the IMS transaction start time.

Event time stamps are not always monotonically increasing

The Transaction Analysis Workbench ISPF dialog log browser displays log records in the order they occur in the input logs. In most cases, the **Time** column values in the log browser are monotonically increasing; the event time stamp of each record is later than the previous record. However, log records that use an event time stamp that is not the log record time stamp, or logs that have been sorted or concatenated in some other order, can disrupt this pattern. The same is true of other outputs from Transaction Analysis Workbench, such as CSV and JSON: depending on the record types and order of records in the input logs, the `time` field values in successive output records are not always monotonically increasing.

Related concepts

Time stamps in CSV and JSON

CSV and JSON output by Transaction Analysis Workbench includes the time stamp of the corresponding input log record, also known as the *event* time stamp. The event time stamp is always included in the output, regardless of any **CSV** or **JSON** command parameters, such as **FIELDS**, that specify which other

fields to include. Depending on the input record type and the selected fields, other fields in the output might also be time stamps. All time stamps in the output, including the event time stamp, have the same format.

Related tasks

Tracking a transaction or an IMS unit of recovery
Tracking displays log records that are related to a particular transaction. For IMS transactions, you can track log records that are related to a particular unit of recovery (UOR) within a transaction.

Related reference

Log browser navigation controls
The main log browser panel (that displays a formatted view of log browser records) contains a variety of controls for navigating logs.

Time column display modes
The **Time** column in the log browser displays the event time stamp of each log record. You can display event time stamps in one of several modes, such as relative to the event time stamp of a selected record, or as the elapsed time between records.

START, STOP (FROM, TO) commands
Specify the report interval. Only log records with an event time stamp within the report interval are included in the output.

Chapter 85. Global fields

Global fields are generic field names defined by Transaction Analysis Workbench that map to record-type specific field names in log records.

Transaction Analysis Workbench uses global fields for the following purposes:

Filtering across all record types
Different log record types can contain the same data, such as transaction code, under different field names. These different field names complicate filtering across log record types. Global fields enable you to filter across all log record types without using record type-specific field names. Transaction Analysis Workbench maps filter values for global fields to the corresponding fields in each log record type.

Not all global fields are relevant to all log record types. Where a log record type does not contain information that corresponds to a global field, the global field value is null for that log record type.

Displaying field values while browsing logs
To display global field values while browsing logs in the ISPF dialog, press the Right function key (F11) to scroll between the available views; some of the views display global fields (and, depending on the log record type, other fields) under the **Description** column.

Tracking
Transaction Analysis Workbench uses some global fields to help determine which log records to display when you enter a transaction tracking line action next to a log record in the ISPF dialog (see the **Used for tracking?** in the following table).

The following table lists all global fields.

Table 45. Global fields

Field name	Description	Format	Length (bytes)	Used for tracking?
Abend	Abend code	Character, in one of the following specific formats: **System abends** S*xxx*, where *xxx* is a string of three hexadecimal digits. For example, S0C4. **User abends (including pseudo abends)** U*nnnn*, where *nnnn* is a string of four decimal digits. For example, U0777. **CICS abends** 4-byte alphanumeric string.	4 (system or CICS abends) or 5 (user abends)	No
ClientID	IMS Connect client ID	Character	8	No
Code	Log record type (first byte) and, where applicable, subtype (second byte)	Hexadecimal	2	No

Table 45. Global fields (continued)

Field name	Description	Format	Length (bytes)	Used for tracking?
Database	Database name	Character	8	No
IMSID	IMS subsystem name	Character (usually 4 bytes, left-aligned and padded with blanks)	8	No
IMSRel "1" on page 650	IMS release	Character	3	No
ITASK	Monitor ITASK number.	Hexadecimal	4	Yes
LogToken	IMS Connect logon token	Hexadecimal date/time store clock (STCK)	8	Yes
LongLock	Deadlock cycle number for IRLM long lock	Hexadecimal	4	Yes
LSN	Log record sequence number	Hexadecimal	8	No
LTerm	Logical terminal name	Character	8	No
LUWID	Logical unit of work ID	*network ID*/*LU name*/*unique value*/*commit count* For example: NET1/DB2LUA/ B7ADFE54E81F/0001	8	No
OrgUOWID	Originating tracking unit of work ID	*originating IMS ID*/*time stamp token* (8-byte character field/8-byte hexadecimal field) For example: IMSA/B765FE4519BE5A40 In a sysplex where the originating IMS ID is not known, specify the IMS ID as an asterisk (*).	See Format	Yes
Port	IMS Connect TCP/IP port	Character	8	No
Program	Program specification block (PSB) name	Character	8	No
RBA	VSAM relative byte address (RBA) or OSAM relative block number (RBN)	Hexadecimal	4	No

Table 45. Global fields (continued)

Field name	Description	Format	Length (bytes)	Used for tracking?
RecToken	Recovery token	*IMS ID/OASN followed immediately by commit number* where OASN is the outstanding recovery units (origin application schedule numbers) (8-byte character field/8-byte hexadecimal field) For example: IMSA/0000002300000001	See Format	Yes
Region	Program specification table (PST) ID	Hexadecimal	2	Yes
ResumeTP	IMS Connect logon token for resume tpipe	Hexadecimal date/time store clock (STCK)	8	Yes
SSID	Subsystem ID (for example: DB2, IMS, or IBM MQ subsystem ID)	Character	8	No
SSN	IMS Connect send sequence number	Hexadecimal	4	No
Terminal	VTAM® node name, Line/Pterm, APPC network ID, or OTMA tpipe	Character	8	No
TranCode	Transaction code	Character	8	No
URID	Unit of recovery ID (DB2, IBM MQ)	Hexadecimal	10	Yes
UserId	User ID	Character	8	No

Table 45. Global fields (continued)

Field name	Description	Format	Length (bytes)	Used for tracking?
UTC "1" on page 650	Universal Time Coordinated (UTC) time stamp. Shows when the event for which the log record was generated occurred, rather than the time when the log record was created. The time displayed in the Time column (referred to as the store clock, or STCK, time) is the time when the log record was created and is therefore different from the UTC. Not all records have a UTC field. You can display the UTC in local or GMT format.	*hh.mm.ss.thmiju* (from midnight, 00.00.00.000000, to 23.59.59.999999)	See Format	No

Table notes:

1. Not available for filtering.

Related reference

Filter condition field names or offsets
The field name or offset in a filter condition specifies the part of a log record that you want to compare with the filter condition value.

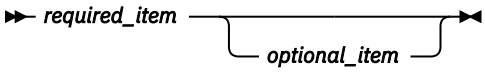
READ
Reads the contents of a record into a stem variable.

Chapter 86. How to read syntax diagrams

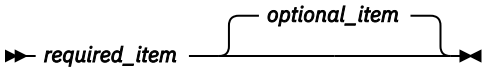
The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

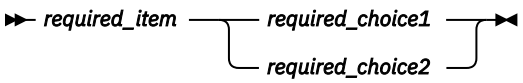


  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

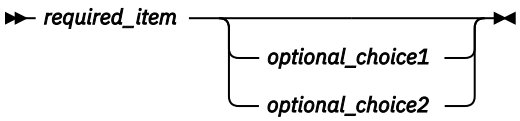


- If you can choose from two or more items, they appear vertically, in a stack.

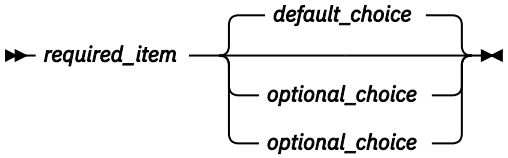
  If you *must* choose one of the items, one item of the stack appears on the main path.



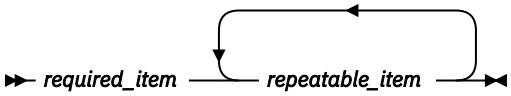
  If choosing one of the items is optional, the entire stack appears below the main path.



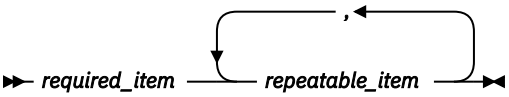
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



- An arrow returning to the left, above the main line, indicates an item that can be repeated.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses; for example, (1).

Notices

This information was developed for products and services offered in the U.S.A.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J64A/G4
555 Bailey Avenue

San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered marks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at: http://www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Product Number: 5697-P37

SC27-6747-04