Application Note

MEASURING DATA THROUGHPUT WITH THE CMX500

Solutions and Tipps for NR FR1 in TDD Downlink Mode

Products:

► R&S®CMX500

► R&S®CMsequencer

► R&S®CMsquares

► R&S®XLAPI

Yong Shi | 1SL379 | Version 2e | 08.2022

https://www.rohde-schwarz.com/appnote/1SL379

ROHDE&SCHWARZ

Make ideas real

Contents

1 Overview

5G New Radio (NR) is a radio technology specified by 3GPP and was first released in 3GPP release 15. It is designed to target three use cases, i.e. enhanced mobile broadband (eMBB), massive machine type communication (mMTC) and ultra-reliable low latency communication (URLLC). Among these three use cases, eMBB represents actually a further evolution of mobile broadband communication from LTE standard. According to the technical performance requirement defined by IMT-2020, by deploying 5G technology, peak data rates of eMBB application are expected to reach 20 Gbps in downlink (DL) and 10 Gbps in uplink (UL) direction, respectively. Typical use cases of eMBB are data hungry applications, such as high resolution 8K video streaming, virtual reality (VR), augmented reality (AR) etc.

Verification of a 5G capable user equipment (UE) with respect to its achievable maximum data throughput under controllable and deterministic test conditions is an essential process during the design phase of the product. Performance centric verification through identification of the data throughput bottleneck, product benchmarking against a golden device enhances tremendously the user experience in the end.

This document focuses on 5G NR frequency range 1 (FR1) with TDD duplex mode in E-UTRAN New-radio Dual Connectivity (ENDC) operating mode. As 5G NR physical layer offers a plethora of flexibility, the motivation here is to provide a kind of guideline of relevant parameter settings to stimulate device under test (DUT)'s max throughput capacity. The status quo of the R&S solutions at the time when the application note is created are described. Shown feature sets are constantly evolving, so the screenshots used and the parameters shown may change.

The remaining document is organized in the following way:

Chapter 2 highlights some key influence aspects of the data throughput in 5G NR

Chapter 3 presents the throughput measurements and test solutions based on R&S®CMX500 radio tester

Last but not least, Chapter 4 concludes the document with a brief summary.

The following abbreviations of R&S® products are denoted in this document:

► R&S®CMX500 Radio Communication Tester is referred to as CMX

► R&S®CMsequencer graphical scripting interface is referred as CMsequencer

► R&S®CMsquares web based graphical user interface is referred to as CMsquares

► R&S®XLAPI scripting interface is referred to as XLAPI

It is assumed that the reader of this application note has fundamental understanding of 5G NR technology. If not, then please refer to the 5G NR eBook [1] for more detailed overview on the fundamentals, procedures, testing aspects of the technology. In addition, the reader should have hands-on experience on CMX platform. If not, please refer to CMX user manual [2].

2 Key Aspects of Maximum Data Throughput

In this chapter, we are going to tackle some key aspects that influence data throughput.

To maximize data throughput, it is all about system deployment, carrier bandwidth, modulation and optimization of the resource to reduce the overhead that in turn leave more resource allocations for the data transmission.

2.1 NR Maximum Data Throughput

Specified in TS38.306 [3], the maximum supported theoretical NR downlink data rate is shown in Equation 1

$$\text{data rate (in Mbps)} = 10^{-6} \cdot \sum_{j=1}^{J} \left(v_{Layers}^{(j)} \cdot Q_m^{(j)} \cdot f^{(j)} \cdot R_{\max} \cdot \frac{N_{PRB}^{BW(j),\mu} \cdot 12}{T_s^{\mu}} \cdot \left(1 - OH^{(j)}\right) \right)$$

Equation 1 NR maximum theoretical downlink data rate (TS38.306 [3])

Parameters	Description
J	The number of the aggregated component carriers (CC) or band combinations
$v_{Layers}^{(j)}$	Maximum supported MIMO layers for j-th CC
$Q_m^{(j)}$	Maximum supported modulation order for j-th CC, 2 (QPSK), 4 (16QAM), 6 (64QAM), 8 (256QAM)
$f^{(j)}$	Scaling factor for j-th CC. Value can be selected among 1, 0.8, 0.75 and 0.4.
R_{max}	A fixed number 948/1024 = 0.926.
$N_{PRB}^{BW(j),\mu}$	Maximum RB allocation in the BW with numerology μ for j-th CC, refer Table 5.3.2-1 in TS38.101-1 [4] and TS38.101-2 [5]
$BW(j)$	Maximum UE supported bandwidth for j-th CC
μ	Numerology μ=0 (15kHz SCS), 1 (30kHz SCS), 2 (60kHz SCS), 3 (120kHz SCS), 4 (240kHz SCS) [6]
T_s^{μ}	Average OFDM symbol duration in a subframe with normal cyclic prefix for numerology μ, $T_s^{\mu} = \frac{10^{-3}}{14 \cdot 2^{\mu}}$
$OH^{(j)}$	Frequency range and direction of data transmission dependent overhead factor. 0.14 (FR1, DL), 0.18 (FR2, DL), 0.08 (FR1, UL), 0.10 (FR2, UL)

Table 1 Description of parameters of the data rate calculation

Scaling factor $f^{(j)}$ is an optional UE capability that reflects the association of the maximum number of layers and maximum modulation order with the band combination [7]. It can take the values 0.4, 0.75, 0.8 and 1. Without specifying the scaling factor in UE capability, the default value 1 is applied.

The R_{max} value 948 given in Table 1 is the uttermost target code rate from 3GPP TS 38.214 [8], Table 5.1.3.1-1 and Table 5.1.3.1-2. The actual applied value should be adopted based on the UE capability which is linked to the Modulation Coding Scheme (MCS) of the associated MCS table, e.g. MCS 19 of Table 5.1.3.1-1 (64QAM MCS table) in 3GPP TS 38.214 [8] indicates the target code rate of 517.

The Overhead (OH) factor $OH^{(j)}$ is introduced to reflect the overhead in downlink PHY signaling, incl. SSB, TRS, PDCCH, DMRS, PTRS and CSI-RS.

For a given numerology (subcarrier spacing SCS), 3GPP TS38.101-1 [4] Table 5.3.2-1 defines the maximum Resource Block (RB) allocation (N_{RB}) with respect to corresponding transmission bandwidth (see in Table 2).

SCS (kHz)	5 MHz	10 MHz	15 MHz	20 MHz	25 MHz	30 MHz	40 MHz	50 MHz	60 MHz	70 MHz	80 MHz	90 MHz	100 MHz
	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}	N_{RB}
15	25	52	79	106	133	160	216	270	N/A	N/A	N/A	N/A	N/A
30	11	24	38	51	65	78	106	133	162	189	217	245	273
60	N/A	11	18	24	31	38	51	65	79	93	107	121	135

Table 2 Maximum resource blocks (TS38.101-1 [4] Table 5.3.2-1)

As defined by 3GPP TS38.101-1 [4] Table 5.3.5-1, each NR band has its individual support of the channel bandwidth with associated SCS. An example shown in Table 3 (extracted from TS38.101-1 [4] Table 5.3.5-1) indicates the support of the channel bandwidth with respect to Subcarrier Spacing (SCS) in NR band n78.

NR Band	SCS (kHz)	UE Channel bandwidth (MHz)												
		5	10	15	20	25	30	40	50	60	70	80	90	100
n78	15		10	15	20	25	30	40	50					
	30		10	15	20	25	30	40	50	60	70	80	90	100
	60		10	15	20	25	30	40	50	60	70	80	90	100

Table 3 Channel bandwidth of NR band n78 (extracted from TS38.101-1 [4], Table 5.3.5-1)

By giving following parameters in Equation 1,

- Number of carriers (J) = 1
- Number of layers (v) = 2
- Modulation (Q = 8) = 256QAM
- MCS = 27 (R_{max} = 948/1024)
- Scaling factor (f) = 1
- Band = n78
- Subcarrier spacing SCS (μ) = 30 kHz (μ = 1)
- Bandwidth = 100 MHz
- Maximum RB allocation = 273
- Overhead = 0.14

The calculated theoretical maximum data throughput = 10^{-6} * 2 * 8 * 1 * (948/1024) * (273 * 12) * (14 * 2^1) / 10^{-3} * (1 - 0.14) = 1168 Mbps

The above described maximum downlink throughput calculation assumes the full occupation of DL slots allocation within each radio frame, i.e. FDD. For TDD operation, due to the time duplexing nature of UL and DL slots allocation within a radio frame, the maximum downlink data throughput is decreased proportionally. In addition, UE supported highest MCS needs to be considered by the calculation that affects the $\boldsymbol{R_{max}}$ value. All in all, the throughput calculated by Equation 1 servers as a theoretical upper bound.

On CMX, the actual scheduled throughput can deviate from the theoretical value due to the applied scheduling configurations, e.g. TDD DL-UL pattern, number of PDSCH DMRS positions etc.

2.2 Optimization of Resource Allocations

Like LTE, Physical Downlink Shared Channel (PDSCH) is shared between different users in NR. Certain amount of the RBs are allocated by the eNB to each user. In order to verify the FR1 TDD downlink maximum throughput, the resources allocation of PDSCH need to be maximized. To achieve this, following aspects need to be considered.

- Position of Synchronization Signal Block (SSB) and Control Resource Set (CORESET)
- TDD slot configuration
- Optimization of PDCCH
- Optimization of PDSCH / PDSCH DMRS
- Optimization of PUCCH / PUSCH
- Mixed slot activation

► Combined HARQ

Additional considerations to boost the throughput are

► LTE: Optimization

► MIMO / Dual Connectivity / Carrier Aggregation

2.2.1 Position of SSB and CORESET0

SSB

Synchronization Signal Block (SSB) comprises the primary synchronization signals (PSS), secondary synchronization signal (SSS) and PBCH (including PBCH data and PBCH DMRS).

Each SSB occupies 240 subcarriers (SC) that corresponds to 20 RBs in frequency domain and 4 OFDM symbols in time domain as shown in Figure 1.

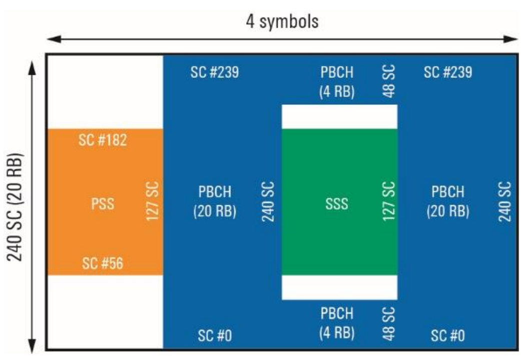


Figure 1 NR Synchronization Signal Block (SSB) [1]

SSB transmission in time domain is based on the SSB pattern (Case A to E) according to Table 4 where maximum number of SSB transmission per half frame is defined. The periodicity of the SSB pattern can be 5ms, 10ms, 20ms, 40ms, 80ms or 160ms. However, UE assumes 20ms as a default periodicity during the initial cell search.

	Subcarrier Spacing (SCS) (kHz)	OFDM start symbol position within a half frame	Value set of n	Maximum number of SSB per half frame Lmax	Frequency Range
Case A	15	{2, 8} + 14n	n = 0 ,1	4	FR1: f <=3 GHz
			n = 0,1,2,3	8	FR1: f > 3 GHz
Case B	30	{4, 8, 16, 20} + 28n	n = 0	4	FR1: f <=3 GHz
			n = 0 ,1	8	FR1: f > 3 GHz
Case C (FDD)	30	{2, 8} + 14n	n = 0 ,1	4	FR1: f <=3 GHz
			n = 0,1,2,3	8	FR1: f > 3 GHz
Case C (TDD)	30	{2, 8} + 14n	n = 0 ,1	4	FR1: f <=2.3 GHz
			n = 0,1,2,3	8	FR1: f > 2.3 GHz
Case D	120	{4, 8, 16, 20} + 28n	n = 0,1,2,3,5,6,7,8,10,11,12,13,15,16,17,18	64	FR2
Case E	240	{8, 12, 16, 20, 32, 36, 40, 44} + 56n	n = 0,1,2,3,5,6,7,8	64	FR2

Table 4 SSB transmission pattern

Although the size of SSB (Figure 1) is fixed, the number of transmission positions within the defined SSB transmission pattern can be limited through a SSB bitmap which is signaled to UE in IE `ssb-PositionsInBurst` of SIB1 and RRC (RRC reconfiguration) message in NR standalone (SA) and non-standalone (NSA) mode, respectively. As illustrated in Figure 2, we leave per default only 2nd position in SSB burst to be activated on CMX.

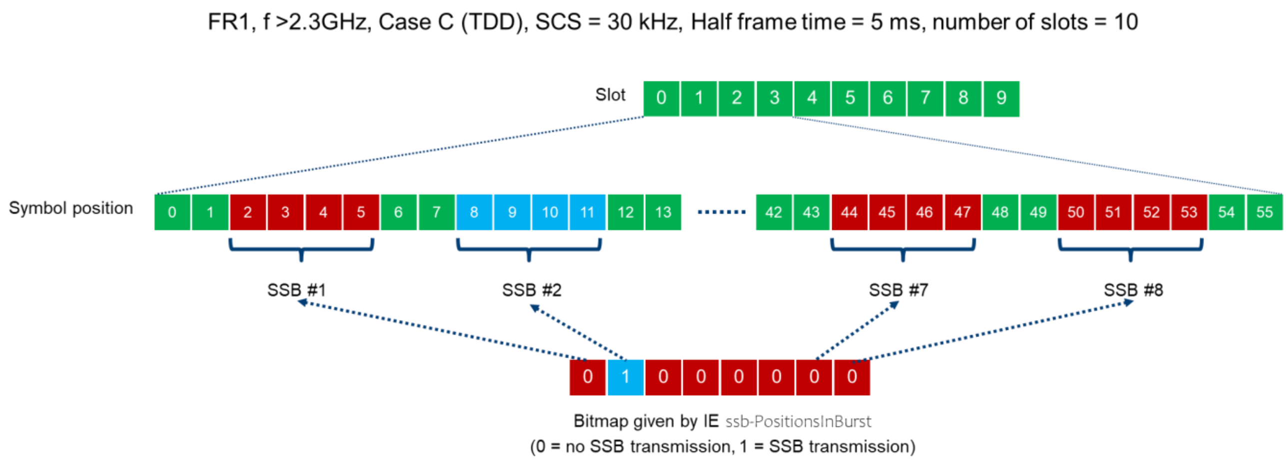


Figure 2 An example of SSB transmission bitmap in NSA mode

Whereas in the frequency domain, the SSB should be positioned to the lower edge of the bandwidth part (BWP) as close as possible for maximum throughput testing. This condition is automatically met as long as the predefined test frequency range (Mid, Low or High) is chosen on CMX. Details are described below.

CORESET/CORESET0

Physical downlink control channel PDCCH carries the DCI (Downlink Control Information) which indicates the DL/UL resources for PDSCH/PUSCH and in addition, the slot format, PUSCH and PUCCH power control command and SRS power control command.

Control Resource Set (CORESET) is a region in the resource grid where PDCCH is located. The combination of time and frequency domain resource determines the CORESET dimension. The UE can be configured in total up to 12 CORESETs in a serving cell which is identified by its index from 0 to 11.

CORESET with index 0 (CORESET0) is a special CORESET carrying DCI for decoding the SIB1 before RRC connection is established. Therefore, a normal CORESET can be configured by RRC while CORESET0 is configured by predefined parameters. Its frequency domain assignment is not like other CORESETs on the 6 RBs grid rather determined relatively to the SSB.

If SIB1 is presented, then the `controlResourceSetZero` and `searchSpaceZero` contained in the IE `PDCCH Config SIB1` of MIB (see Figure 3) are the index number within the applicable look-up tables[1] in 3GPP TS38.213 Chapter 13 [9]. Out of the look-up table, the size of CORESET0 in the resource grid (number of RBs, RB offset to SSB and number of symbols) is determined by the index number given by `controlResourceSetZero`. CORESTO monitoring occasions (system frame number and slot information that UE needs to know to monitor the PDCCH) is determined by `searchSpaceZero`. For more readings, refer to 3GPP TS38.213 Chapter 13 [9].

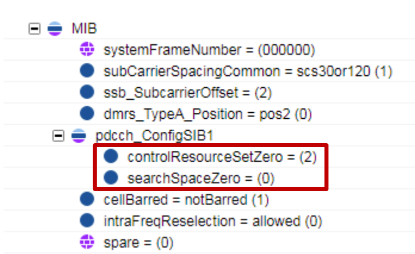


Figure 3 Example of CORESET0 configuration in MIB

[1] The selection of look-up table depends on the SSB SCS, PDCCH SCS and minimum bandwidth of the operating band.

Figure 3 gives an example of CORESET0 configuration (`controlResourceSetZero = 2`) in MIB. By looking up the Table 13-4[2] in 38.213 [9], configurations given in Table 5 are linked to index 2 are applied where offset RBs refers to the number of RBs relative to the lowest start RB of SSB. The other way around, if the CORESET0 should start at RB 0 to align with the lower edge of the BWP in order to leave more resources allocated for PDSCH, then the SSB should start at RB 2.

Index	SS/PBCH block and CORESET multiplexing pattern	Number of RBs $N_{RB}^{CORESET}$	Number of Symbols $N_{symb}^{CORESET}$	Offset (RBs)
2	1	24	2	2

Table 5 CORESTO configuration in Table 13-4, 3GPP 38.213 [9] with controlResourceSetZero = 2

Selecting an arbitrary carrier frequency of an NR band and configuring the parameters in such a way that SSB and CORESET0 are positioned in the edge of the BWP is a time tedious task and often failure prone. To cope with this challenge, predefined 3GPP test frequency range (Mid, Low or High) can be utilized. The standardized test frequencies of each individual NR band can be found in TS38.508-1 [10], Table 4.3.1.1.1.x-y, where placeholder x indicates the NR band, y indicates the associated SCS (1=15 kHz SCS, 2=30 kHz SCS, 3=60 kHz SCS). In each table, the listed parameters associated to 'Mid', 'Low' or 'High' test frequency range ensures that the SSB and CORESET0 are positioned to the lower edge of the BWP as close as possible and RB offset between SSB and CORESET0 is considered as well.

As an example, Table 6 is the test frequency definitions extracted from TS38.508-1 [10], Table 4.3.1.1.1.78-2 for n78, 30 kHz SCS.

[2] This table is selected based on configuration (SSB SCS 30 kHz, PDCCH SCS 30 kHz, 10 MHz min bandwidth for band n78 defined in 3GPP TS38.101-1 Table 5.3.5-1 [4])

Bandwidth [MHz]	*carrierBandwidth* [PRBs]	Range		Carrier centre [MHz]	Carrier centre [ARFCN]	point A [MHz]	*absoluteFrequencyPointA* [ARFCN]	*offsetToCarrier* [Carrier PRBs]	SS block SCS [kHz]	GSCN	*absoluteFrequencySSB* [ARFCN]	k_{SSB}	CORESET#0 Offset [RBs] Note 1	CORESET#0 Index Note 1	offsetToPointA (SIB1) [PRBs] Note 1
10	24	Downlink	Low	3305.01	620334	3300.69	620046	0	30	7711	620352	18	2	2	4
		&	Mid	3549.99	636666	3508.95	633930	102		7881	636672	6	2	2	208
		Uplink	High	3795	653000	3609.24	640616	504		8051	652992	16	1	1	1010
15	38	Downlink	Low	3307.5	620500	3300.66	620044	0	30	7711	620352	20	2	2	4
		&	Mid	3549.99	636666	3506.43	633762	102		7879	636480	6	1	1	206
		Uplink	High	3792.48	652832	3604.2	640280	504		8048	652704	16	3	3	1014
20	51	Downlink	Low	3310.02	620668	3300.84	620056	0	30	7711	620352	8	2	2	4
		&	Mid	3549.99	636666	3504.09	633606	102		7878	636384	18	3	3	210
		Uplink	High	3789.99	652666	3599.37	639958	504		8044	652320	2	1	1	1010
40	106	Downlink	Low	3320.01	621334	3300.93	620062	0	30	7711	620352	2	2	2	4
		&	Mid	3549.99	636666	3494.19	632946	102		7871	635712	6	3	3	210
		Uplink	High	3780	652000	3579.48	638632	504		8030	650976	8	0	0	1008
50	133	Downlink	Low	3325.02	621668	3301.08	620072	0	30	7711	620352	16	1	1	2
		&	Mid	3549.99	636666	3489.33	632622	102		7867	635328	18	0	0	204
		Uplink	High	3774.99	651666	3569.61	637974	504		8024	650400	18	3	3	1014
60	162	Downlink	Low	3330	622000	3300.84	620056	0	30	7711	620352	8	2	2	4
		&	Mid	3549.99	636666	3484.11	632274	102		7864	635040	6	3	3	210
		Uplink	High	3769.98	651332	3559.38	637292	504		8016	649632	4	0	0	1008
80	217	Downlink	Low	3340.02	622668	3300.96	620064	0	30	7711	620352	0	2	2	4
		&	Mid	3549.99	636666	3474.21	631614	102		7857	634368	18	2	2	208
		Uplink	High	3759.99	650666	3539.49	635966	504		8003	648384	10	3	3	1014
90	245	Downlink	Low	3345	623000	3300.9	620060	0	30	7711	620352	4	2	2	4
		&	Mid	3549.99	636666	3469.17	631278	102		7853	633984	18	0	0	204
		Uplink	High	3754.98	650332	3529.44	635296	504		7996	647712	8	3	3	1014
100	273	Downlink	Low	3350.01	623334	3300.87	620058	0	30	7711	620352	6	2	2	4
		&	Mid	3549.99	636666	3464.13	630942	102		7850	633696	18	2	2	208
		Uplink	High	3750	650000	3519.42	634628	504		7989	647040	4	3	3	1014

Note 1: The CORESET#0 Index and the associated CORESET#0 Offset refers to Table 13-4 in TS 38.213 [22]. The value of CORESET#0 Index is signalled in the four most significant bits of the IE pdcch-ConfigSIB1 in the MIB. The offsetToPointA IE is expressed in units of resource blocks assuming 15 kHz subcarrier spacing for FR1 and 60 kHz subcarrier spacing for FR2.

Table 6 Test frequencies for NR band n78 and SCS 30 kHz (TS38.508-1, Table 4.3.1.1.1.78-2 [10])

Thus, following parameters need to be set exemplary on CMX to position SSB and CORESET0 for the max downlink throughput testing:

- ► Frequency Range, e.g. FR1
- ► Duplex mode, e.g. TDD
- ► Frequency Band Indicator, e.g. N78
- ► Subcarrier Spacing, e.g. 30 kHz
- ► Carrier Bandwidth, e.g. 100 MHz (set DUT's maximum supported bandwidth)
- ► Range, choose one of the predefined ranges 'Mid', 'High' or 'Low'.

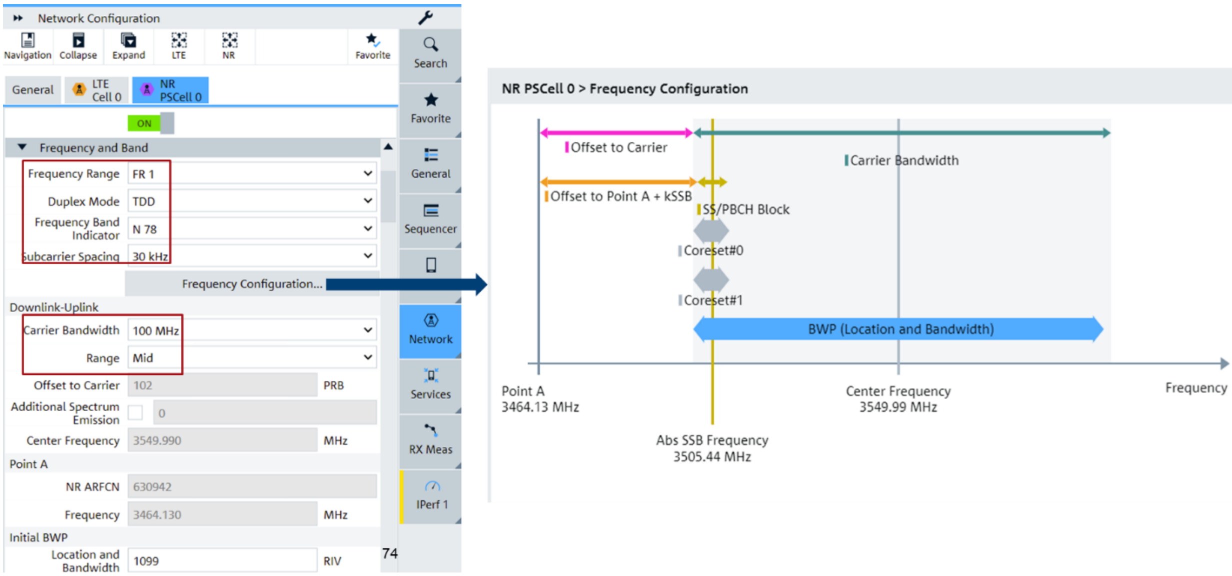


Figure 4 Frequency settings and visualization of frequency config on CMX

Figure 4 highlights the frequency settings in CMsquares. Intuitively, the actual position of SSB and CORESET in frequency domain of the BWP is visualized in the 'Frequency Configuration' window. As we can clearly see that both of them are located at the lower edge of the BWP.

2.2.2 TDD DL-UL Pattern

5G NR provides high flexibility to configure the transmission and receiving in time domain in TDD mode. The parameters `dl-UL-TransmissionPeriodicity`, `nrofDownlinkSlots`, `nrofDownlinkSymbols`, `nrofUplinkSlots` and `nrofUplinkSymbols` determine the TDD pattern for UL and DL allocation in a radio frame.

The configuration in Table 7 enables maximum downlink resource allocation for PDSCH which is essential for downlink maximum throughput testing in TDD mode. Whereas the number of uplink slots and symbols are kept on the minimum level for just sending the ACK/NACK in the uplink direction for the HARQ process.

Parameter	Value	Description
dl-UL-TransmissionPeriodicity	5 ms (10 slots)	Periodicity of TDD DL/UL pattern
nrofDownlinkSlots	8	Number of consecutive full DL slots at the beginning of each DL-UL pattern
nrofUplinkSlots	1	Number of consecutive full UL slots at the end of each DL-UL pattern
nrofDownlinkSymbols	12	Number of consecutive DL symbols in the beginning of the slot following the last full DL slot
nrofUplinkSymbols	1	Number of consecutive UL symbols in the end of the slot preceding the first full UL slot

Table 7 TDD DL/UL pattern for max downlink throughput

TDD DL-UL pattern configuration given in Table 7 is illustrated in Figure 5 where slot 0 to slot 7 are scheduled for downlink, slot 8 is a partial slot and last slot 9 is for uplink. In partial slot 8, the first 12 symbols are for downlink, the last symbol is for uplink.

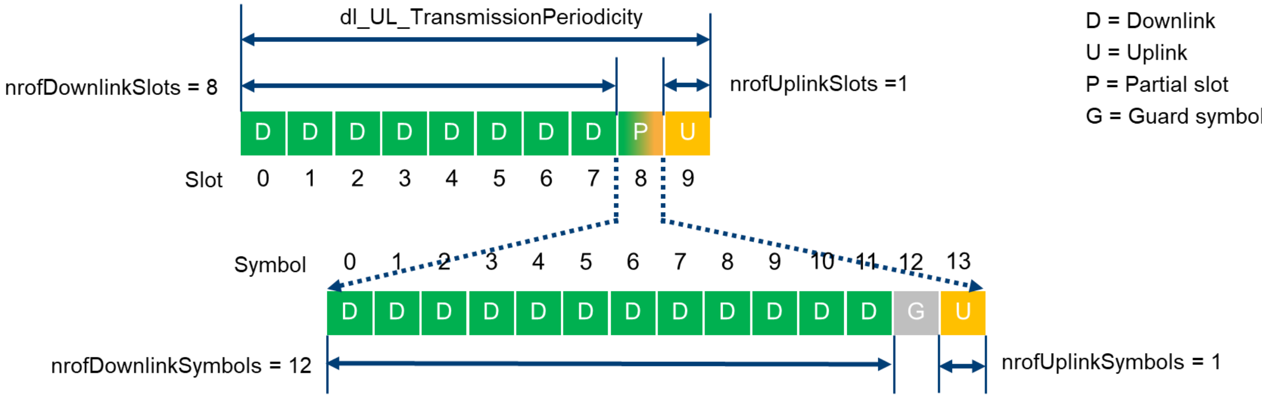


Figure 5 TDD DL/UL pattern according to configuration of Table 7

In CMsquares, TDD DL-UL pattern can be configured as shown in Figure 6.

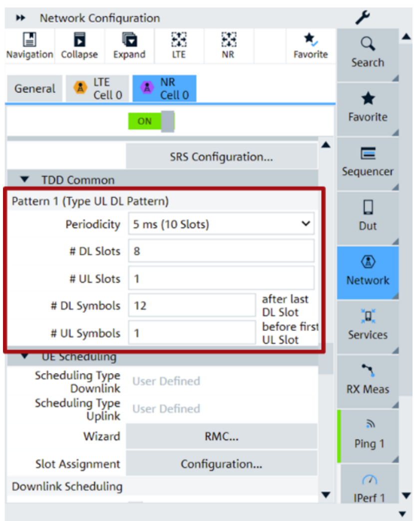


Figure 6 Configuration of TDD DL-UL pattern in CMsquares

2.2.3 Optimization of PDCCH

Optimization of PDCCH is performed in both time and frequency domain to reduce its dimension in the resource grid.

Aggregation level (AL) determines the size of a Search Space (SS). In Table 8, a list of ALs and their mapping with respect to control channel elements (CCEs), RE, resource element groups (REGs) and PDCCH size is given. For scheduling PDSCH, DCI format 1_1 is used. We apply here AL2 to convey DCI 1_1 with considerable good forward error correction performance.

Aggregation Level	Number of CCEs	Number of RE	Number of REG	PDCCH (QPSK modulated, excl. DMRS) (bits)
1	1	72	6	108
2	2	144	12	216
4	4	288	24	432
8	8	576	48	864
16	16	1152	96	1728

Table 8 UE specific Search Space (USS) Aggregation Levels

In the time domain, PDCCH is reduced to 1 OFDM symbol. This assignment is implicitly configured through PDSCH resource allocation in time domain (see 2.2.4 for details).

AL can be set in CMsquares as shown in Figure 7.

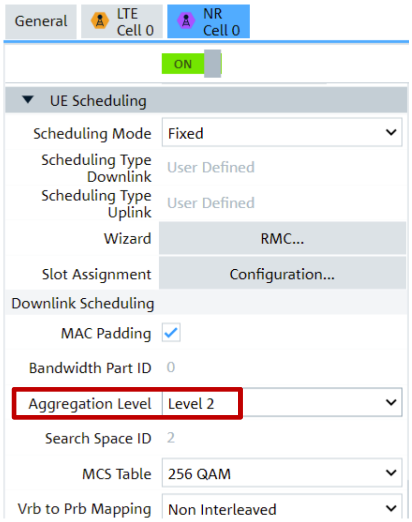


Figure 7 Aggregation Level (AL) setting for PDCCH in CMsquares

Optimization of PDCCH in both time and frequency domain is illustrated in Figure 8.

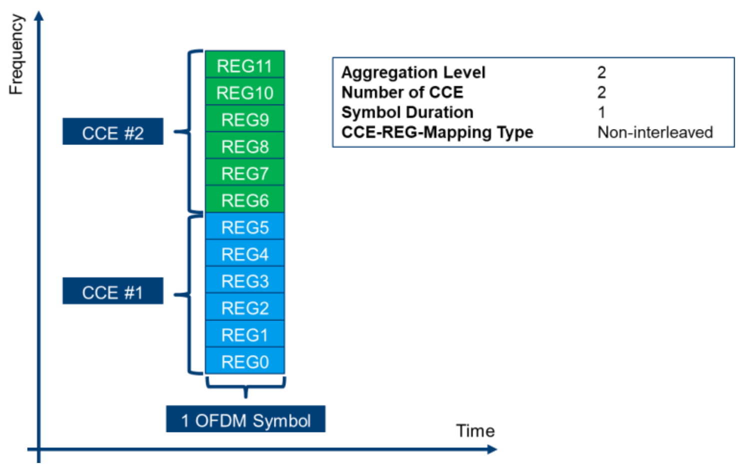


Figure 8 PDCCH optimization (Aggregation Level = 2, Symbol duration = 1)

2.2.4 Optimization of PDSCH

Recall that TDD DL-UL pattern is configured for max downlink throughput testing according to the parameter settings mentioned in chapter 2.2.2.

Now, we will get a level deeper to see how to optimize PDSCH resource allocation in time domain with respect to the assigned number of symbols in each slot.

In time domain, following aspects have to be taken into accounts:

► For all downlink slots (slot 0 to slot 7), reserve 1 symbol (symbol index 0) for PDCCH to reduce the control overhead (see 2.2.3). The remaining 13 symbols (starting from symbol index 1) are allocated for PDSCH. Figure 9 depicts the time domain resource assignment for TDD DL slots.

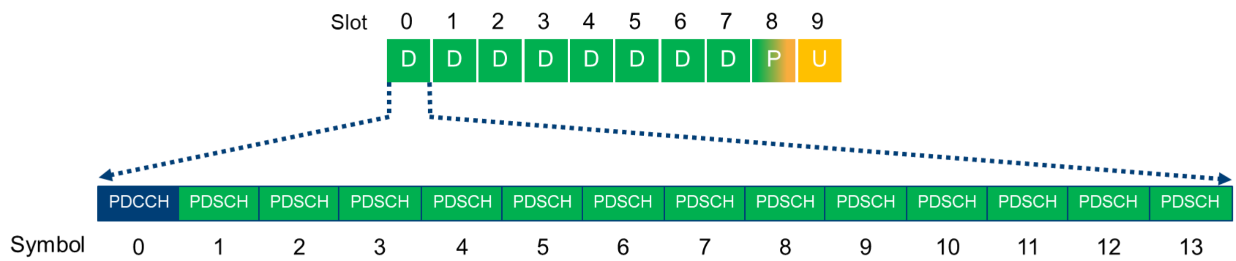


Figure 9 TDD downlink slot configuration for maximum downlink throughput

► In partial slot 8, use 12 DL symbols (symbol 0 is reserved for PDCCH), 1 UL symbol (the last symbol for PUCCH). One symbol (symbol 12) remains unassigned which serves as a guard period. The guard period is necessary for a TDD system to ensure enough switching time from DL to UL transmission. Time domain assignment of partial slot 8 is illustrated in Figure 10 below.

  Due to the internal processing time of individual UE, the optimization by utilizing the partial slot 8 for TDD is not always possible. Only UE with high processing performance can apply this kind of optimization. Therefore, this is optional and can be activated or deactivated in UE scheduling configuration in CMsquares if required.

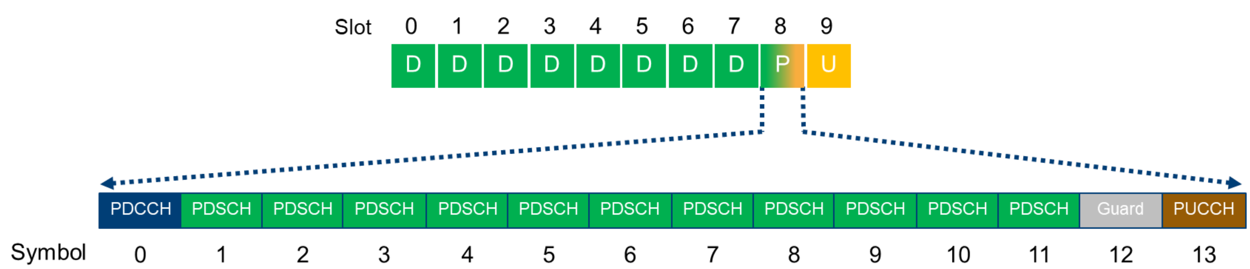


Figure 10 TDD partial slot configuration for maximum downlink throughput

In frequency domain, the maximum available number of RBs are determined by the channel bandwidth and SCS. For example, if 100 MHz channel bandwidth and 30 kHz SCS is considered, then max 273 RBs can be assigned according to Table 2 on Page 4.

► For DL slot 1 to slot 7, complete 273 RBs are allocated to PDSCH without exceptions.

► DL slot 0 should be scheduled carefully. Based on the CORESET0 predefined configuration (see Chapter 2.2.1), SSB start RB is 2 (see Table 6, considered example: n78, 30 kHz SCS, 100 MHz CBW, 'Mid' frequency range). SSB itself occupies 20 RBs. Therefore, the earliest start RB for PDSCH should be 23. Thus, the remaining maximum number of RBs for PDSCH in slot 0 are 250. According to SSB position bitmap in time domain (see the example in Chapter 2.2.1), SSB occupies 8th, 9th, 10th and 11th symbol of slot 0. By taking all these facts into account, resources in slot 0 are now assigned as shown in Figure 11 below.

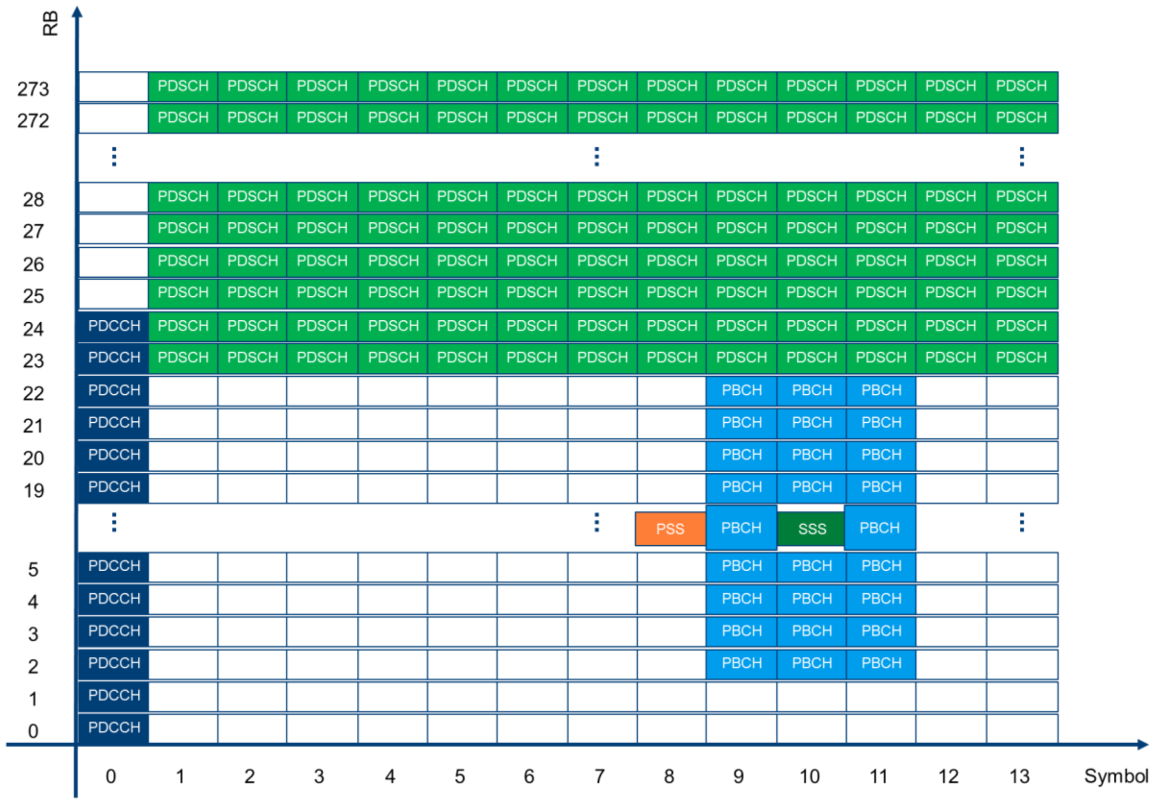


Figure 11 Resource allocation of Slot 0 (symbol 0 is reserved for PDCCH)[3]

- Schedule DL data in slot 0-8 with possible MIMO scheme and maximum MCS 27 if '256 QAM' MCS Table is applied. These settings are UE capability dependent.
- Allow minimum number of PDSCH DMRS. Try to adopt no additional PDSCH DMRS positions (set dmrs_AdditionalPosition to pos0)[4]

Figure 12 shows an example of PDSCH settings in CMsquares based on the optimization considerations explained in above text with few exceptions. MCS 19 is chosen in this case which reflexes maximal MCS support of the DUT being tested. Mixed slot 8 is deactivated which is limited by the used DUT too.

[3] For sake of simplicity, the occupation of DMRS is not illustrated in the graph.
[4] No additional PDSCH DMRS position (Pos 0) may cause higher BLER for UE with bad receiver performance. In this case, one additional PDSCH DMRS position (Pos 1) needs to be configured which will then reduce the scheduled throughput.

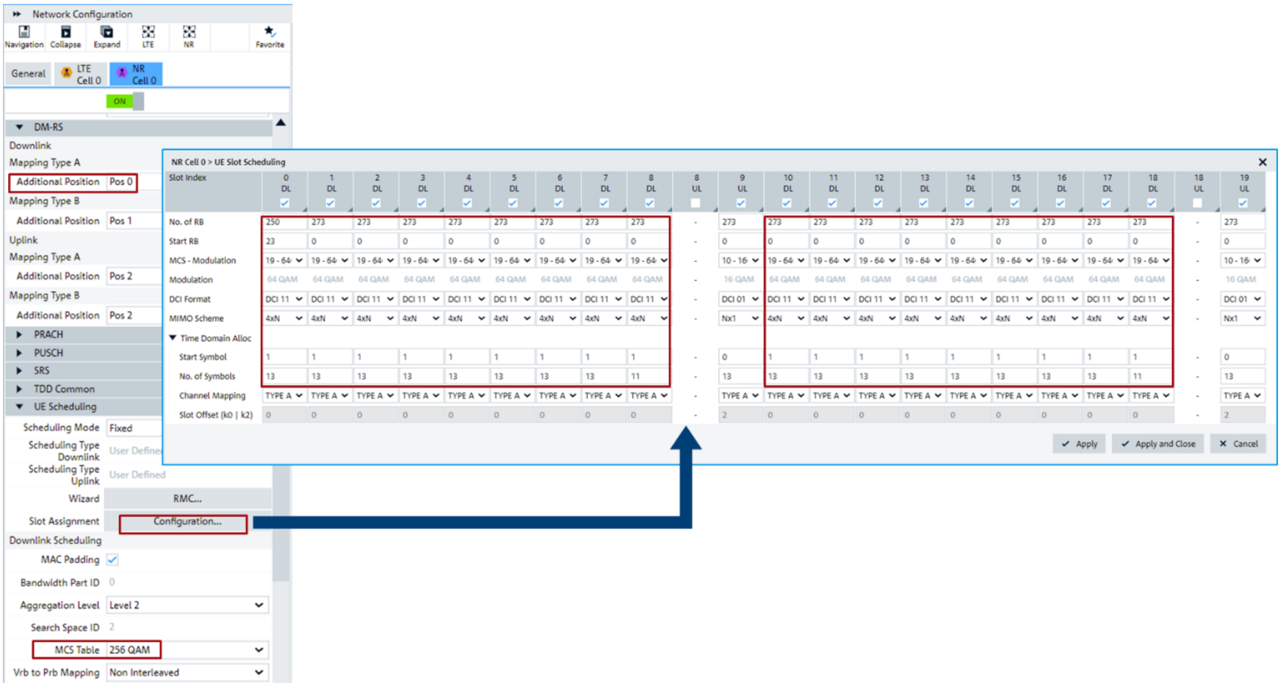


Figure 12 PDSCH scheduling in CMsquares with activated partial slot 8

2.2.5 Optimization of PUCCH/PUSCH

PUCCH/PUSCH is used for the UE to send uplink control information (UCI) to gNB.

UCI contains following contents

- ACK/NACK of PDSCH
- Scheduling request (SR)
- Channel status indicator (CSI)

For DL throughput testing, resources allocated for UL channels need to keep on a minimal level. Therefore, followings are considered for PUCCH/PUSCH optimization:

- Short PUCCH is used (PUCCH format 2) which allows only 1 or 2 OFDM symbols.
- Set the last symbol in partial slot 8 for PUCCH, see also Figure 10
- Set PUSCH in slot 9 to 13 symbols (symbol 0-12) and set last symbol for PUCCH. The resource allocation of UL slot 9 is illustrated below in Figure 13.

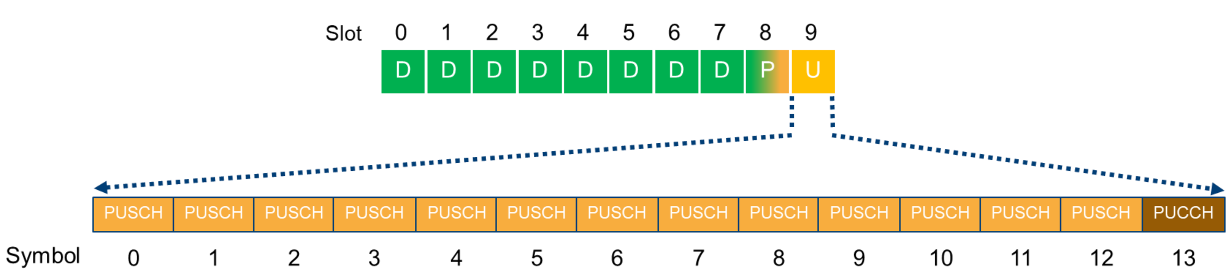


Figure 13 TDD uplink slot configuration for maximum downlink throughput

2.2.6 Modulation

Several modulation types are defined in NR to be used for PDSCH, i.e. QPSK, 16QAM, 64QAM and 256QAM that corresponds to modulation order 2, 4, 6 and 8, respectively. 3GPP 38.214 [8], Chapter 5.1.3 specifies three MCS tables (MCS index table 1, 2 and 3). The combination of MCS table and MCS index determines the modulation type/modulation order and the target code rate of the PDSCH.

To perform maximum throughput testing, proper selecting of MCS table in CMsquares (see Figure 14) and choosing of MCS index in the UE scheduling configuration (see Figure 12, e.g. MCS 19) for PDSCH is important to achieve the maximum target code rate. As a rule of thumb, to reach higher DL throughput, always select higher modulation order. However, drawback is the higher vulnerability that is being introduced as the so-called Euclidian distance shrinks and the bit error rate probability rises. Thus, the caused data retransmissions will negate the gain obtained from higher order modulation. A trade-off between modulation order and number of retransmissions due to bit error rate has to be considered. Nice MCS tuning feature is developed in CMsquares to explore the optimal MCS index (see 3.5.2 for the details). Apparently, the selection of MCS table and MCS index depends on the UE capability.

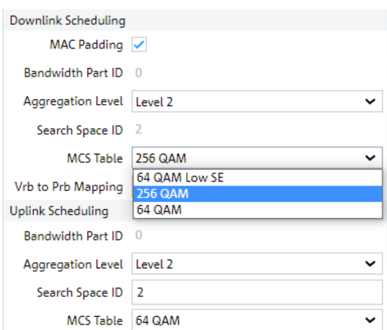


Figure 14 MCS table selection in CMsquares

Table 9 provides the mapping information between 3GPP specified MCS tables and MCS table selections in CMsquares. For more information about MCS, please refer to 3GPP 38.214 [8] Chapter 5.1.3.

3GPP 38.214 [8] MCS Table for PDSCH	MCS Table in CMsquares
Table 5.1.3.1-1: MCS index table 1	64QAM
Table 5.1.3.1-2: MCS index table 2	256QAM
Table 5.1.3.1-3: MCS index table 3	64QAM Low SE

Table 9 Mapping of MCS tables defined in 3GPP 38.214 [8] and MCS tables in CMsquares

2.2.7 Hybrid Automatic Repeat Request

Hybrid automatic repeat request (HARQ) is a MAC protocol to improve the re-transmission. If the receiver detects the error in the received data, it will buffer the data and at the meanwhile request the transmitter for the retransmission.

Unlike the fixed HARQ ACK/NACK timing (4 ms in FDD) in LTE, NR applies a fully configurable parameter K1 to define the offset in number of slots between the DL slot where the data is scheduled on PDSCH and the reception of the associated acknowledgement (ACK/NACK) carried on PUCCH in UL.

Depending on whether partial slot 8 is activated or not, K1 is optimized in such a way that the HARQ of DL slots are aggregated and sent in one UL slot (i.e. in UL slot 9 or UL portion of the partial slot 8). This optimization is automatically performed by Maximum Throughput Wizard (see 3.4) or by running the XLAPI script (see 3.6) on CMX. Details are illustrated in Figure 15 and Figure 16 showing that the K1 assignment of each DL slot with activated and deactivated partial slot 8, respectively.

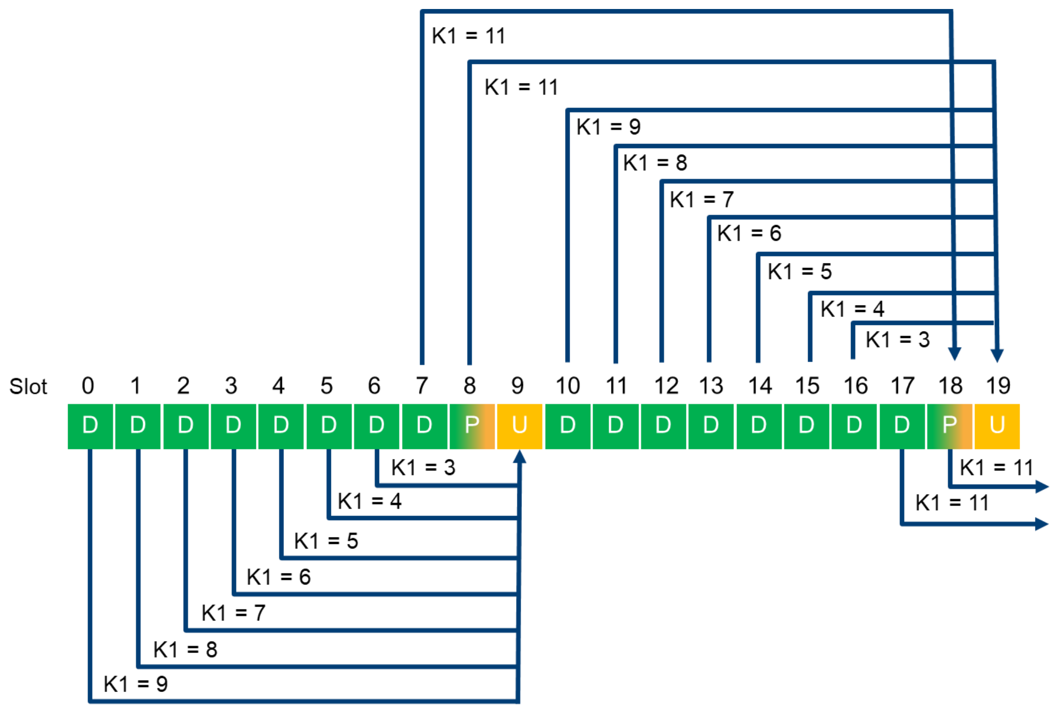


Figure 15 K1 configuration over one radio frame (20 slots, 30 kHz SCS) with activated partial slot 8

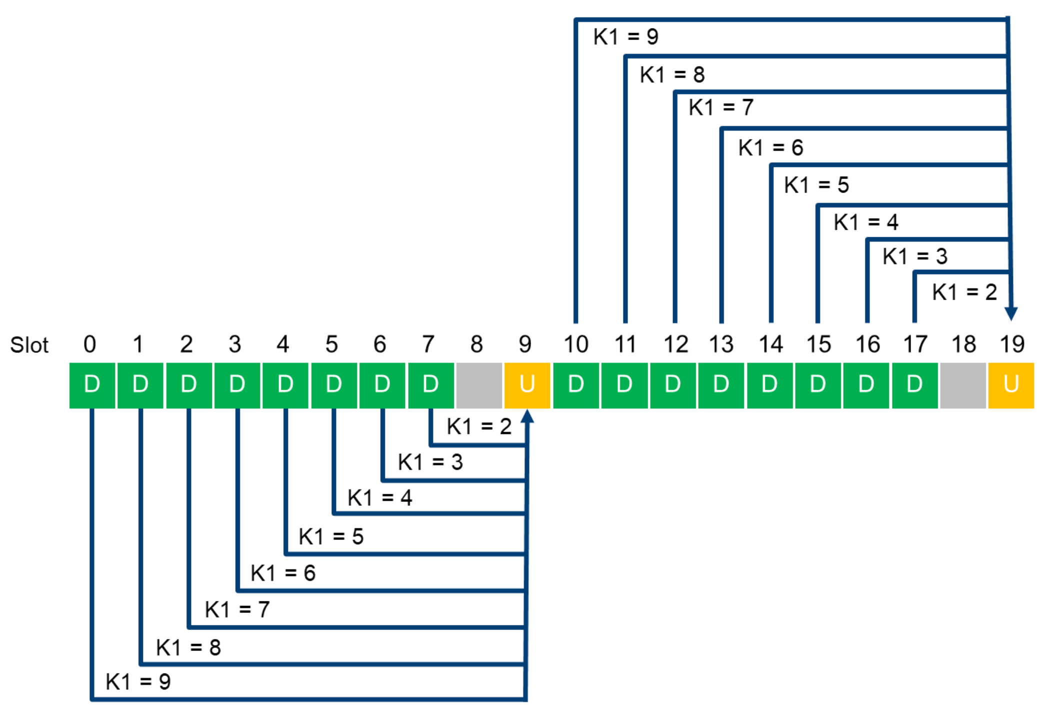


Figure 16 K1 configuration over one radio frame (20 slots, 30 kHz SCS) with deactivated partial slot 8

2.2.8 Optimization in LTE

In NSA case, LTE acts as a master node or anchor that can be utilized for data transmission in addition to NR. To maximize the data throughput over LTE, following aspects need to be considered:

- Set proper DUT supported duplex mode (FDD or TDD) and supported band.
- Set DUT supported maximum channel bandwidth and associated maximal number of RBs in LTE. E.g. 20 MHz channel bandwidth corresponds to 100RBs

► Use MIMO, e.g. use MIMO 2x2, transmission mode: spatial multiplexing

► Set highest supported MCS, e.g. MCS 27 (with 256QAM Table)

► Reduce PDCCH to 1 symbol

Figure 17 presents an overview of the configurations in LTE in CMsquares that need to be prepared for the maximum data throughput over LTE.

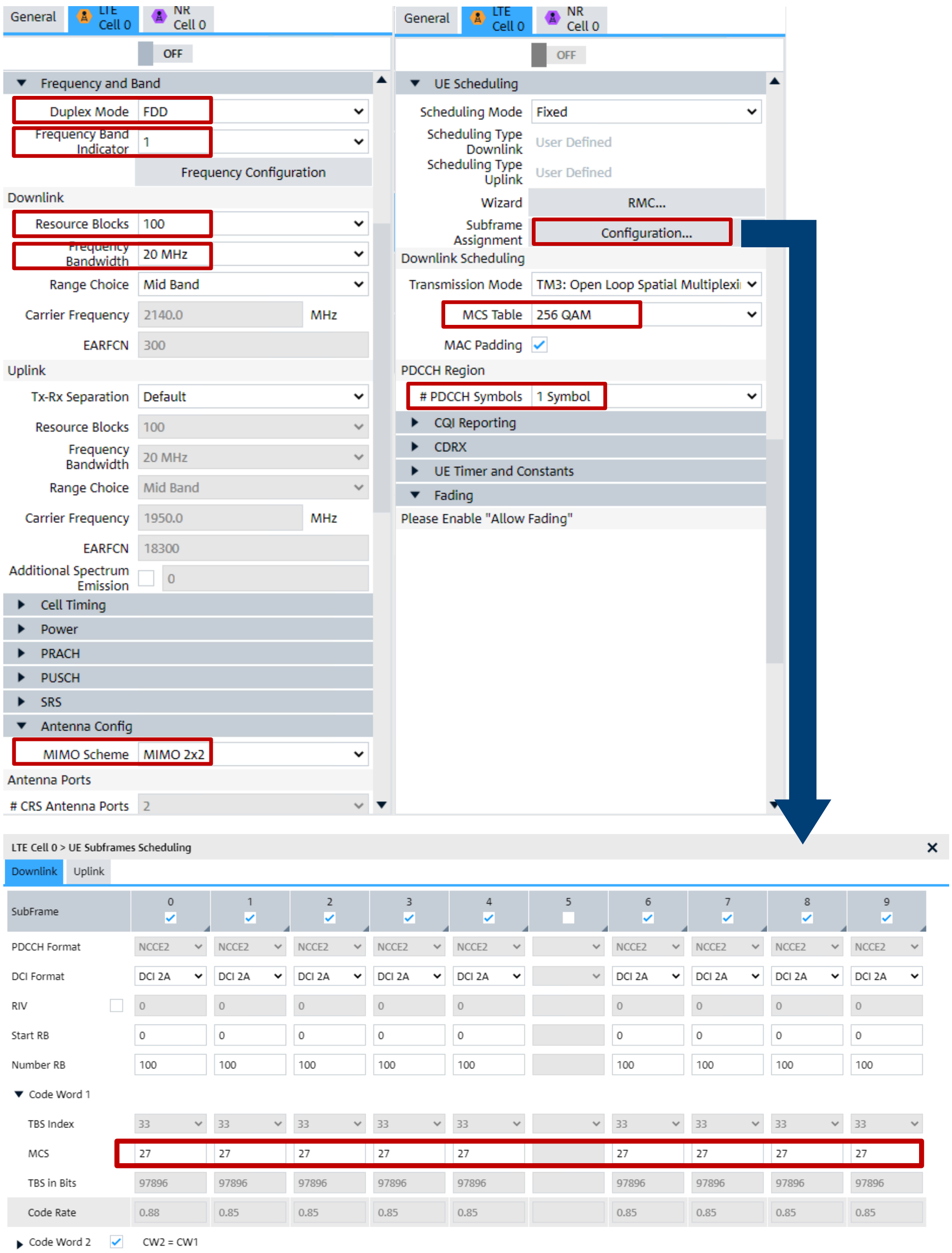


Figure 17 LTE configurations for max downlink throughput testing

Bear in mind that the split data radio bearer (SCG split bearer) needs to be selected during the EN-DC activation procedure which utilizes LTE data traffic in addition to NR. Per default, only SCG bearer without split is going to be established. For more details, refer to 2.2.9.3.

2.2.9 MIMO / Dual Connectivity / Carrier Aggregation

2.2.9.1 MIMO

MIMO is a pretty established multiple antenna technology in a modern wireless communications system, such as in NR. This allows the data signals to be sent on multiple MIMO layers simultaneously over the same radio channel. By deploying MIMO technology, higher data rate can be achieved. As given in Equation 1 on page 4, MIMO is one of the determine factors that influence the overall data throughput. Apparently, the more MIMO layers the UE supports, the higher data throughput is going to be expected.

The DL MIMO settings in CMsquares are part of the UE PDSCH scheduling as shown in Figure 12.

In this document, MIMO 2x2 is given as an example based on the DUT capability. MIMO support depends also on the hardware capacity of a test and measurement equipment (see more details in 3.1.1).

2.2.9.2 Carrier Aggregation

Carrier Aggregation (CA) is an approach to concatenate multiple component carriers to increase the channel bandwidth which then in turn enhances the overall data throughput. As indicated in Equation 1, higher number of aggregated component carriers contribute to higher data throughput in the end. There are different kinds of CA scenarios specified, i.e. inter-band contiguous, inter-band non-contiguous, intra-band. 3GPP defines the predefined particular band combinations for CA. In comparison to dual connectivity (DC) (see 2.2.9.3), CA splits the user traffic between carriers on MAC layer.

2.2.9.3 Dual Connectivity

Dual connectivity (DC) allows a UE to simultaneously connect to multiple cell groups, i.e. Master Cell Group (MCG) and Secondary Cell Group (SCG). In contrast to CA, DC has a bearer split in PDCP layer.

Typical DC deployment scenario in NR is so called ENDC operation mode in NSA mode where LTE carrier servers as MCG and NR carrier as SCG.

In ENDC operation, four dedicated radio bearer (DRB) types are defined as shown in Figure 18, i.e. MCG bearer, SCG bearer, MCS split bearer and SCG split bearer. They differ in the PDCP implementation (either NR PDCP or E-UTRA PDCP) and underneath RLC/MAC/PHY layer.

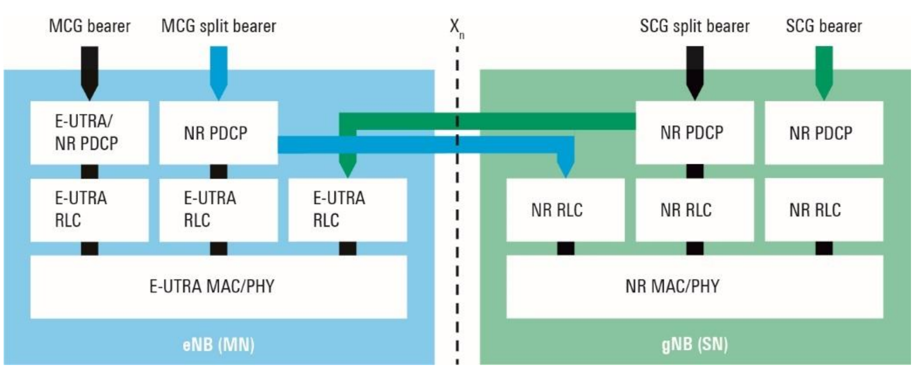


Figure 18 NR bearer concept [1]

Per higher layer signaling, a suitable bearer type can be adopted to meet the needs of applications. Peak data throughput needs to aggregate data traffics over both LTE (MCG) and NR (SCG). In this case, SCG split bearer has to be considered, since it utilizes both NR and LTE RLC/MAC/PHY layer underneath to transfer the data.

As per default, the ENDC mode on CMX establishes a SCG bearer. If desired, SCG split bearer can be established during ENDC mode activation process. The configuration steps in CMsquares are shown in Figure 19.

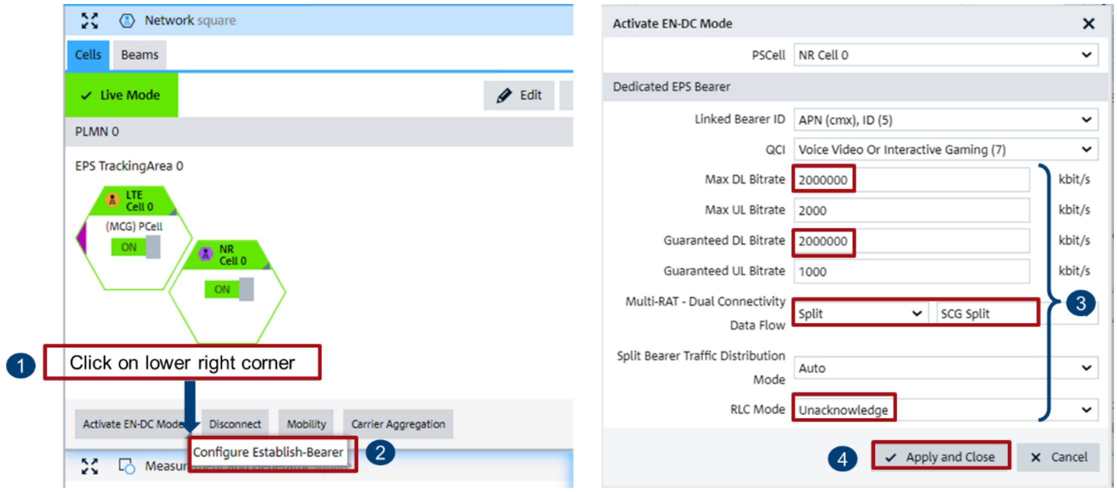


Figure 19 Configure SCG split bearer in CMsquares

1. In 'Network square', click on little triangle symbol at lower right corner of the 'Activate EN-DC Mode' button
2. Click on opened 'Configure Establish-Bearer' option
3. In 'Activate EN-DC Mode' window, choose 'SCG Split' and set max expected DL bitrate and guaranteed DL bitrate accordingly, select RLC 'Unacknowledge' mode.
4. Confirm the settings with 'Apply and Close'

3 Solutions and Tipps

CMX is a mobile radio tester that is applicable for testing a UE from different aspects, i.e. RF, Protocol, IP services etc., including the data throughput testing. It is an ideal test platform to fulfill the measurement requirements in which CMX emulates various test scenarios with configurable network parameters. There are several test approaches (Throughput Wizard, CMsequencer, XLAPI) that can be adopted on CMX for throughput testing. In this section, these test solutions are introduced.

For detailed information about the CMX operation, please refer to CMX Base user manual [2].

3.1 System Requirements

3.1.1 Hardware

As a scalable test measurement instrument, CMX is the central piece of the hardware requirement. Along the product evolution path, CMX is continually developed from the initial setup in combination with CMW, including minimum footprint configuration (1xCMX+1xCMW) right up to the maximum CMX500+CMWflexx4 setup (1xCMX+4xCMW). Figure 20 and Figure 21 shows exemplarily the minimum footprint setup and CMX500+CMWflexx2 (1xCMX+2xCMW) setup, respectively.

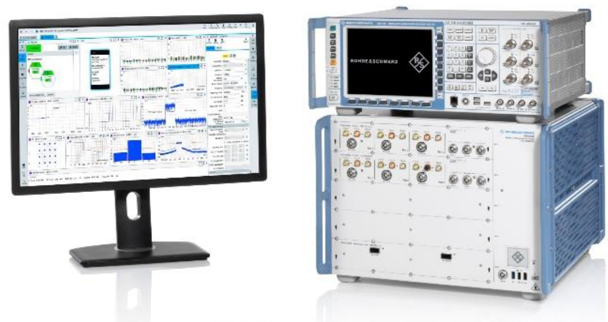


Figure 20 CMX minimum footprint setup

Figure 21 CMX500+CMWflexx2 setup

A milestone of the product development is achieved through the launch of CMX One-Box-Tester (CMX-OBT). The highly integrated CMX-OBT features LTE signaling/measurement, NR signaling/measurement and data application services which are all included in one single box (see Figure 22). Please refer to the press release of the product launch in [11].

Figure 22 CMX One-Box-Tester (CMX-OBT) setup

More information about each setup mentioned above, including the cabling etc. can be referred in [2].

To allow a particular network configuration, certain hardware setup requirement needs to be fulfilled. Table 10 below provides an overview of the network configuration versus its minimum required HW setup.

Network Configuration	Minimum Required HW Setup
LTE 4x4	Minimum footprint setup or CMX-OBT
LTE 2x2, NR 2x2	Minimum footprint setup or CMX-OBT
NR 4x4	Minimum footprint setup or CMX-OBT
LTE 4x4, NR 4x4	CMX500 + CMWflexx2 or CMX-OBT

Network Configuration	Minimum Required HW Setup
LTE 2x2, NR 4x4, NR 2x2	CMX500 + CMWflexx2 or CMX-OBT
LTE 2x2, LTE 2x2, NR 4x4	CMX500 + CMWflexx2 or CMX-OBT
NR 4x4, NR 4x4	CMX500 + CMWflexx2 or CMX-OBT
LTE 4x4, NR 4x4, NR 4x4	CMX500 + CMWflexx3 or CMX-OBT
LTE 2x2, LTE 2x2, NR 4x4, NR 4x4	CMX500 + CMWflexx3 or CMX-OBT
LTE 4x4, LTE 4x4, 2 x LTE 2x2, NR 4x4	CMX500 + CMWflexx3[7] or CMX-OBT
LTE 2x2, LTE 4x4, LTE 4x4, NR 4x4	CMX500 + CMWflexx2[7] or CMX-OBT
LTE 4x4, LTE 4x4, LTE 4x4, NR 4x4	CMX500 + CMWflexx2[7] or CMX-OBT
LTE 2x2, LTE 2x2, LTE 4x4, NR 4x4, NR 4x4	CMX500 + CMWflexx3[7] or CMX-OBT
LTE 4x4, LTE 4x4, LTE 4x4, LTE 4x4, LTE 4x4, NR 4x4	CMX500 + CMWflexx3[7] or CMX-OBT

Table 10 Summary of network configuration vs. CMX HW setup

3.1.2 Software

3.1.2.1 Composite Software

Composite Software (CSW) is a complete software package containing all the software components that are required for the operations on CMX (system control, cell configurations, measurements etc.). It includes:

- ► Web based user interface WebUI (CMsquares)
- ► Signaling testing (for protocol stack verifications, NR and legacy technologies)
- ► RF testing (for RF measurements, NR and legacy technologies)
- ► Data application testing (application servers and data service measurements)
- ► CMsequencer (generation of test scripts / test plan and test automation)
- ► XLAPI (CMX scripting interface)
- ► Message logging tool

In addition to CSW, an extra tool called Automation Manager is needed as long as the DUT control via CMsquares is desired (see more details in 3.1.2.3). DUT control is an integral part of all test solutions. With the help of Automation Manager, manual intervention with DUT is therefore waived.

3.1.2.2 Throughput Testing Tools

In this section, commonly used tools for throughput testing are described. They are either downloadable or accessible directly from Data Application Unit (DAU)[8] home page by entering DAU IP '172.22.1.201' or FQDN 'www.cmx.dau/tools' in DUT's browser provided that the DUT is in the attached state and DRB is established beforehand.

3.1.2.2.1 Traffic Generator

Traffic generator is part of the built-in web server of the DAU. It provides fast check of the UL and DL data rate.

Access: DAU home page > Traffic Generator

---

[7] Require LTE baseband combining
[8] Data Application Unit (DAU) is a piece of hardware in CMX that allows End-to-End (E2E) IP data transfer and performs user plane (U-Plane) tests for an IP connections to a DUT.

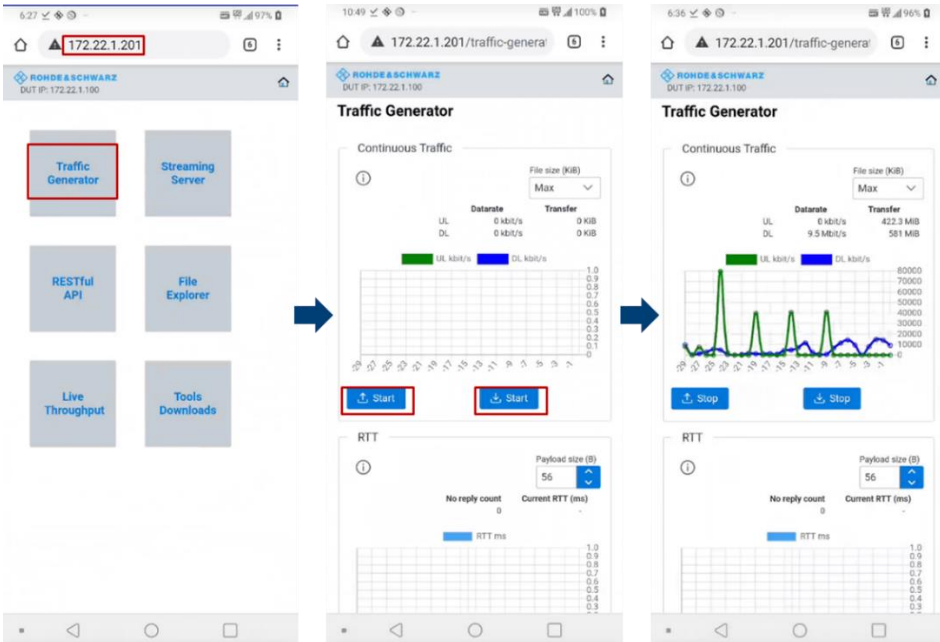


Figure 23 Traffic generator

3.1.2.2.2 iPerf2

iPerf2 is a commonly used network testing tool to verify TCP and UDP throughput in both UL and DL data connection between two communication peers. It can be operated in both server and client mode. The peer configured in server and client mode expects to receive and send the data packets, respectively. If not otherwise stated, the term iPerf is interchangeable with iPerf2 in the remaining of this document.

Figure 24 illustrates from both CMX and DUT aspect the associated iPerf mode and direction of data transmission. So, the server and the client represent the receiver and sender, respectively.

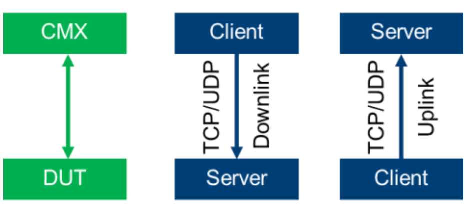


Figure 24 iPerf mode from both CMX and DUT aspect

In case the iPerf application does not exist on DUT, download and install it first.

Access: DAU home page > Tools Downloads > Select 'iPerf2' > Select 'iPerf2 Android apk' to start to download the installation file for Android devices

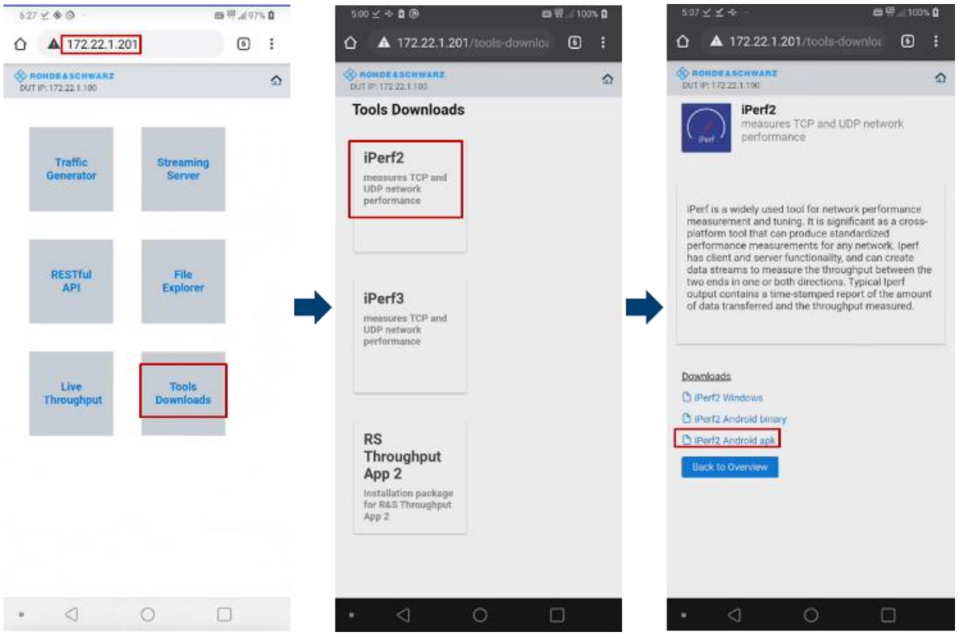


Figure 25 Install iPerf2 tool on DUT for Android OS

After the iPerf tool is downloaded and installed, launch the tool on DUT (see Figure 26).

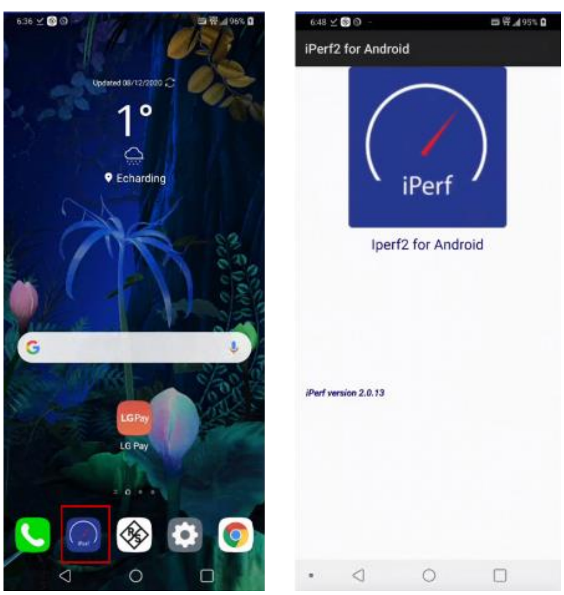


Figure 26 Launch iPerf tool on DUT

On CMX, iPerf tool is available together with the installation of the CSW software that is accessible from CMsquares. Parameter configurations of the iPerf tool are populated as configuration fields in CMsquares. The iPerf service square is generated automatically by Max Throughput Wizard (see 3.4) as part of the Max Throughput workspace.

To run iPerf tool, please bear followings in mind:

► The same transport protocol type (UDP or TCP) and port number should be configured on both server and client side.

► Always start iPerf service on the server side at first, then on the client side.

In Table 11, some frequently used iPerf command arguments are listed (for more details about iPerf, please refer to [12]). The last column in the table shows the corresponding fields in iPerf configuration square of the CMsquares.

iPerf Argument	Description	Comment	Configureation Field in CMsquares
-c	Client mode	Client only option	Mode
-s	Server mode	Server specific	Mode
-u	Use UDP	General option[9]	Protocol
-p	Server port for the server to listen on and the client to connect to. Default port is 5001	General option	Port
-i	Output interval of measurement report in second	General option	Interval
-t	Measurement time duration in second	General option	Statistic Count
-f	Display data formatting. b: bits / B: bytes / k: kilobits / K: kilobytes / m: megabits / M: megabytes / g: gigabits / G: gigabytes	General option	See footnote[10]
-b	Bandwidth (specified in UDP), e.g. -b 1M. It gives the maximum bit rate to be transferred for UDP	Client, UDP specific	Bandwidth
-l	Length of butter to read or write, default values are: 128 KB for TCP, 1470 Bytes for UDP over IPv4, 1450 Bytes for UDP over IPv6.	General option	Packet Size
-w	This argument sets the socket buffer sizes to the specified value. For TCP, this sets the TCP window size that is determined by bandwidth delay product (BDP). For UDP it is just the buffer which datagrams are received in, and so limits the largest receivable datagram size.	General option	Window Buffer Size
-M	Set TCP maximum segment size (MSS).	General option, TCP specific	Packet Size
-P	Number of parallel client threads to run	Client specific	Parallel Connections

Table 11 Frequently used iPerf arguments

In the following text, fundamentals of TCP MSS, TCP window size and parallel connection in conjunction with iPerf tool are explained. Usually, these parameters can be used to tune the throughput that under certain circumstance positively influence the data throughput.

TCP Maximum Segment Size (MSS)

It is a parameter that specifies the largest amount of data in bytes that a device can receive in a single TCP segment (exclude TCP header), It is decided during the three-way handshake of TCP connection establishment.

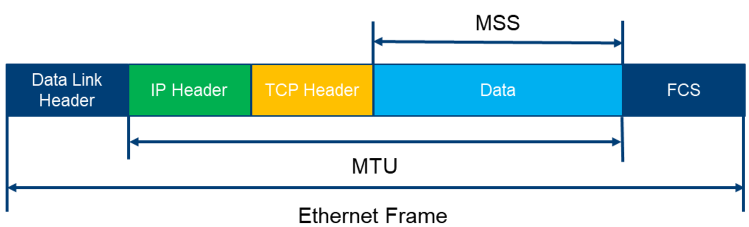


Figure 27 Ethernet Frame Structure

As illustrated in Figure 27, MSS is calculated as follows

$$MSS\ in\ bytes = MTU - IPv4\ or\ IPv6\ header\ length - TCP\ header\ length$$

Equation 2 MSS size calculation

[9] General option is valid for both client and server
[10] The unit is selectable in the drop-down menu of the associated parameter in CMsquares

Given that 1500 bytes for Ethernet Maximum Transmission Unit (MTU), 20/40 bytes for IPv4/IPv6 header and 20 bytes for TCP header (without TCP options), it results in maximum 1460 bytes and 1440 bytes MSS in Ethernet, for IPv4 and IPv6, respectively.

TCP MSS can be explicitly specified by using `–M` option of the iPerf command and '`Packet Size`' field in CMsquares.

TCP Window Size

In TCP world, to ensure the data sent by the sender does not overflow the receiver, TCP flow control is essential. The TCP window size or TCP receive window size in conjunction with receiver socket buffer size plays a vital role in the standardized flow control procedure.

The allocated receiver socket buffer is used to store received data (before the receiving application processes them) and the receiver sets the TCP receive window size (Advertised Window) in the TCP segment header to indicates to the sender the amount of free space remaining in its buffer. The sender is not allowed to send more than the Advertised Window number of bytes unless another ACK (with new Advertised Window) is received. The TCP receive window size should never exceed the allocated socket buffer size, since

$$TCP\ receive\ window\ size = Receiver\ socket\ buffer\ size - number\ of\ bytes\ waiting\ to\ be\ pulled\ by\ the\ receiving\ application$$

TCP receive window size has great impact on the data transmission rate of the sender and thus influences the data throughput. Too small window size causes frequent overrun of receiver buffer that urges the sender to pause which has negative effect on the data throughput. To optimize the data throughput, TCP receive window size should be set big enough so that the sender keeps the pipeline full, i.e. sending the data continuously. A throughput delay product or bandwidth delay product (BDP) is considered as a starting point to configure the minimum TCP receive window size to achieve optimal throughput. The calculation of minimum TCP receive window size is given in Equation 3 below.

$$TCP\ receive\ window\ size\ (Bytes) = Throughput\ (Kbps)\ /\ 8 \times RTT(ms) = BDP\ (Bytes)$$

Equation 3 Calculation of minimum TCP receive window size

The operating system configures the default TCP receive window size, e.g. 6 Mbyte for Linux OS. User specific TCP receive window size can be configured by using iPerf's `–w` option and '`Window Buffer Size`' field in CMsquares as the case may be.

Parallel Connections

Sometimes, due to the limit of the OS in terms of window size, the throughput is limited in a single iPerf connection. In that case, a solution to further increase the throughput is to use multiple parallel streams by using `–P` option of the iPerf command and '`Parallel Connections`' field in CMsquares.

3.1.2.2.3 iPerf3

Similar to iPerf2, iPerf3 is a newly implemented network tool for performance measurement and tuning. It has a smaller and simpler code base. Unfortunately, it isn't backwards compatible with iPerf2.

iPerf3 tool for Windows OS can be downloaded from DAU home page

Access: DAU home page > Tools Downloads > Select 'iPerf3' > Select 'iPerf3 Windows' to start download the installation file for Windows OS

3.1.2.2.4 R&S Throughput App2

R&S Throughput App2 (in short: Throughput App) is a fully automated throughput testing tool developed by R&S. It controls the throughput generator and sink from CMX, contains iPerf (TCP/UDP, IPv4/IPv6), FTP and latency measurement. The usage of Throughput App is always in conjunction with IP Tune (see 3.5).

It supports throughput measurements in both uplink and downlink directions even for several parallel streams and automatic retrieval of the logs from DUT. The tool features:

- FTP client
- HTTP client
- iPerf client and server
- Natperf same as iPerf but allows to traverse Network Address Translation (NAT), refer to 3.3
- Ping
- UniPing client and server (supports transport protocol UDP/TCP)
- Stun same as UniPing but allows to traverse NAT, refer to 3.3
- Runs on Android, Windows 10, Linux and iOS

In case the application does not exist on DUT, download and install it first.

Access: DAU home page > Tools Downloads > Select 'RS Throughput App 2' to start to download and install the app on DUT

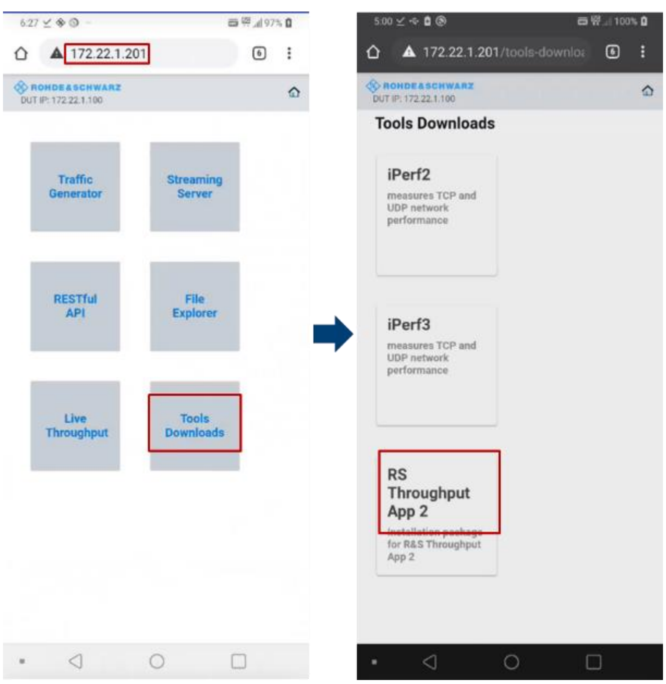


Figure 28 Install RS Throughput App 2 on DUT

Launch 'R&S Throughput App 2'

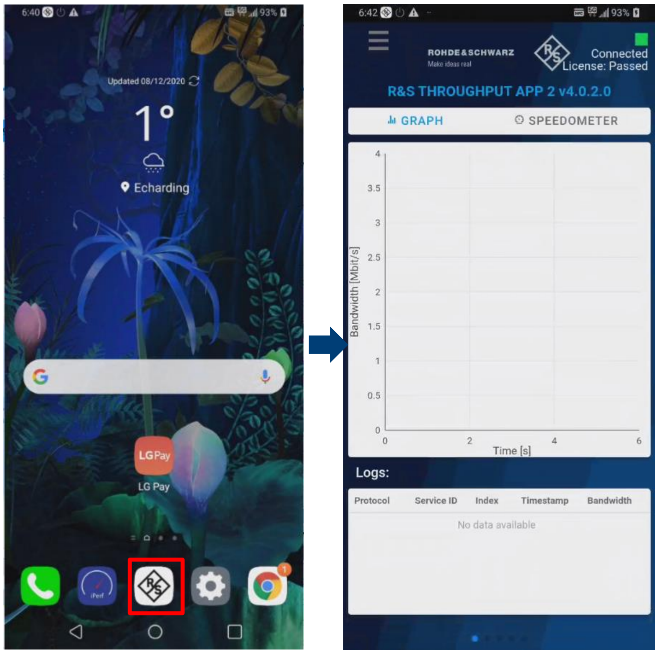


Figure 29 Launch RS Throughput App2

Please refer to [13] for more detailed information about the usage of 'R&S Throughput App 2'.

3.1.2.3 Automation Manager

Automation Manager is an optional external tool developed by R&S to control a DUT, e.g. power cycle the DUT, switch on/off airplane mode, place a call etc. It becomes mandatory if test automation is desired. In this case, DUT control should be set to 'Automatic' in CMsquares (see Figure 30). As a result, Automation Manager will undertake DUT control instead of manual interaction on DUT.

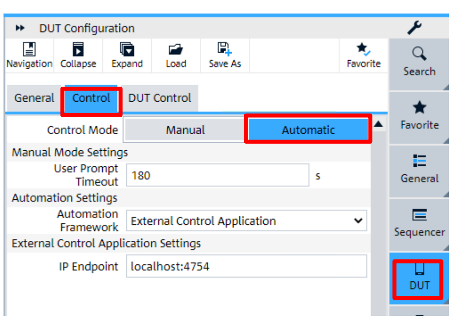


Figure 30 Automatic DUT control in CMsquares

After Automation Manager is separately installed (not included in the CSW installation) and launched on CMX, an icon appeared in the Windows status bar of the CMX (see Figure 31) indicates that it is up and running on the system.

Figure 31 Automation Manager running status in Windows status bar

In general, Automation Manager is a sort of interpreter. Its input is the text expression. Through a defined mapping between the text expression and associated action command in Automation Manager, the corresponding action command is forwarded to the connected DUT. The DUT executes then the action command in the end. The action command adopted here can be any acceptable commands implemented on the DUT, e.g. adb command for Android OS, AT command etc. In this document, we will focus on only AT commands as action commands.

In the following text, an example is shown on how to include the AT commands for turning on or off airplane mode in Automation Manager.

First of all, ensure the correct channel is added and configured that allows the communication between the DUT and Automation Manager. Normally, DUT is connected to the CMX via USB cable. The communication port is mapped to a virtual COM port in the Windows system that can be identified through device manager utility (Modems section) in Windows. An example shown in Figure 32 indicates a configuration of a channel with name 'UE' in automation manager where COM4 is used as a virtual COM port (Access in Automation Manager: 'Tools' > 'Channel Setup').

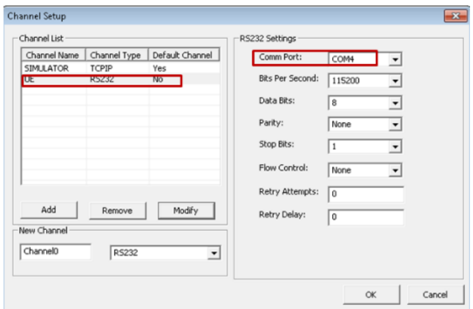


Figure 32 An example of channel setup in Automation Manager

'Forward Conversion' Tab in Automation Manager's main GUI takes care of the conversion between the text expression and associated action command. Figure 33 shows an example in 'Forward Conversion' tab that allows to turn on/off airplane mode on DUT via Automation Manager.

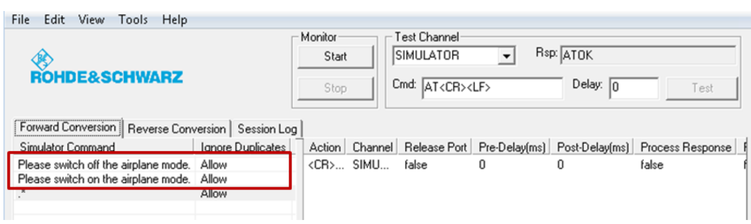


Figure 33 'Forward Conversion' tab in Automation Manager

Detailed definition of expressions and their associated action commands from Figure 33 are shown in Figure 34 below.

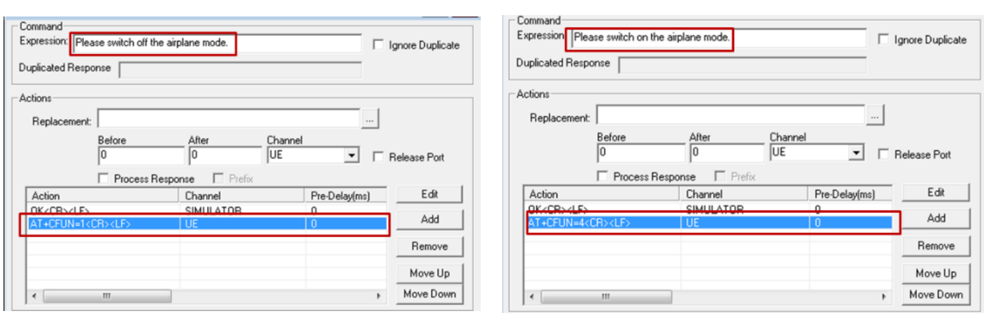


Figure 34 Definition of expressions and their action commands in Automation Manager (turn ON/OFF DUT using airplane mode by AT commands)

Table 12 summarizes mapping of the AT commands associated to the airplane mode and the accepted command expressions in Automation Manager. It is important to be noted here that the exact expressions given in the table should be entered in the Automation Manager as can be seen in Figure 34.

Expression	Action	Remark
Please switch off the airplane mode.	AT+CFUN=1<CR><LF>	Switch off the airplane mode on DUT
Please switch on the airplane mode.	AT+CFUN=4<CR><LF>	Switch on the airplane mode on DUT

Table 12 DUT controls in Automation Manager (airplane mode ON/OFF)

It is recommended to test the functioning of DUT control before the other measurement activities start. Enable automatic DUT control in CMsquares (see Figure 30) and then proceed with following steps (see Figure 35)

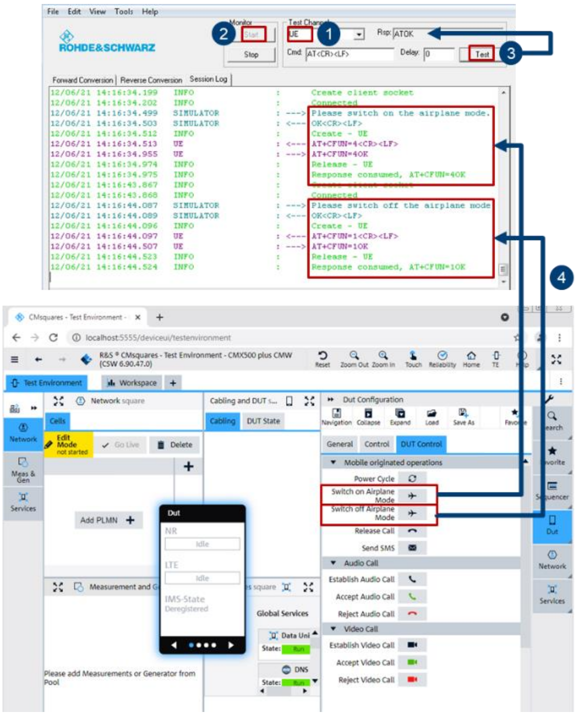


Figure 35 Start and verify DUT control function

1. Select the channel that is associated to the DUT (not channel 'SIMULATOR') in Automation Manager
2. Press 'Start'
3. Press 'Test' to test the connection. If the connection is successful, the string 'ATOK' should be returned in the field 'Rsp:'
4. Furthermore, in CMsquares, go to 'DUT Control' tab and press on 'Switch on Airplane Mode' and 'Switch off Airplane Mode', the corresponding AT command will be sent from Automation Manger to DUT and executed on DUT respectively. Verify the execution results presented in the 'Session Log' tab in Automation Manager to ensure the successful DUT control.

For more details about the Automation Manager, please refer to [14].

3.2 DUT Management

To route the signal from CMX to the right DUT antenna port, and vice versa, mapping information of DUT's supported band and its associated antenna port needs to be kept and the physical cabling between the DUT's antenna port and CMX's RF port has to be configured as a preliminary requirement on CMX before starting the network emulation. For doing that, DUT Management is the central place on CMX to maintain those settings.

In the subsequent text, most important operations in DUT management are described.

1. Pressing icon on the global toolbar enters DUT management area in CMsquares (see Figure 36).

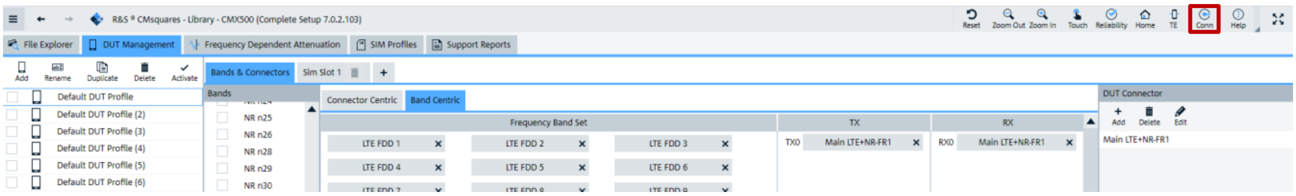


Figure 36 Access to DUT Management in CMsquares

2. If needed, create a new DUT by choosing 'Add' option within DUT Management (see Figure 37).

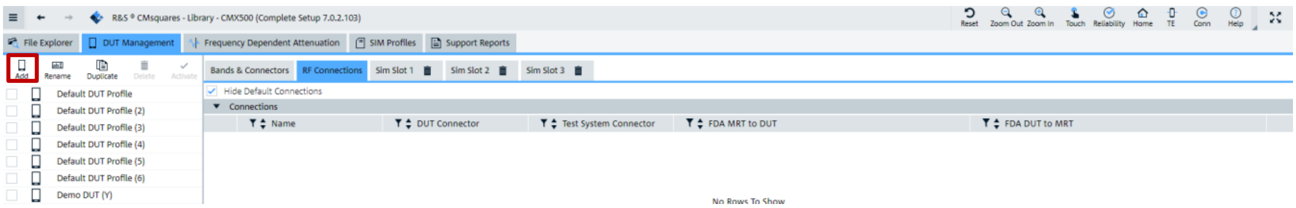


Figure 37 Add a new DUT in DUT Management

3. For the selected DUT, create an entry for each DUT antenna connector. An example in Figure 38 shows that 6 antenna connectors are entered for the selected DUT.

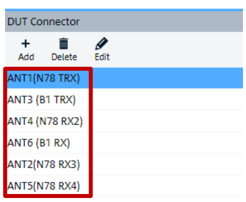


Figure 38 List of DUT antenna connectors

4. For each DUT connector, pressing on + sign in the center pane opens a dialogue window to specify the Tx and Rx direction of the connector. The supported band on that antenna connector can be then added by drag and drop the band id from the list in the left pane.

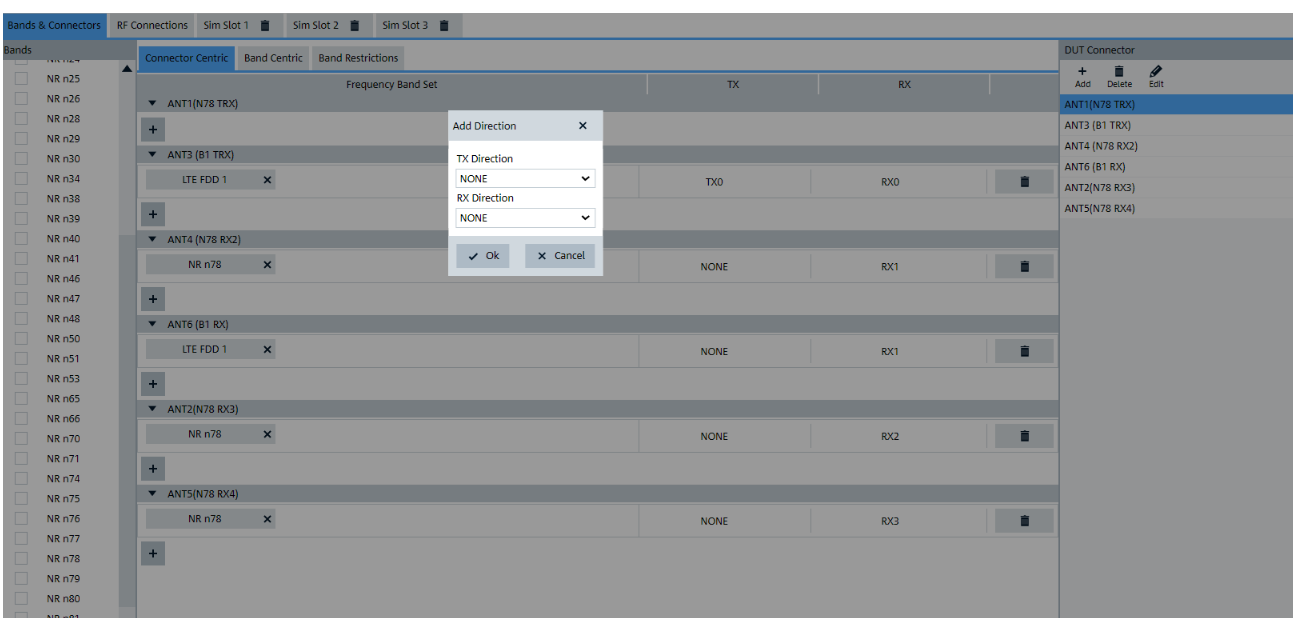


Figure 39 Configure the DUT connector

As an example, all configured DUT connectors in connector centric view is shown in Figure 40.

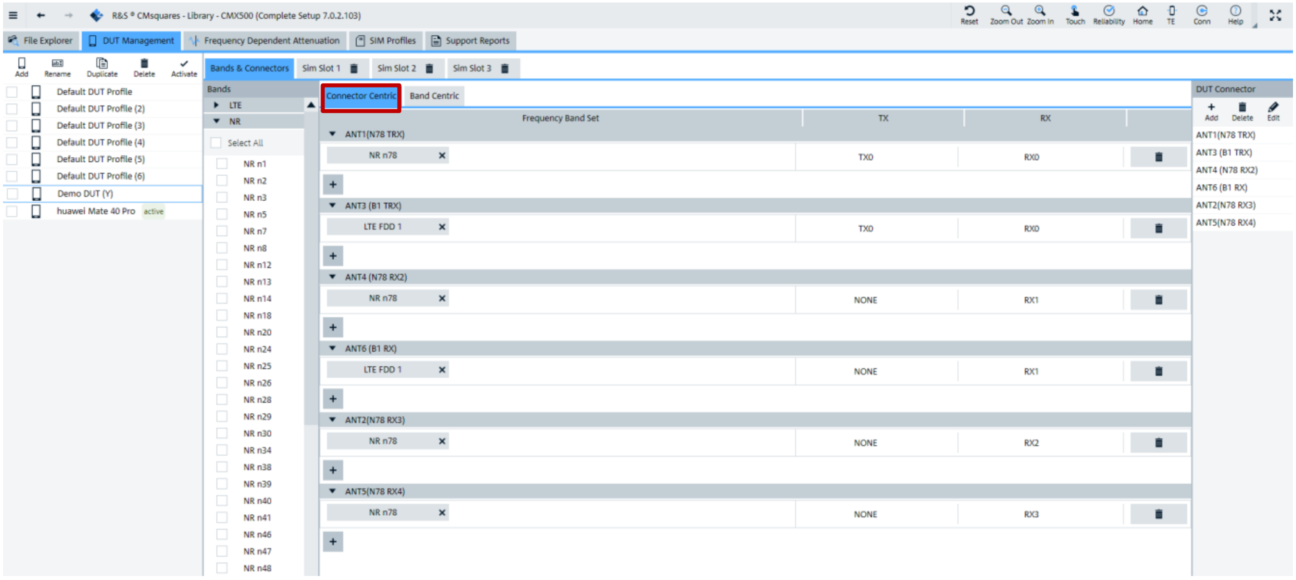


Figure 40 Overview of DUT connector configurations in 'Connector Centric' view

Alternatively, the same configuration is represented in band centric view as shown in Figure 41.

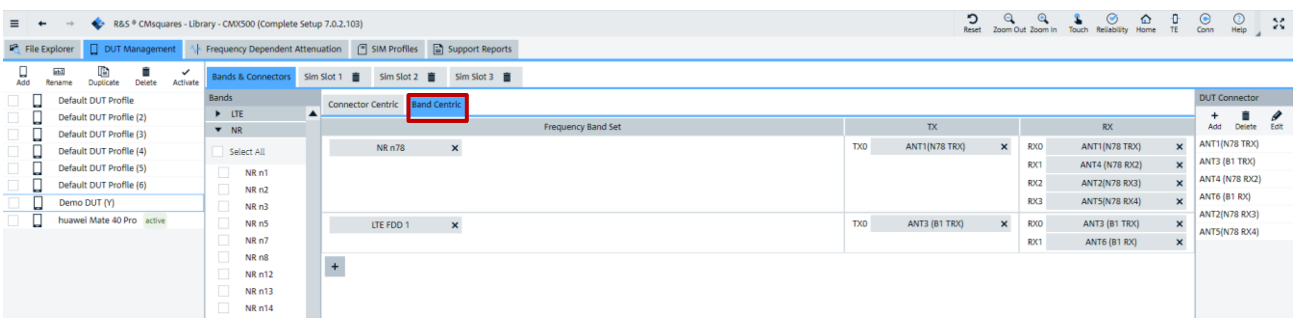


Figure 41 Overview of DUT connector configurations in 'Band Centric' view

5. The desired DUT has to be activated which then turns the DUT status to be labeled as 'active' (see Figure 42

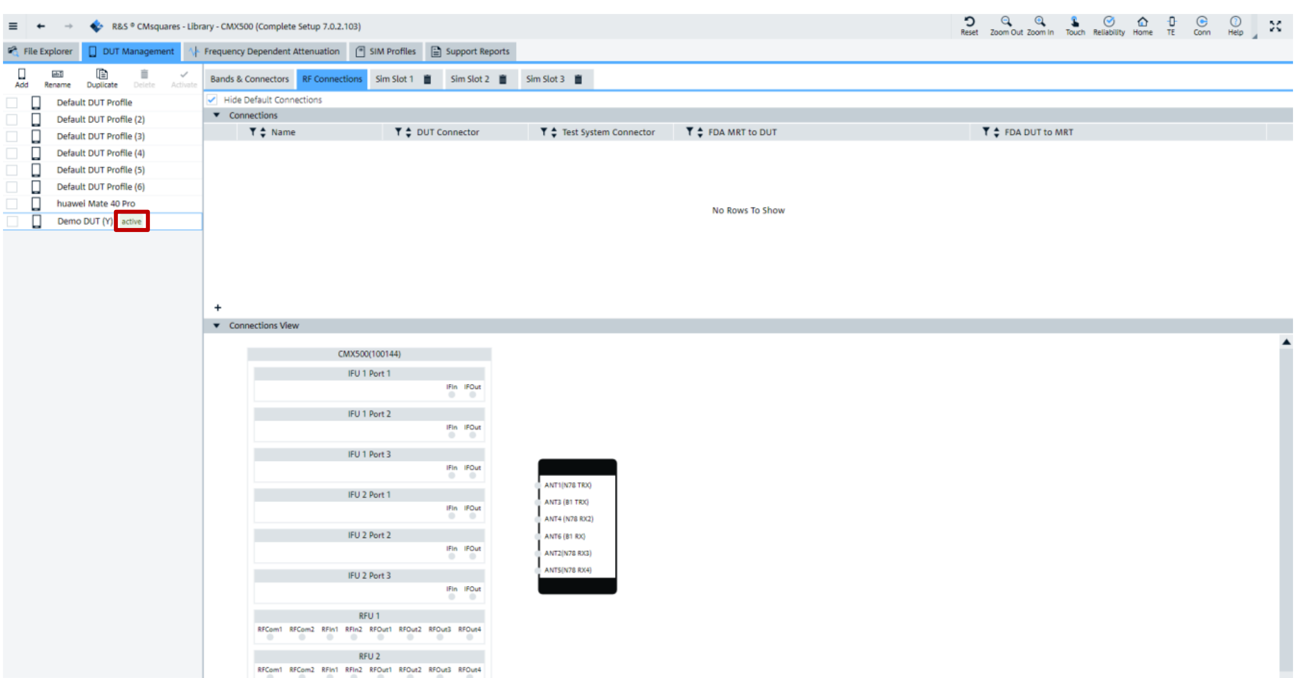


Figure 42 Indication of an active DUT

6. For a DUT with active status, RF connections can be defined which should reflect the physical cabling between the DUT connectors and RF connectors on CMX.

   In 'RF Connection' tab, press + sign to add a new RF connection, select a DUT connector and specify the corresponding physical RF connector of the CMX, give the FDA[11] for both directions (MRT to DUT and DUT to MRT if this is required).

   Figure 43 shows as an example all configured RF connections in the upper part and graphic representation of the RF connections in the low part of the pane.

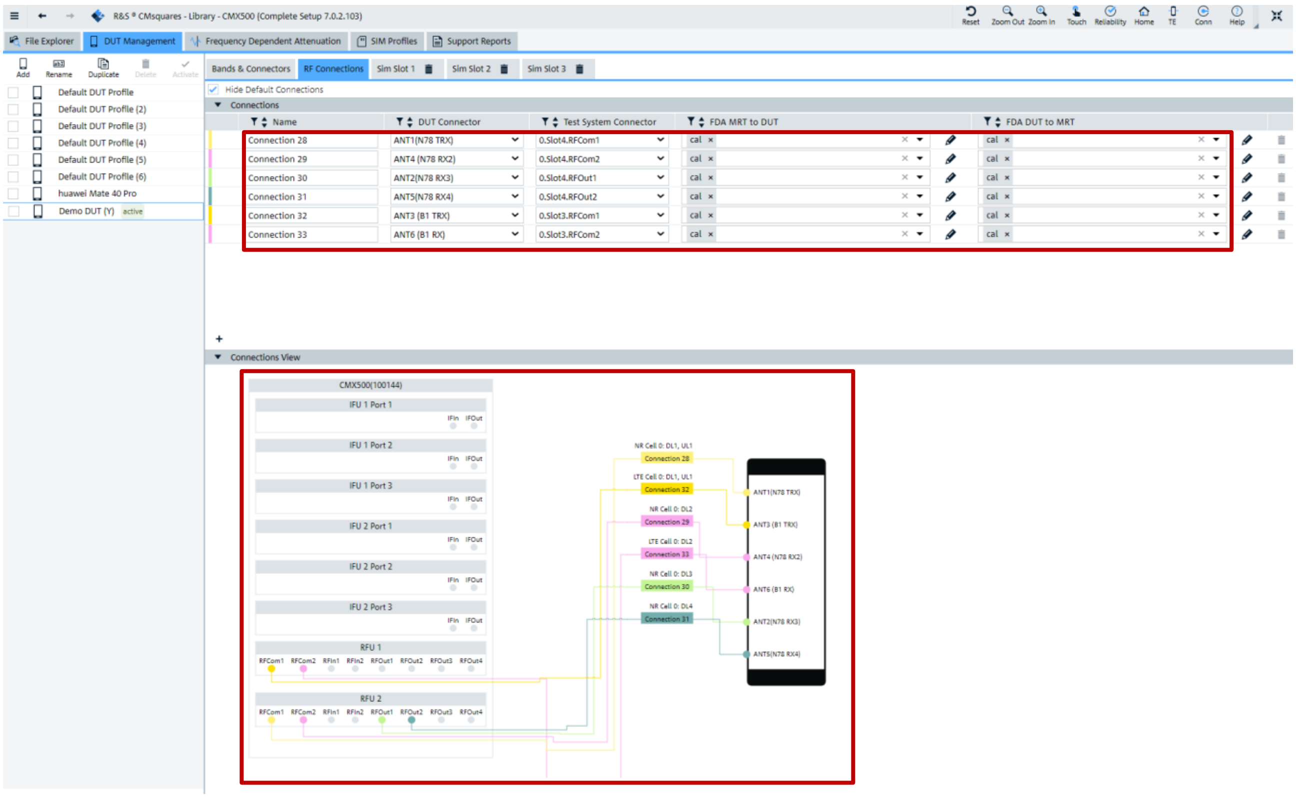


Figure 43 List of all RF connections

7. DUT and its RF connections to CMX are completely defined. It is ready to be tested from now on.

3.3 Supported Test Topologies

CMX supports both direct connection and Network Address Translation (NAT)-firewall traversal connection for throughput measurement as depicted in Figure 44 and Figure 45, respectively.

[11] FDA = Frequency Dependent Attenuation

Figure 44 Direct connection (support of Android OS)

Network Address Translation (NAT)-firewall traversal (Tethered DUTs)

Figure 45 NAT firewall traversal (Tethered DUTs)

In this document, throughput measurement in direction connection mode is described.

3.4 Max Throughput Wizard in CMsquares

Max Throughput Wizard is a use case provided in CMsquares that performs the optimization of the cell configurations which is necessary for UE's peak throughput testing. Simply saying, it provides the user an easy entry point of the cell configurations. In the remaining of this document, the term Wizard stands for Max Throughput Wizard.

The Wizard is accessible either directly from the home page of CMsquares (see Figure 46)

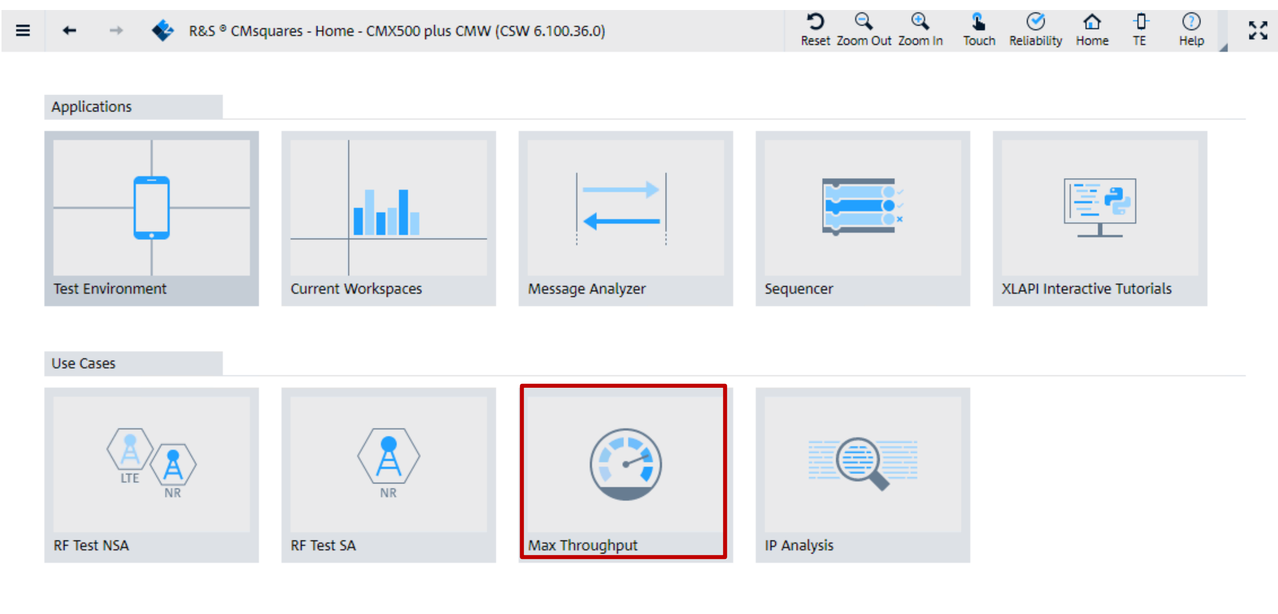


Figure 46 Launch the Wizard from CMsquares home

Or, alternatively, from the CMsquares' burger menu > 'Use Cases' > 'Max. Throughput' (see Figure 47)

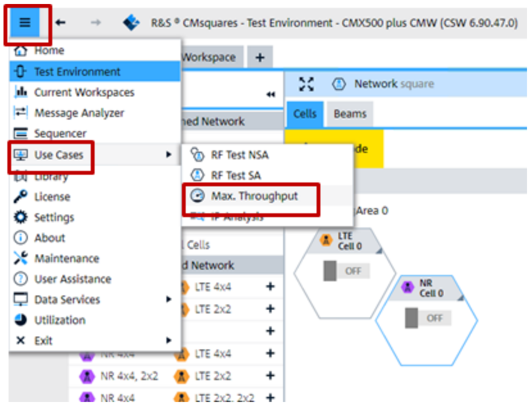


Figure 47 Launch the Wizard from CMsquares menu

For further readings, please refer to [13].

3.4.1 Steps in Wizard

Step 1 - Network Setup (see Figure 48)

It is highly recommended to use the DUT control of the CMsquares. Make sure that 'Use DUT Control' is checked, provided that Automation Manager is configured properly and up and running (see 3.1.2.3).

Select the frequency range, predefined network setup, e.g. LTE 4xN, NR 4xN. (see also Chapter 3.1.1 to know more about the network configuration versus hardware setup). Make sure that the selected network setup matches the actual hardware capability in your premise. Press 'Next' to proceed. It will take a little bit time for the system to start the network emulation.

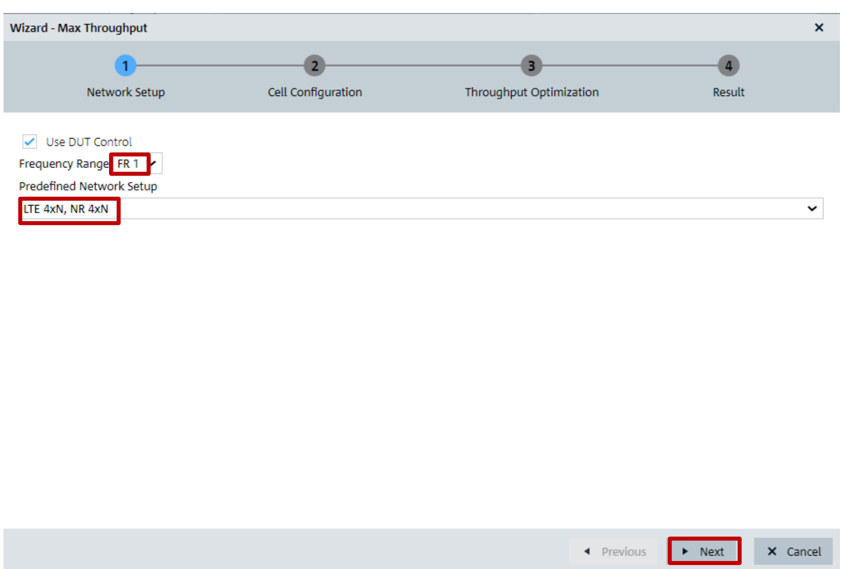


Figure 48 Wizard step 1 - Network Setup

Step 2 - Cell Configuration (see Figure 49)

In step 2 (cell configuration step), it is recommended to set 'Point A Location' as 'Mid', 'Low' or 'High' as shown in the upper part of Figure 49. The advantage of having these predefined frequency assignments is that CMX allocates SSB to the lower edge of the BWP as explained in Chapter 2.2.1. CMX leaves option 'User Defined' for point A position for advanced user as shown in the lower part of Figure 49. This provides extra flexibility to position the SSB in the resource grid. However, it requires the user to have deep knowledge of NR physical layer.

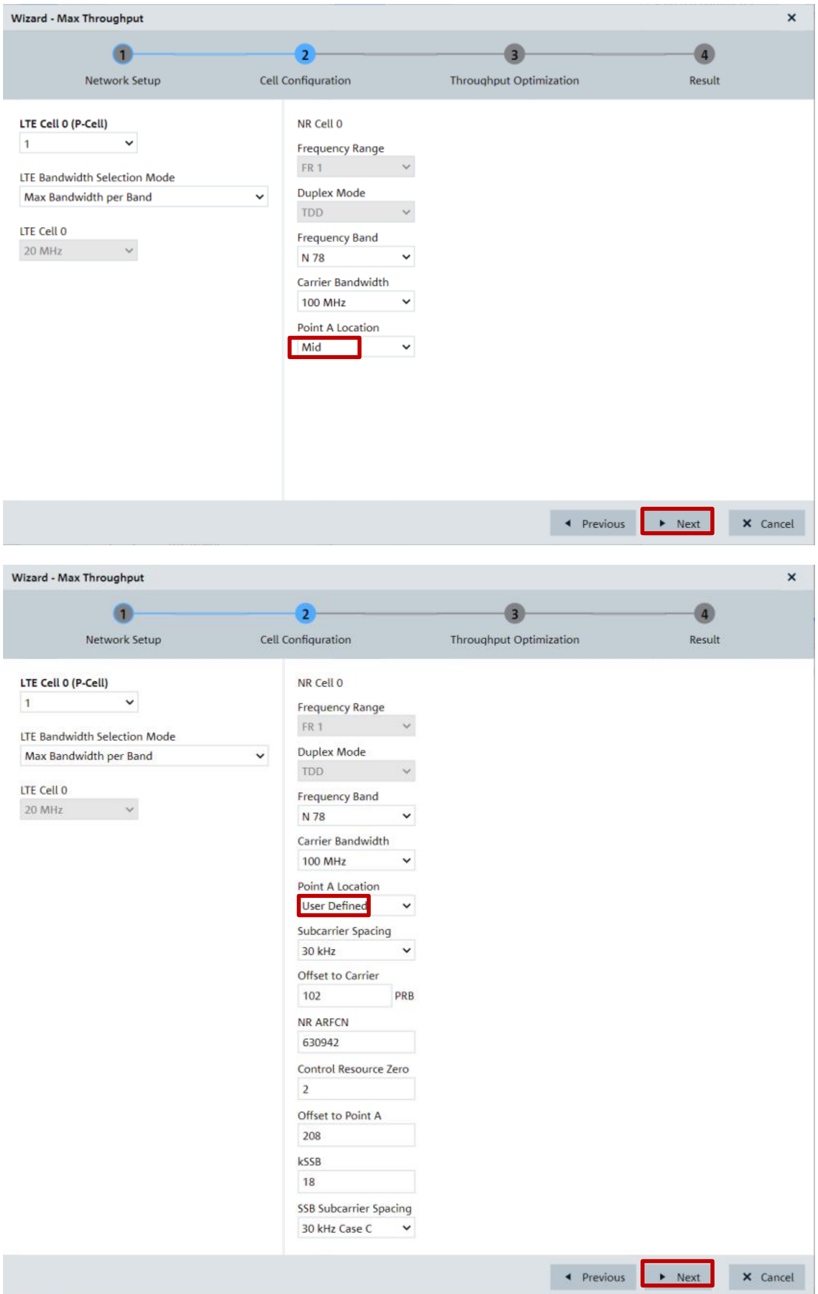


Figure 49 Wizard step 2 - Cell Configuration (upper: predefined Point A Location / lower: user defined Point A location)

After all the configurations are set in step 2, press 'Next' to proceed.

Step 3 - Throughput Optimization (see Figure 50)

In this step, the Wizard provides optimization measures and sets DL-MCS to boost DL throughput.

► DMRS optimization

With this option, DMRS Pos0 (no additional DMRS position) is configured in RRC message. Otherwise, default Pos2 is configured. If Pos1 is desired, then set it manually in the CMsquares under NR network configuration after exit from the Wizard.

► PDCCH optimization

It configures 1 symbol for PDCCH and rest of 13 symbols for PDSCH per DL slot (For more details, refer to Chapter 2.2.4).

- Mixed slot activation

  It enables partial slot 8 for PDSCH. This option depends on the UE capability. Only high performant UE supports this feature (see also Chapter 2.2.4).

- DL-MCS

  This can be individually configured depending on the actual maximum supported MCS indicated by the UE capability (see Chapter 2.2.6 for more information about MCS). Per default, it is set to max value 27.

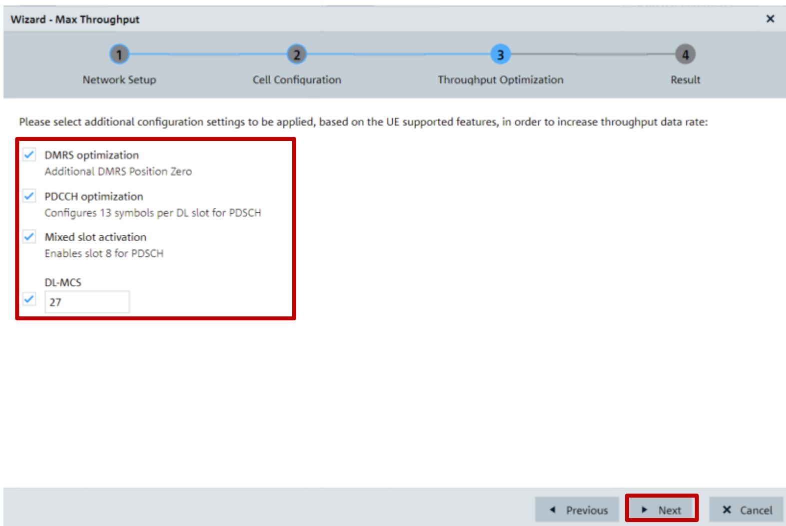


Figure 50 Wizard step 3 - Throughput Optimization

After step 3, press 'Next' to proceed.

As a result of settings in step 3, following actions are taken in CMX

- Cells are configured with optimized parameter settings. Afterward, the cells are turned on and enter idle state
- DUT is switched on (i.e. switch off the airplane mode) via Automation Manager
- DUT enters ENDC mode

Step 4 - Result (see Figure 51)

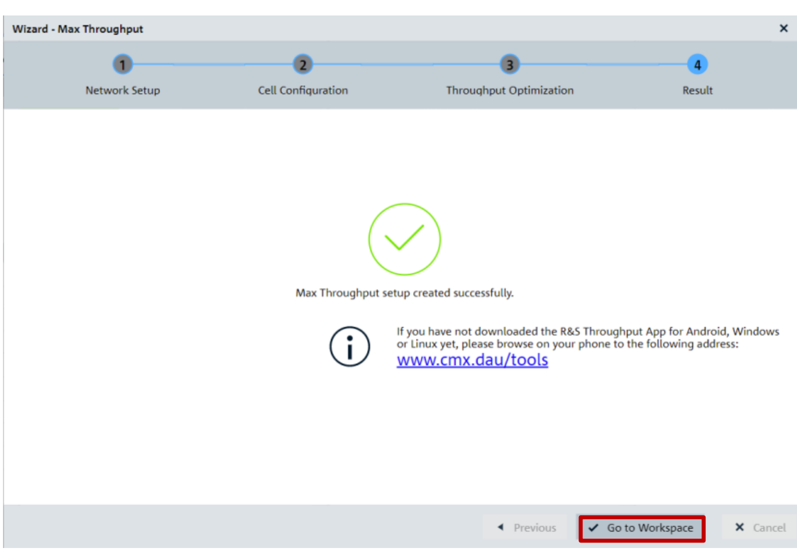


Figure 51 Wizard step 4 - Successful

Now, step 4 of the Wizard opens a result page as shown in Figure 51. Press 'Go to Workspace' will exit the Wizard and enter the created 'Max. Throughput' workspace in CMsquares (see details in 3.4.2).

After running through the Wizard, the NR cell parameters are set accordingly in CMsquares as shown in Figure 52. As mentioned in the beginning of this section, the Wizard provides an entry point of cell configurations optimized for max throughput testing. Based on that, it is always possible to adjust those parameters directly in CMsquares to meet individual test needs. For example, change DMRS Additional Position from Pos0 to Pos1.

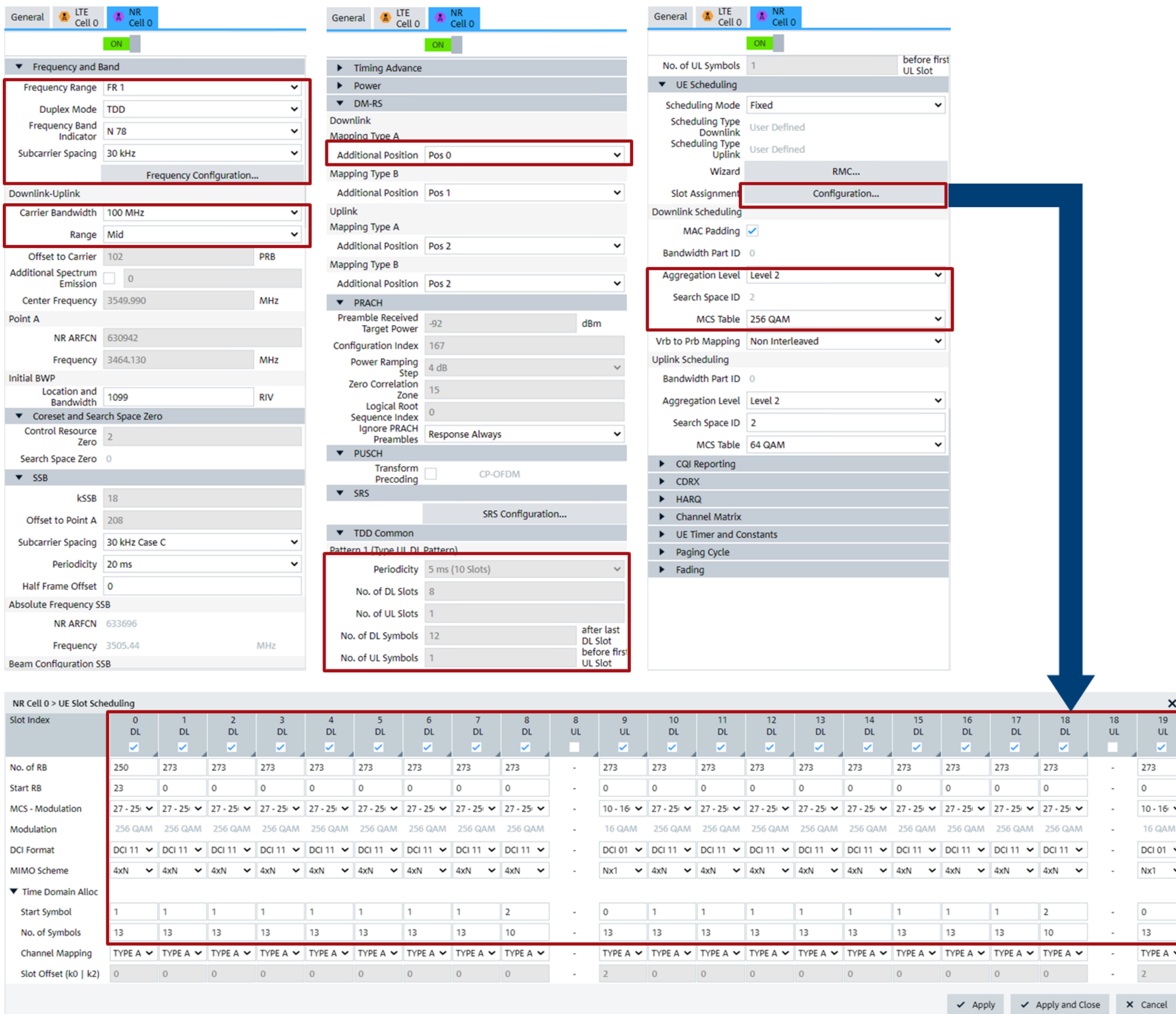


Figure 52 NR cell parameter settings in CMsquares after the execution of the Wizard

3.4.2 Measurements in Maximum Throughput Workspace

A maximum throughput workspace[12] (see Figure 53) is generated after the Wizard is went through. The workspace has the collection of following service squares

[12] All service squares can be accessed in CMsquares individually independent of the Wizard execution. For more details on how to access, please refer to [13] and [21].

- ► Block Error Rate BLER (see 3.4.2.1)
- ► Ping RTT (see 3.4.2.2)
- ► iPerf (see 3.4.2.3)
- ► Throughput (see 3.4.2.4)

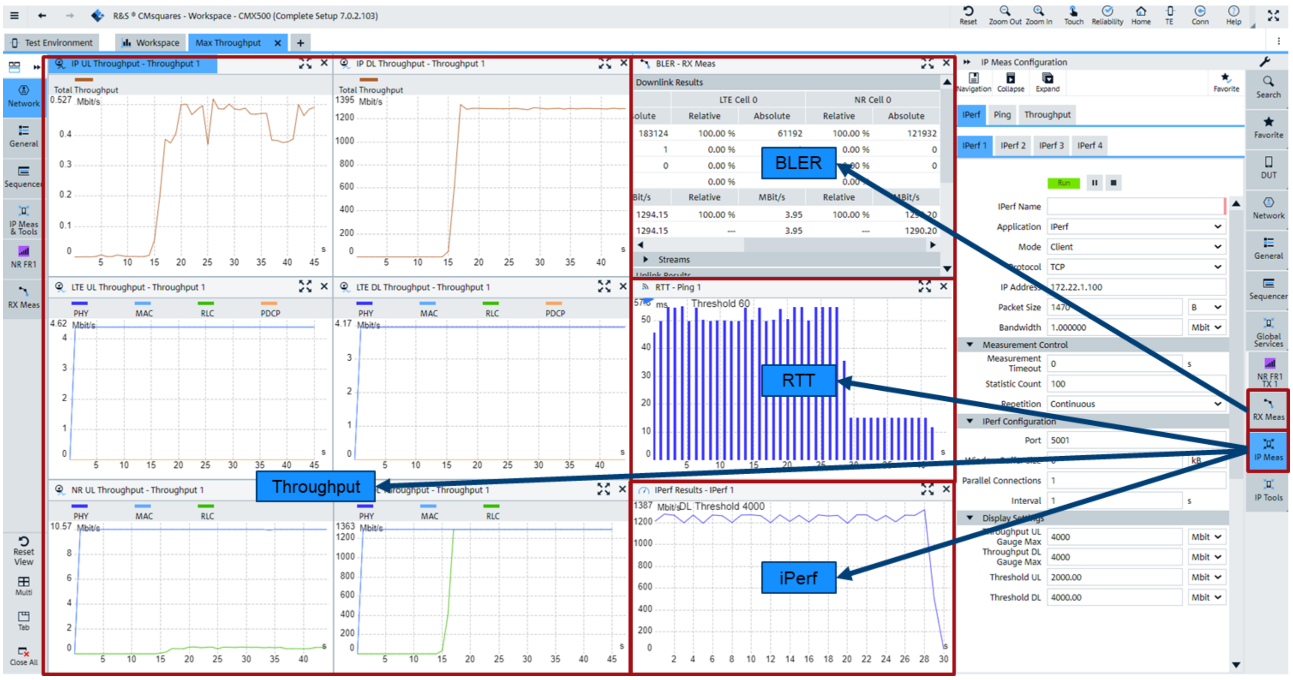


Figure 53 Overview of 'Max Throughput' workspace in CMsquares

As indicated in Figure 53, BLER measurement is accessible from 'RX Meas' service and the other measurements are accessible from 'IP Meas' service from the side-bar correspondingly.

3.4.2.1 Block Error Rate

Block Error Rate (BLER) is a quantitative measurement of quality in telecommunication. It is defined as ratio of the number of erroneous packets to the total number of transmitted packets on MAC layer. Different error ratio calculations are considered and implemented in CMsquares. Typical error rate calculation is based on the sum of non-acknowledged (NACK) and DTX packets over the entire number of transmitted packets.

It is essential to achieve good BLER result before further verification of the maximum throughput on the higher layers. Typical target BLER should be below 10%.

Select 'RX Meas' service from side-bar, select 'Continuous' mode and press ► key to start the BLER measurement (see Figure 54).

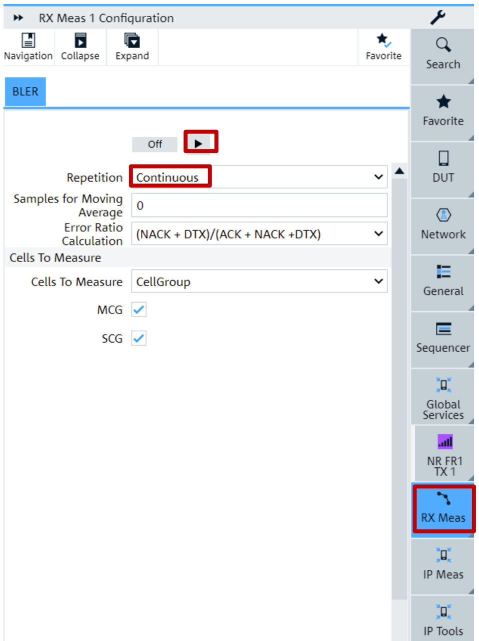


Figure 54 Start BLER measurement in CMsquares

In 'BLER-RX Meas' square, the scheduled DL data rate and actual measured DL BLER and data rate over different radio access technology (RAT) are presented. Example in Figure 55 shows the scheduled DL data rate has 1295.11 Mb/s over NR cell and DUT achieves 0% DL BLER. In addition to DL, UL BLER measurement, UL data throughput and scheduled UL throughput are presented there as well.

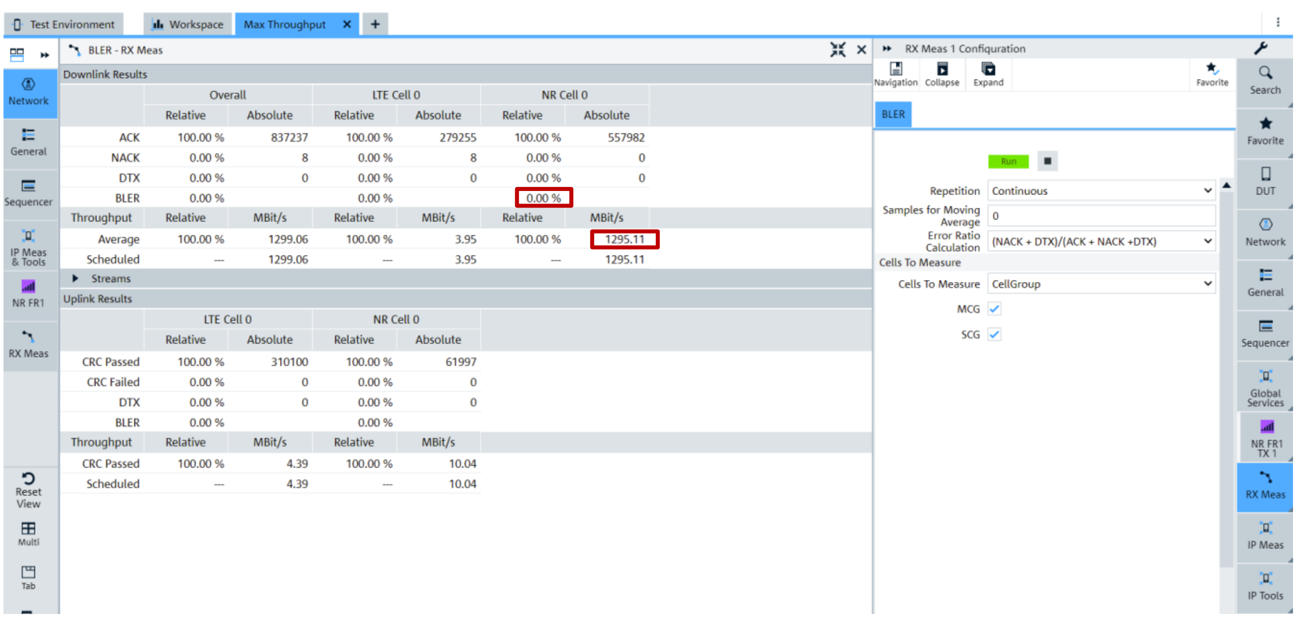


Figure 55 BLER measurement

3.4.2.2 Round-trip Time

Round-trip time (RTT) is another fundamental performance metric of a UE which represents the time length that it takes for a data packet to travel from the sender to receiver plus the time for the acknowledgement in the reverse direction. It is measured by PING test and gives an indication of speed and reliability of the data connection.

Select 'IP Meas' service from the side-bar and choose 'Ping' measurement in CMsquares, specify the destination IP address (default: 172.22.1.100) and press ▶ key to start the PING measurement.

The result example shown in Figure 56 indicates the average 14.92 ms RTT of the connection.

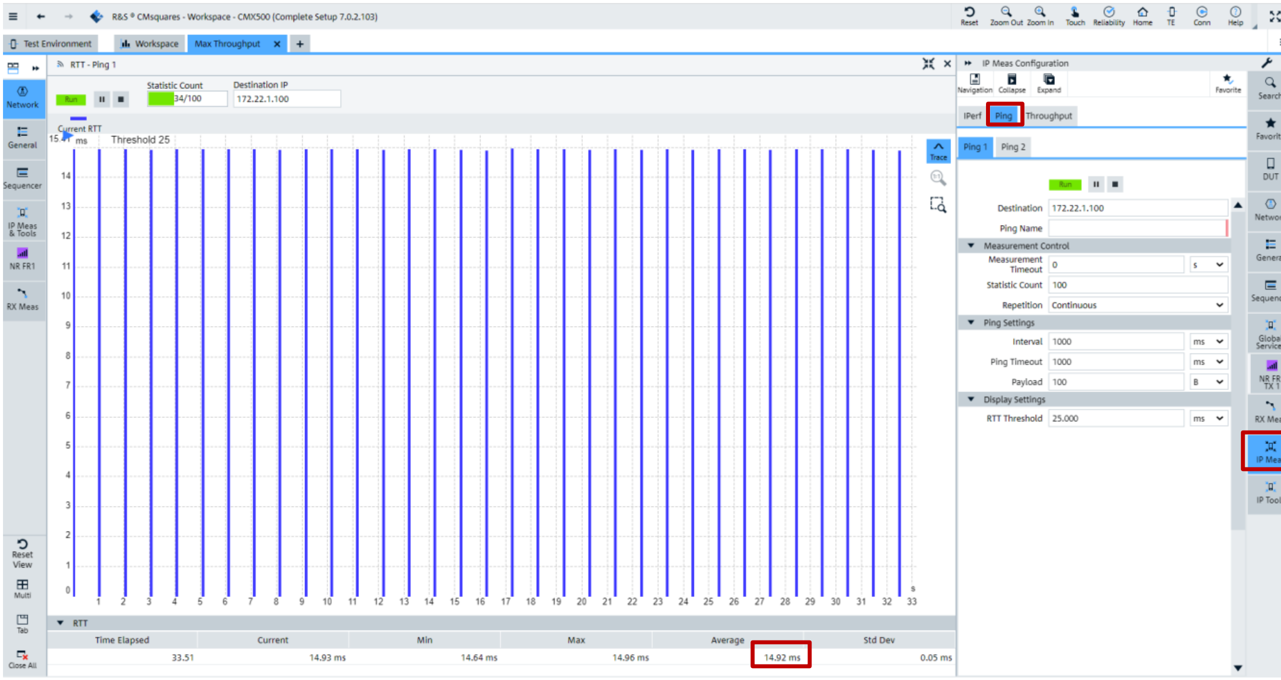


Figure 56 Round Trip Time (RTT) measurement

3.4.2.3 iPerf

In this section, four different test scenarios using iPerf tool to verify the throughput are described. An overview of the tests is given below in Table 13.

Scenario	Section	iPerf Mode[13]		Direction	Transport Protocol
		DUT	CMX		
1	3.4.2.3.1	Server	Client	Downlink	TCP
2	3.4.2.3.2	Server	Client	Downlink	UDP
3	3.4.2.3.3	Client	Server	Uplink	TCP
4	3.4.2.3.4	Client	Server	Uplink	UDP

Table 13 Overview of iPerf test scenarios

For basic information about iPerf tool, please refer to Chapter 3.1.2.2 or [12].

The main focus of this document is to show how to verify the maximum downlink throughput. Therefore, the optimizations are only done in downlink direction. But, for sake of information completeness, this section also covers iPerf measurements in uplink direction (3.4.2.3.3 and 3.4.2.3.4). In contrast to downlink, the optimizations in uplink are not performed. That means, the uplink throughputs shown in this document do not reflect DUT's maximum capacity.

In this section, the iPerf commands of the examples are issued based on following considerations:

► Without otherwise specified, default port `5001` is used on both client and server side. Therefore, iPerf `-p` option is omitted in the iPerf command and default port `5001` is kept in CMsquares, too.

[13] Server = Receiver, Client = Sender

► Without otherwise configured, default DUT IP address `172.22.1.100` is used.

► Parallel iPerf option `-P` is not in use. Because single iPerf connection is already enough to measure the maximum throughput of the used DUT.

► Default TCP window size is in use (detailed explanation, please refer to 3.4.2.3.1 and 3.4.2.3.3). That means, iPerf `-w` option in the iPerf command is omitted or 'Window Buffer Size' field of iPerf service square in CMsquares is set to `0` (default setting), whenever it applies.

3.4.2.3.1 Scenario 1 (Downlink, TCP)

In general, TCP receive window size is an important parameter that influence the TCP data throughput. A proper window size has to be configured at the receiver side in order to achieve optimal data throughput (see also 3.1.2.2.2).

Example shown in Figure 55 and Figure 56 deliver scheduled DL throughput 1295 Mb/s and RTT 14.92 ms, respectively. To achieve optimal throughput, a minimum TCP window size (BDP) of 2.415 Mbyte is required according to Equation 3. Since iPerf tool applies already 6 MB as default TCP window size which is larger than our calculated value here, there is no need to specify explicitly the TCP window size in iPerf command, i.e. iPerf `-w` option is omitted here.

iPerf configurations of our example are shown in Table 14.

	DUT	CMX
Mode	Server (Receiver)	Client (Sender)
Protocol	TCP	TCP
TCP window size	6 MB[15]	
iPerf command	iPerf -s -i 1 -t 60	

Table 14 Example iPerf configuration of TCP for downlink

To perform the test, please follow the procedure below:

1. On DUT, enter iPerf command '`-s -i 1 -t 60`' in the input field of iPerf application and press 'start' key to receive TCP packets (see Figure 57).

[15] iPerf default TCP window size on DUT

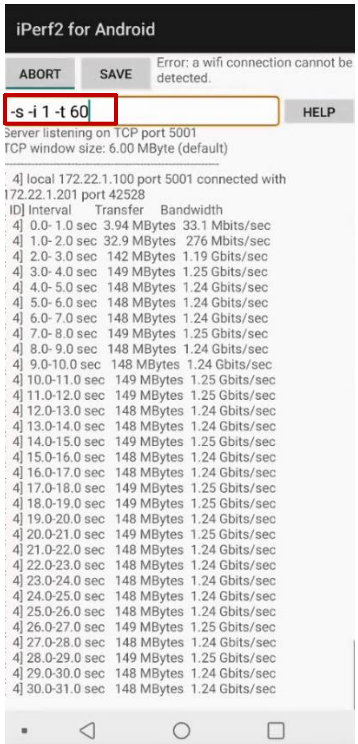


Figure 57 iPerf command on DUT (Server, TCP) and measurement result

2. Configure iPerf in CMsquares and press ▶ key to start the measurement (see Figure 58).

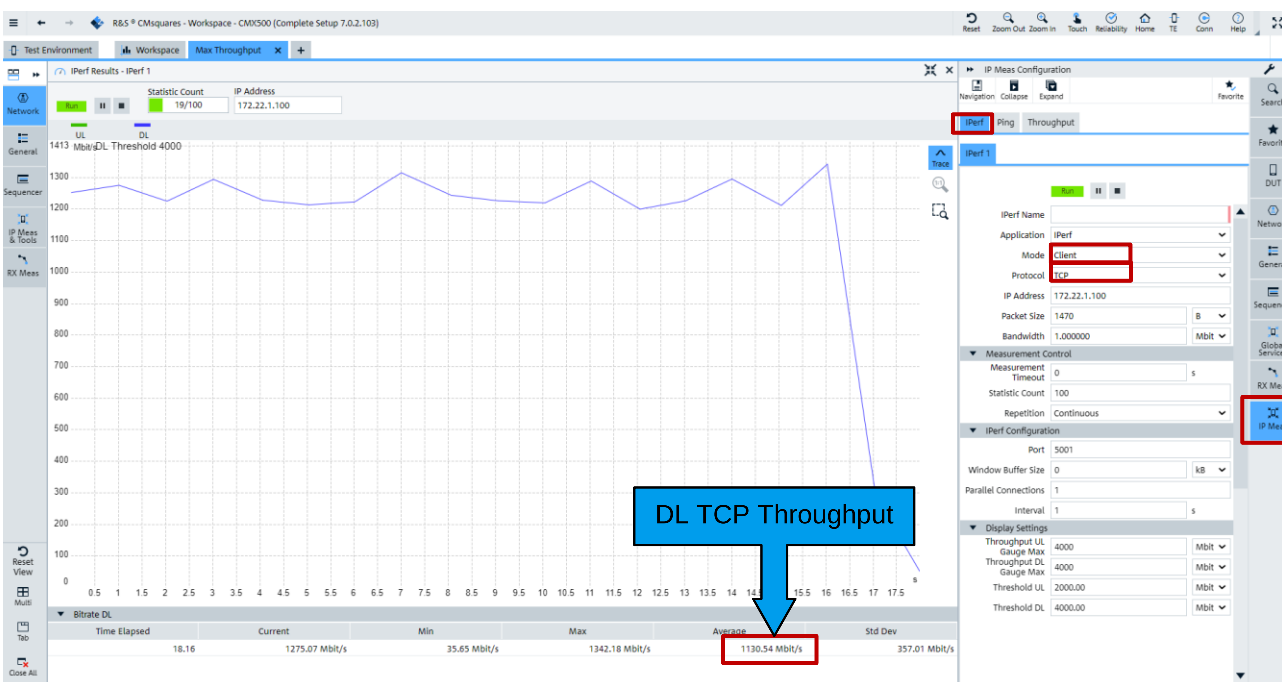


Figure 58 iPerf settings in CMsquares (Client, TCP) and measurement result

3. Read the measurement result in CMsquares (see Figure 58) or on DUT (see Figure 57), i.e. average DL TCP throughput in CMsquares shows 1130.54 Mbps

3.4.2.3.2 Scenario 2 (Downlink, UDP)

For throughput testing using UDP, unlike TCP case shown in 3.4.2.3.1, client sends UDP data constantly at the rate given by the iPerf command line bandwidth option '`-b`'. There is no acknowledgement from server. The data rate observed on the client side is constant even if server is not running. So, always check actual UDP throughput at the server side, not on the client side.

Example shown in Figure 55 gives the scheduled DL throughput 1295 Mb/s. Based on that, bandwidth parameter on the client side needs be set to a value that is no lower than the scheduled DL throughput, e.g. 1300 Mb/s, to ensure that the test condition for maximum throughput of UDP DL is met. On the server side, big enough buffer size needs also to be specified through iPerf command line option '`–w`'. Default UDP buffer size used by iPerf tool on DUT is 256 kB which might be too small at high data rate. 4 MB buffer size as an experience value is recommended to be used here.

iPerf configurations of our example are shown in Table 15.

	DUT	CMX
Mode	Server (Receiver)	Client (Sender)
Protocol	UDP	UDP
Bandwidth		1300 Mbit/s
Buffer size	4 MB[16]	
iPerf command	iPerf -s -u -w 4M -i 1 -t 60	

Table 15 Example iPerf configuration of UDP for downlink

To perform the test, please follow the procedure below:

1. On DUT, enter the iPerf command '`-s –u –w 4M –i 1 –t 60`' in the input field of iPerf application and press 'start' key to receive UDP packets (see Figure 59).

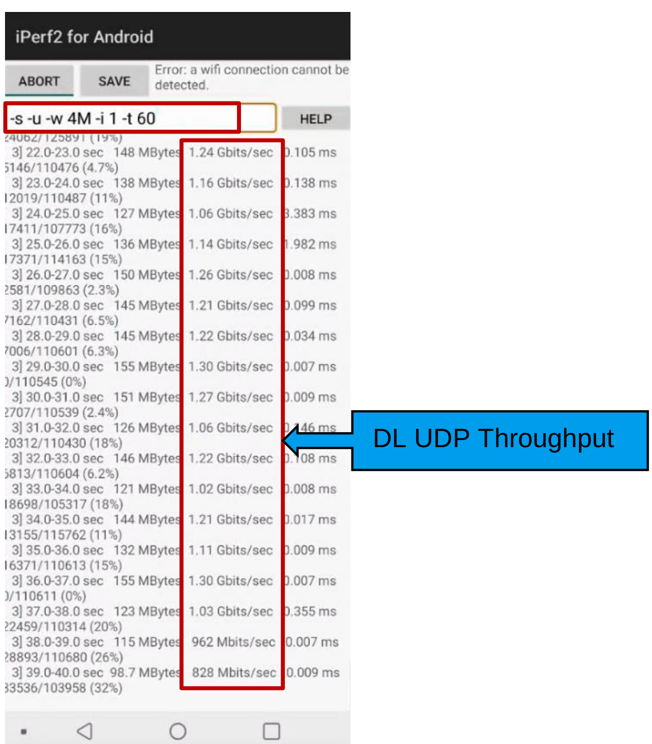


Figure 59 iPerf command on DUT (Server, UDP) and measurement result

---

[16] Experience value

2. Configure iPerf in CMsquares, make sure the bandwidth is set no lower than the scheduled rate (see Rx measurement in Figure 55 to obtain the scheduled rate) and press ▶ key to start to send UDP packets at the configured data rate (see Figure 60).

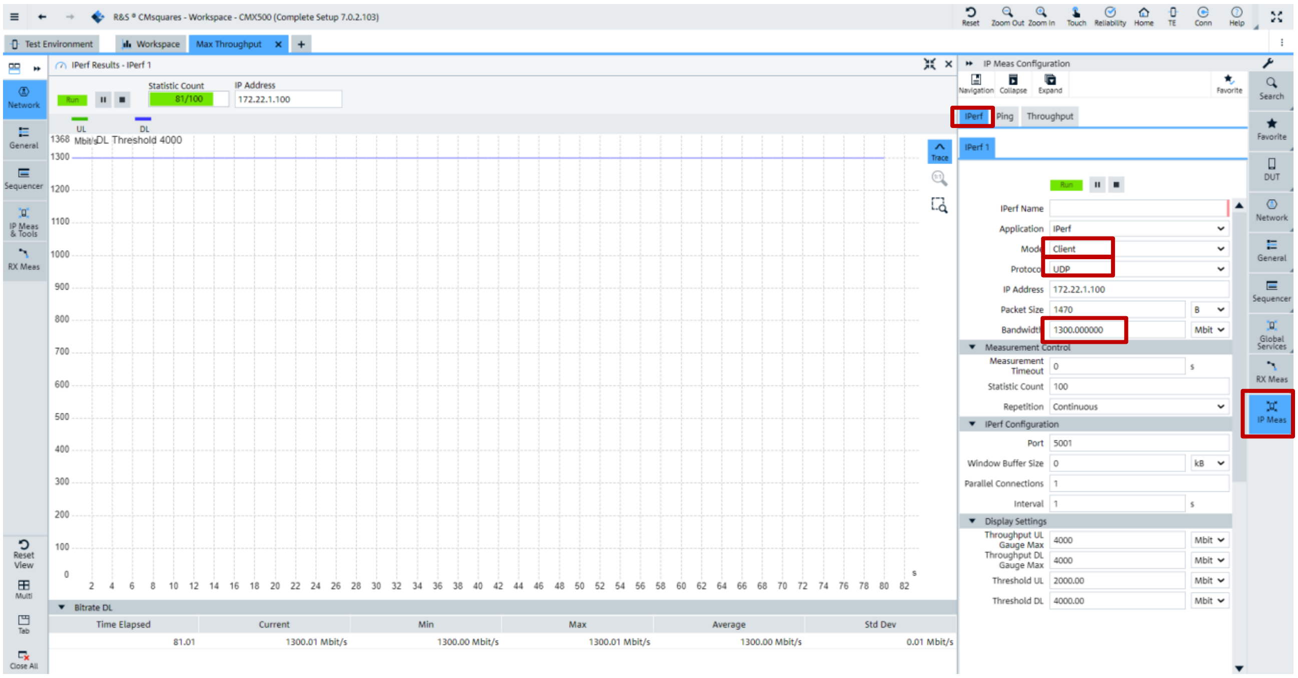


Figure 60 iPerf settings in CMsquares (Client, UDP)

3. Read the measurement result on DUT, i.e. on receiver (see Figure 59). In our example here, UDP DL throughput measured in iPerf tool is about 1.2 Gb/s.

3.4.2.3.3 Scenario 3 (Uplink, TCP)

Same as for the DL case explained in Chapter 3.4.2.3.1, to measure throughput in an optimal way, minimum TCP window size based on BDP has to be calculated (see 3.1.2.2.2).

In our example here, scheduled UL throughput is indicated in Rx measurement in CMsquares (shown in Figure 55). It is expected to achieve 10.04 Mb/s throughput. RTT is about 14.92 ms (see Figure 56). Therefore, minimum TCP window size is 18.7 kB according to Equation 3. The default window size used by iPerf is 6 MB which is larger than our calculated window size here, therefore, the default value will still be applied. In CMsquares, the value '0' in window buffer size field in iPerf service means the default value is used.

iPerf configurations of our example are shown in Table 16.

	DUT	CMX
Mode	Client (Sender)	Server (Receiver)
Protocol	TCP	TCP
Window buffer size		0
iPerf command	iPerf -c 172.22.1.201[17] -i 1 -t 60	

Table 16 Example iPerf configuration of TCP for uplink

To perform the test, please follow the procedure below:

1. Configure iPerf in CMsquares and press ▶ key to start to receive TCP packets (see Figure 61).

[17] 172.22.1.201 is the default IP address of CMX DAU

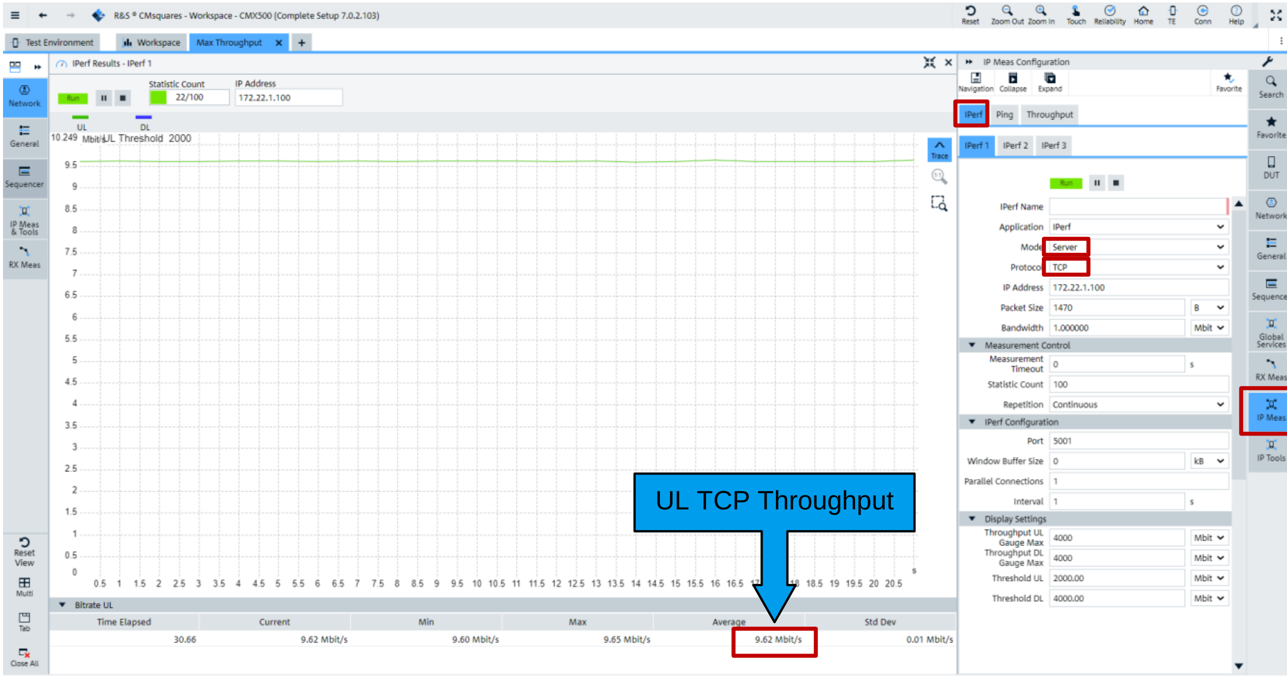


Figure 61 iPerf settings in CMsquares (Server, TCP) and measurement result

2. On DUT, enter the iPerf command '`-c 172.22.1.201 -i 1 -t 60`' in the input field of iPerf application and press 'start' key to send TCP packets. (see Figure 62).

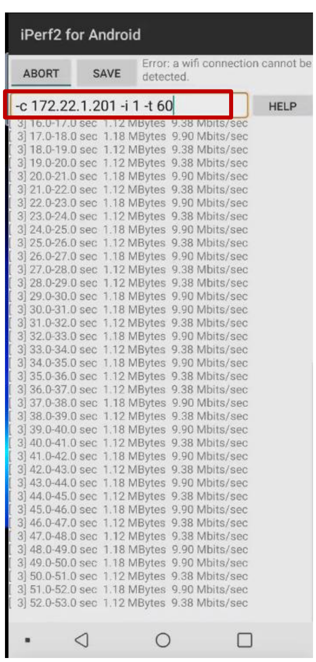


Figure 62 iPerf command on DUT (Client, TCP)

3. Read the measurement result in CMsquares (see Figure 61). i.e. average UL TCP throughput is 9.62 Mb/s

3.4.2.3.4 Scenario 4 (Uplink, UDP)

Same as for the DL case explained in Chapter 3.4.2.3.2, for UDP UL throughput testing, client (DUT) sends UDP data to server (CMX) at constant rate. On the client side, bandwidth (use `-b` option in iPerf command) larger than the expected throughput needs to be specified. On the server side, big enough buffer size has to be specified if the default one is not sufficient.

For example, from the UL Rx measurement shown in Figure 55, we already know that the target UL throughput is about 10.04 Mb/s. For UDP UL, we can refer to this value and assign a larger value, e.g. 15 Mb/s, to iPerf bandwidth `-b` option. Therefore, iPerf command on client side turns to be '`iPerf -c 172.22.1.201 -u -b 15M -i 1 -t 60`'.

iPerf configurations of our example are shown in Table 17.

	DUT	CMX
Mode	Client	Server
Protocol	UDP	UDP
Window Buffer Size		0[18]
iPerf command	iPerf -c 172.22.1.201[17] -u -b 15M -i 1 -t 60	

Table 17 Example iPerf configuration of UDP for uplink

To perform the test, please follow the procedure below:

1. Configure iPerf in CMsquares and press ▶ key to start to receive UDP packets (see Figure 63).

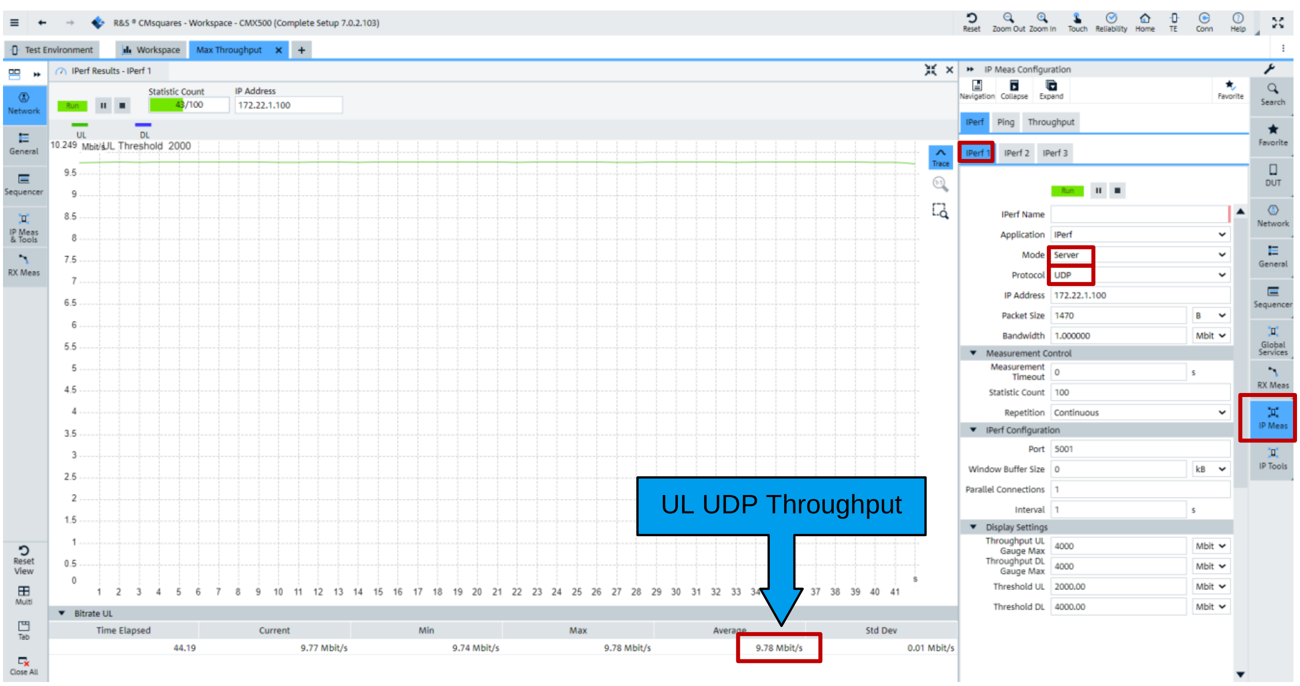


Figure 63 iPerf settings in CMsquares (Server, UDP) and measurement result

2. On DUT, enter the iPerf command '`-c 172.22.1.201 -u -b 15M -i 1 -t 60`' in the input field of iPerf application and press 'start' key to send UDP packets (see Figure 64).

[18] Value '0' means the default UDP buffer size is used. On CMX, it is 8192 kB.

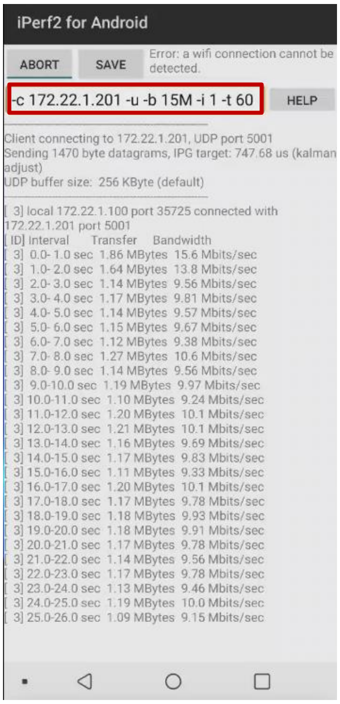


Figure 64 iPerf command on DUT (Client, UDP)

3. Read the measurement result in CMsquares (see Figure 63), i.e. average UDP UL throughput is 9.78 Mb/s

3.4.2.4 Throughput

'Throughput' square in CMsquares examines the throughput in both UL and DL direction in each layer (from lower PHY layer up to the upper protocol stack layer) of each RAT, as well as the overall throughput on IP layer. Figure 65 depicts a diagram of the throughput measurements in CMsquares[19]. It offers an intuitive way to identify the data throughput bottleneck throughout the whole communication layers.

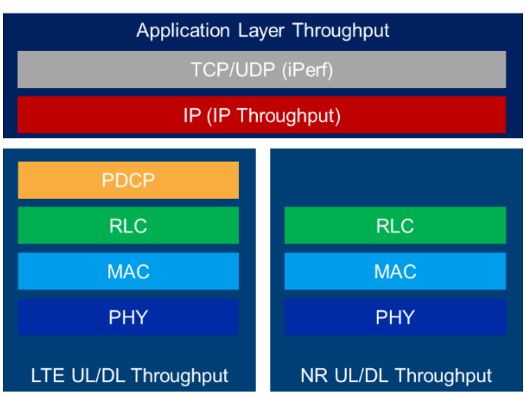


Figure 65 Throughput measurements in CMsquares

To perform throughput test, some preconditions have to be fulfilled. MAC padding option (see Figure 66) is enabled as per default. This is essential for throughput testing on PHY and MAC layer in both LTE and NR.

[19] iPerf throughput measurement belongs to a separate service in CMsquares (refer to Chapter 3.4.2.3)

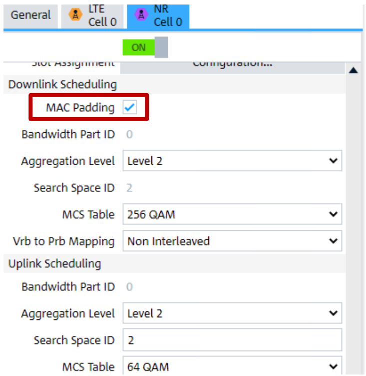


Figure 66 Enable MAC Padding in NR

To verify the throughput beyond RLC layer, additional IP data traffic needs to be generated, e.g. through traffic generator, or through iPerf session.

For ENDC mode, throughput above RLC layer can be observed on either LTE or NR, or on both depending on DRB establishment mode (either MCG, SCG or Split mode). See Chapter 2.2.9.3 for more detailed information about DRB establishment.

Figure 67 presents the throughput measurements in CMsquares.

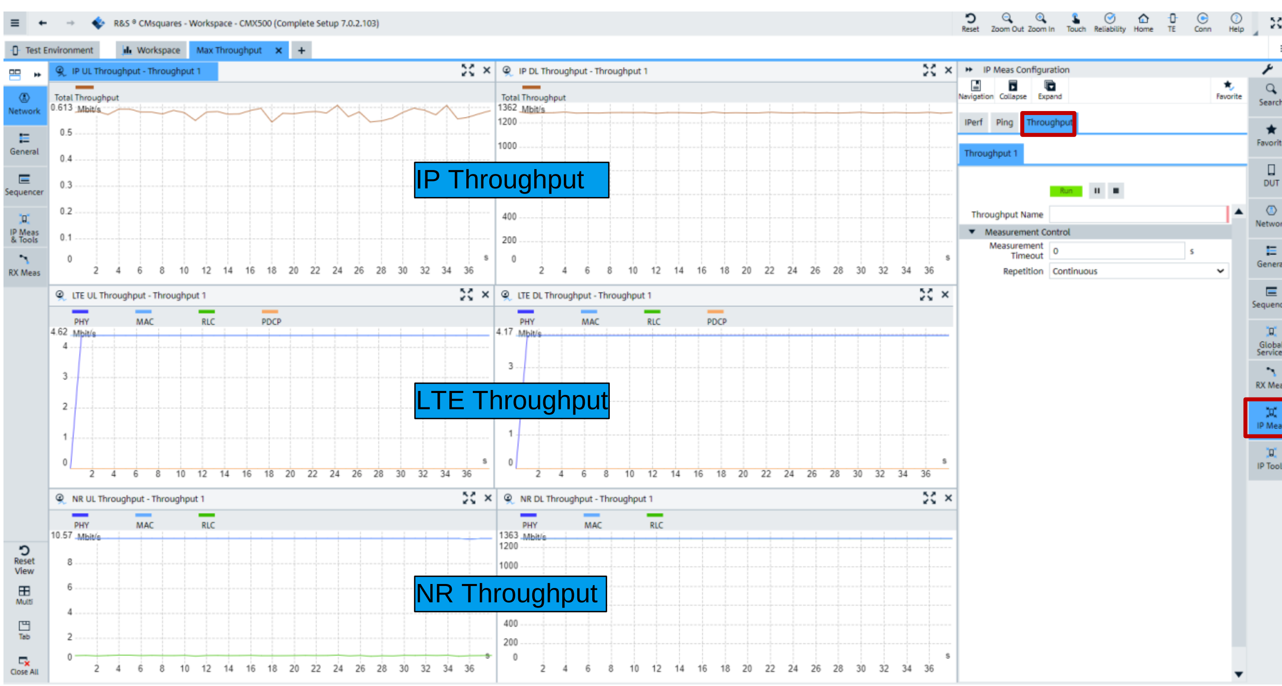


Figure 67 Throughput measurements in CMsquares

3.5 IP Tune

IP Tune is a service that resides in CMX as a frontend for Throughput App (see 3.1.2.2). With IP Tune service, some parameters are fine tuned to explore the limit of the DUT's achievable throughput, e.g. MCS tuning, UDP and TCP tuning.

3.5.1 Preparation

Before IP Tune is started, make sure followings are prepared.

1. Run Max Throughput Wizard as described in 3.4.1 to optimize the cell configurations
2. Launch 'R&S Throughput App2' on DUT (see 3.1.2.2)
3. Follow the steps shown in Figure 68 to enable IP Tune service in case the service is not populated in CMsquares

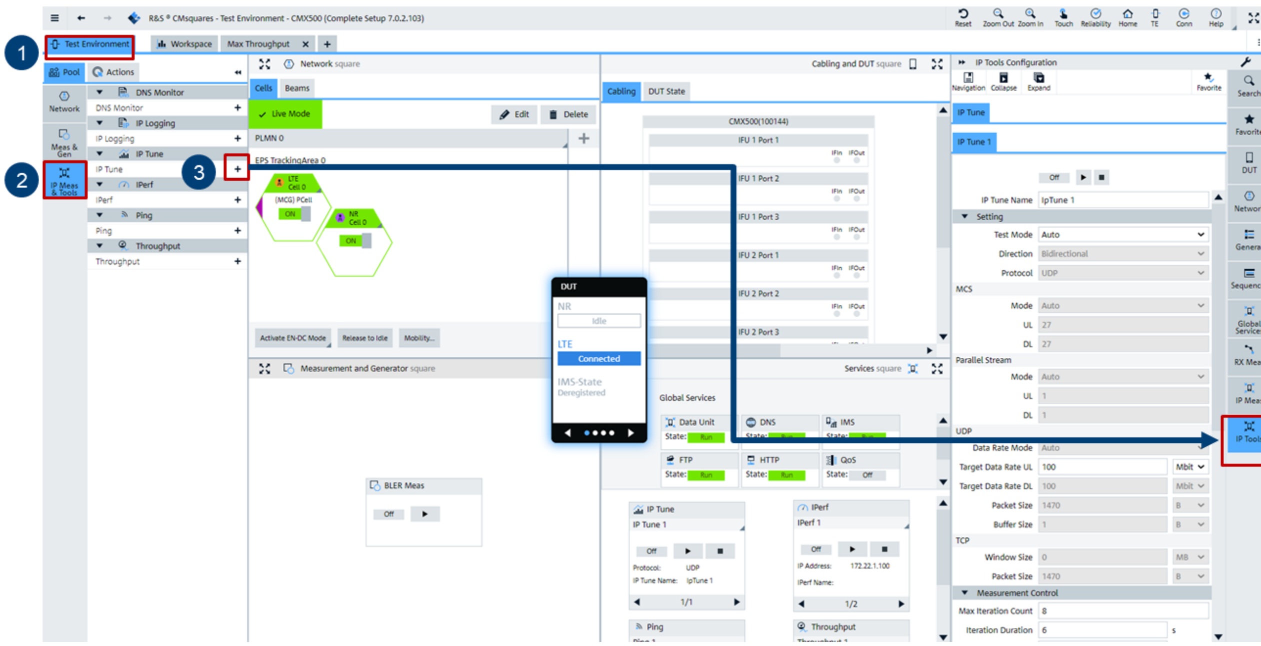


Figure 68 Enable and populate IP tune service in CMsquares

4. Follow the steps shown in Figure 69 to create IP Tune workspace in CMsquares

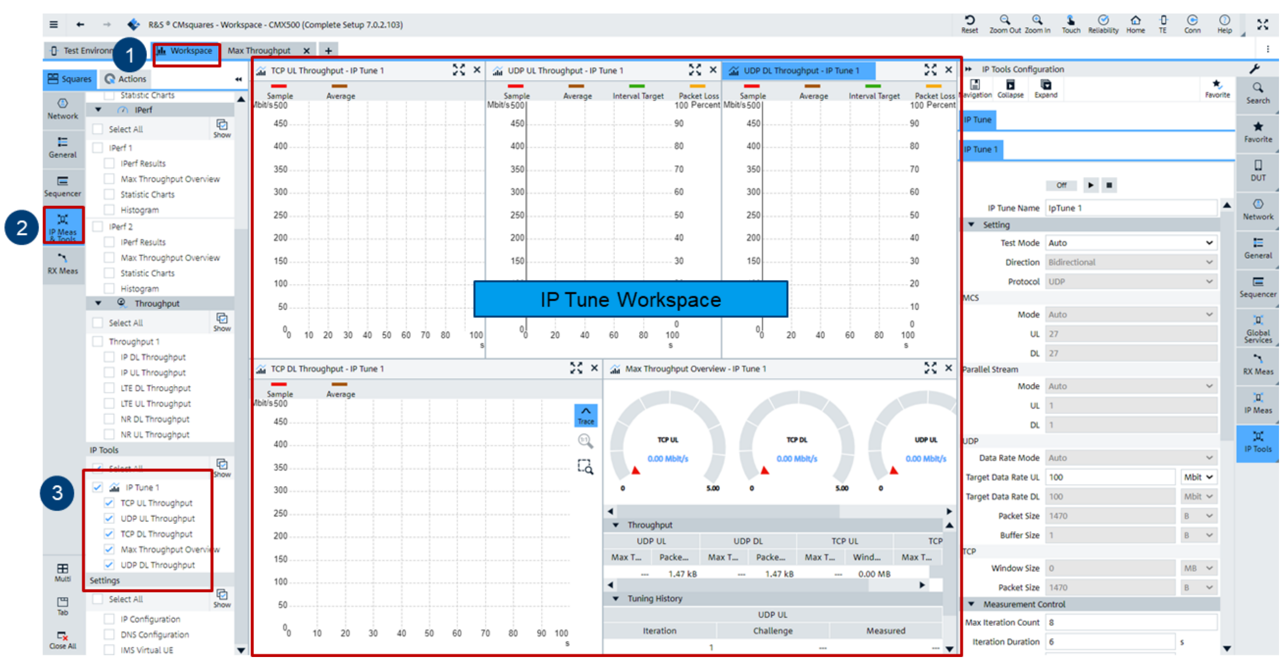


Figure 69 Create IP Tune workspace in CMsquares

3.5.2 Tuning with IP Tune

After the preparation steps described in chapter 3.5.1, it is now ready to use IP Tune for testing with following steps:

1. As shown in Figure 70, select 'IP Tune' service, and if desired, configure IP Tune service to run the test in user defined mode in CMsquares.

   In auto mode, UDP DL, TCP DL, UDP UL, TCP UL throughput will be tuned sequentially. User can define the number of the iterations for the tuning that is configured by 'maximum iteration count' parameter in CMsquares.

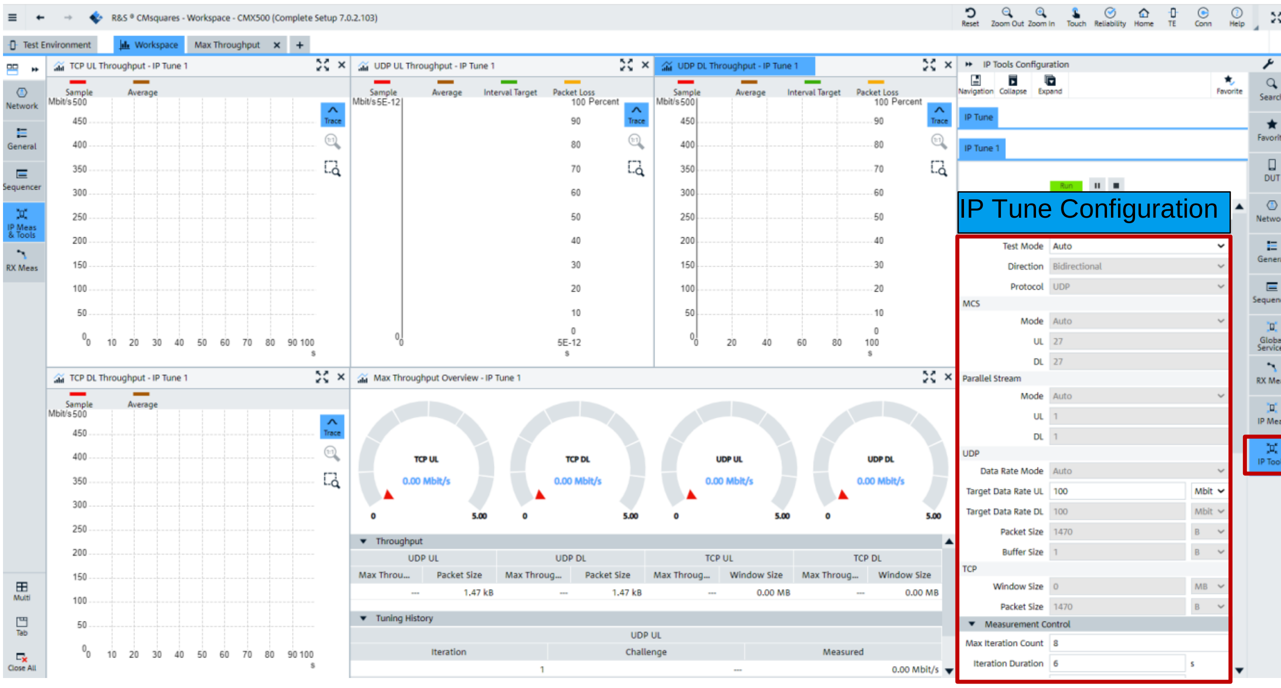


Figure 70 IP Tune configuration in CMsquares

2. Press ▶ key to start IP Tune. Optimal MCS is tuned firstly. This can be observed in event window located at the bottom of the CMsquares (Click on the status bar opens the event window). Example shown in Figure 71 indicates that DUT is tuned with MCS 27 that achieves the maximum throughput of 1882 Mbit/s on MAC layer.

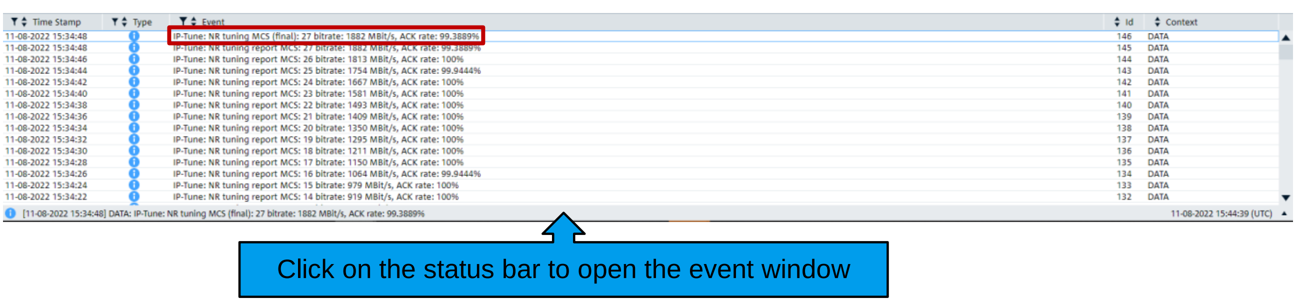


Figure 71 MCS tuning in IP Tune

3. As long as MCS tuning is terminated, the connection status in Throughput App on DUT is changed to 'Connected' status (see Figure 72). The IP tuning is then started in CMsquares. In each iteration, the target throughput is adapting. The tuning is terminated if measured throughput converges the target or the number of iterations reaches the configured number of 'maximum iteration count'.

Figure 72 Connected status shown in RS throughput app

4. Read the tuning results. An example of results is shown in Figure 73.

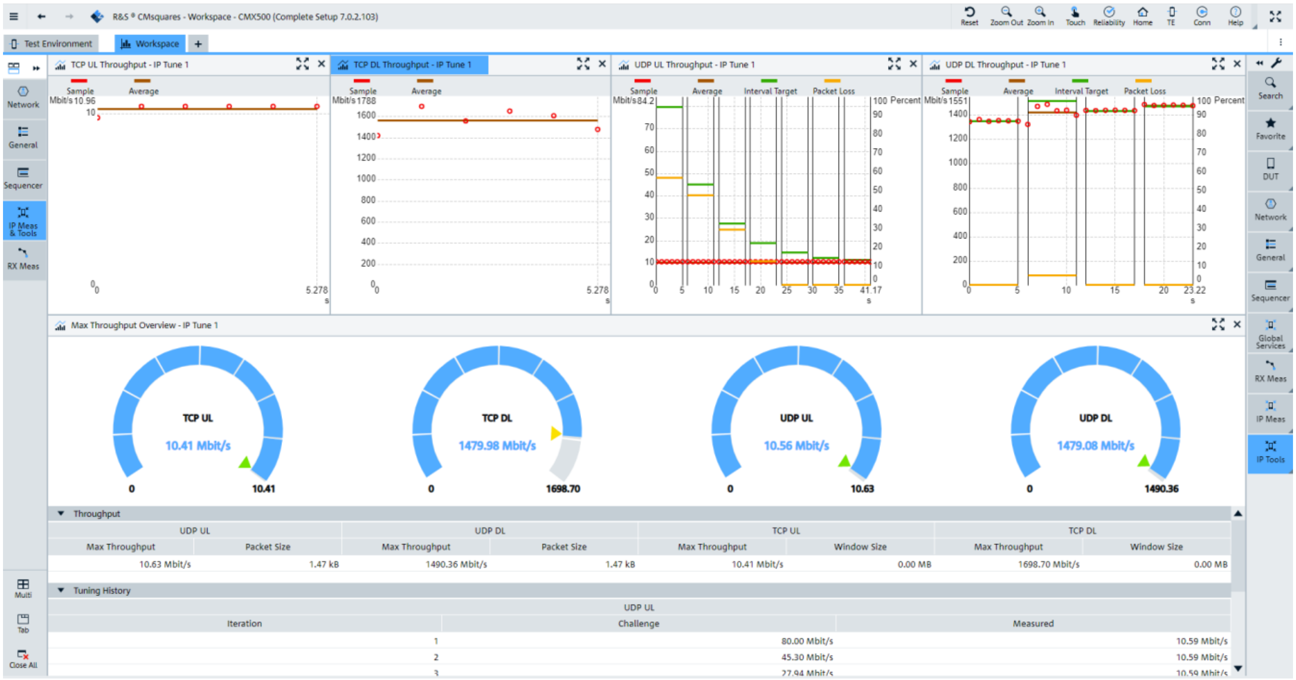


Figure 73 Example of IP Tune results

3.6 Throughput Testing with XLAPI Solution

XLAPI is a scripting interface on CMX utilizing Python as a scripting language which has lean code structure in contrast to object language, e.g. C++, and its instructions are self-explanatory. Python script is created and executed in integrated development environment (IDE), e.g. PyCharm. If desired, the created script can be either executed locally on CMX or remotely from a PC that is LAN-connected to CMX. In addition, the XLAPI script can be converted to a CMsequencer script block for execution in CMsequencer (see Chapter 3.7).

It is beneficial to use XLAPI to cover versatile test needs, when user wants to have higher degree of controls over signaling procedures, L3 parameter settings and test flows.

A bunch of sample scripts are provided in XLAPI script package `kf600x` for various test purposes. Users can simply take those off-the-shelf sample scripts as a basis and elaborate their own ones. For example, sample scripts listed in Table 18 are suitable for NSA throughput verification that contain both DL and UL throughput measurements under various configurations.

Script	Test Purpose	Comment
nr_04_01	NSA end-to-end downlink throughput up to 1 Gbit/s	IPERF performance with NR MIMO 2x2, No additional DM-RS positions
nr_04_02	NSA end-to-end downlink throughput up to 2 Gbit/s	IPERF performance with NR MIMO 4x4, No additional DM-RS positions
nr_04_03	NSA end-to-end downlink throughput up to 942.3 Mbit/s	IPERF performance with NR MIMO 2x2, 1 additional DM-RS position
nr_04_04	NSA end-to-end downlink throughput up to 1.88 Gbit/s	IPERF performance with NR MIMO 4x4, 1 additional DM-RS position

Script	Test Purpose	Comment
nr_04_05	NSA end-to-end downlink throughput of 2.2 Gbit/s, split bearer	IPERF performance in NSA over split bearer
nr_04_06	NSA end-to-end downlink throughput of 3 Gbit/s, split bearer	IPERF performance in NSA over split bearer NR + LTE CA
nr_04_07	NSA FR2 end-to-end downlink throughput	IPERF performance in NSA with FR2 NR Cell
nr_04_07b	NSA FR2 end-to-end downlink throughput (MIMO 2x2)	IPERF performance in NSA with FR2 NR Cell (MIMO 2x2)
nr_04_07c	NSA FR2 end-to-end downlink throughput (MIMO 2x2, 256QAM)	IPERF performance in NSA with FR2 NR Cell (MIMO 2x2, 256QAM)
nr_04_07d	NSA end-to-end throughput, with FR2 and NR 4CA	
nr_04_07e	NSA end-to-end throughput, with FR2 and NR 8CA	
nr_04_08	NSA e2e uplink throughput up to 270 Mbit/s	IPERF performance of 270 Mbit/s with NR PUSCH single-layer transmission
nr_04_09	NSA e2e uplink throughput over MN & SN paths (split bearer)	Uplink performance with uplink data routed over MN and SN paths
nr_04_10	NSA e2e uplink throughput up to 250 Mbit/s, UL MIMO (64QAM)	Uplink performance of up to 250 Mbit/s with NR PUSCH dual-layer transmission
nr_04_11	NSA e2e uplink throughput up to 368 Mbit/s, UL MIMO (64QAM)	Uplink performance of up to 368 Mbit/s with NR PUSCH dual-layer transmission
nr_04_12	NSA e2e uplink throughput up to 1 Gbit/s, UL MIMO (256QAM)	Uplink performance of up to 1 Gbit/s with NR PUSCH dual-layer transmission
nr_04_20	NSA downlink throughput up to 2 Gbit/s. Parameterizable	

Table 18 NR NSA throughput performance XLAPI sample scripts (CMX-KF600X)

In this chapter, some tips are given to show how to deal with the XLAPI test scripts.

3.6.1 Preparations

Make sure followings are prepared before running the XLAPI test scripts.

- ► Activate the DUT and make sure the RF connections configured reflects the physical cabling on CMX (refer to 3.2 for more details)
- ► Install the XLAPI package via Installation Manager on CMX (see XLAPI user manual [15], Chapter 3)
- ► Load the installed XLAPI package as a project in PyCharm (see XLAPI user manual [15], Chapter 3.4).

  Go to menu 'File' > 'Open', navigate to XLAPI version that is intended to be loaded, e.g. 10.77.3, and confirm with 'OK' button (see Figure 74)

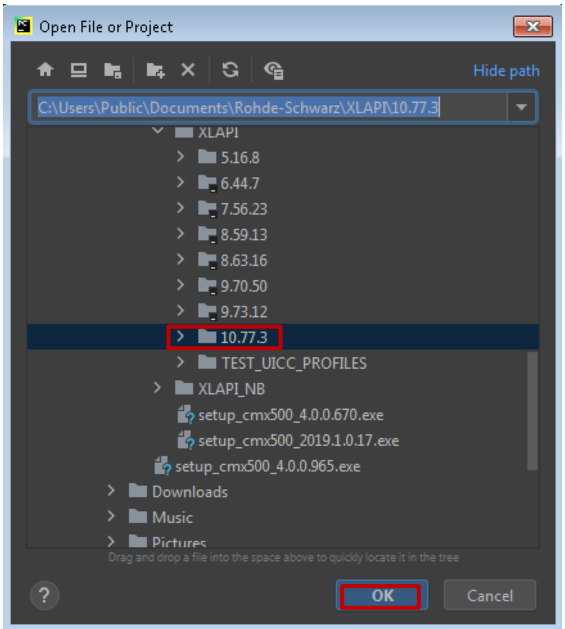


Figure 74 Open a project in PyCharm

► Confirm with 'This Window' in the upcoming pop-up window (see Figure 75)

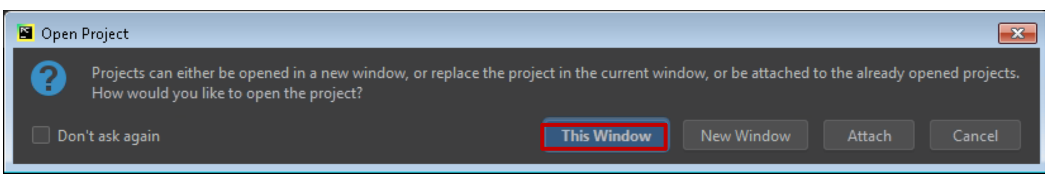


Figure 75 Confirmation window

► Now the actual project is loaded in PyCharm, e.g. 10.77.3. As a result, the whole project structure as shown in Figure 76 is created, including the virtual environment.

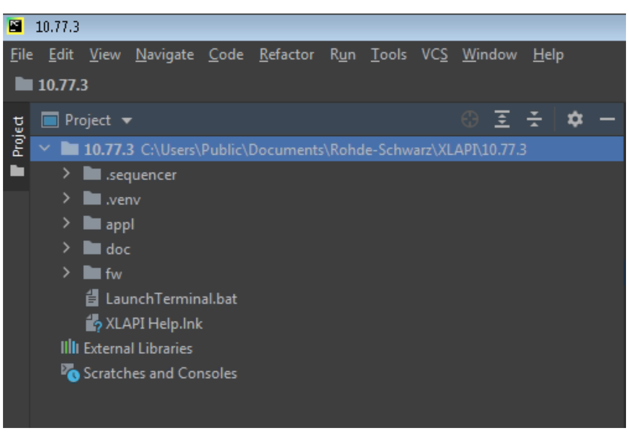


Figure 76 XLAPI project structure

Make sure that the '.venv' folder is not in red. Otherwise, it means the Python interpreter is not associated as shown in Figure 77.

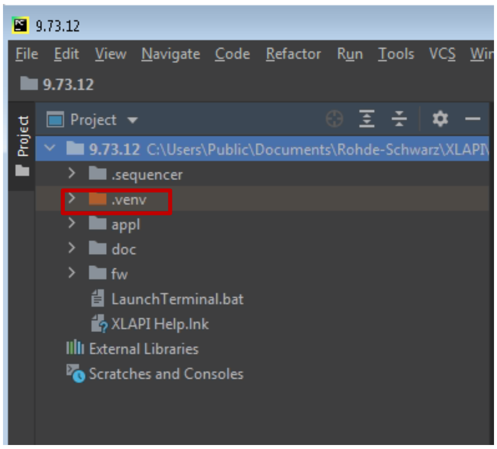


Figure 77 .venv in red (missing Python interpreter)

Do following steps to associate a Python interpreter to the project

► Setup the Python interpreter in PyCharm. Go to 'File' > 'Settings' > 'Project' > 'Python Interpreter'

► Press on ⚙ next to the selection field (see Figure 78), select option 'Add …' in the opened menu,

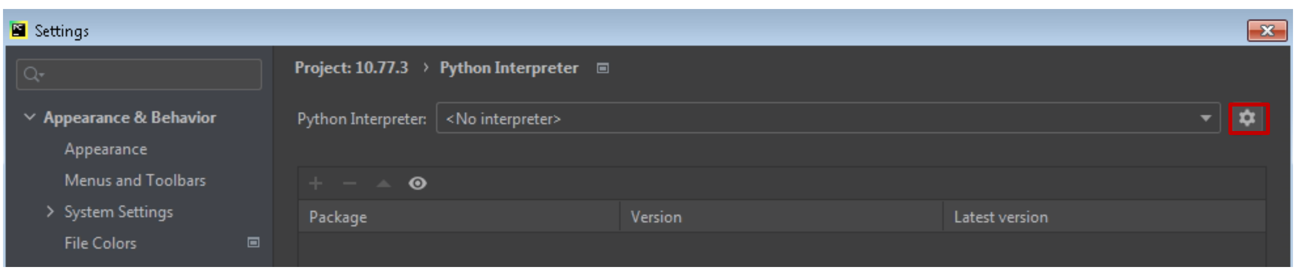


Figure 78 Empty Python interpreter

► Choose option 'Existing environment', specify the path of the interpreter associated with the current XLAPI version if it is not automatically given in the configuration line, and then click on 'OK' to confirm (see Figure 79).

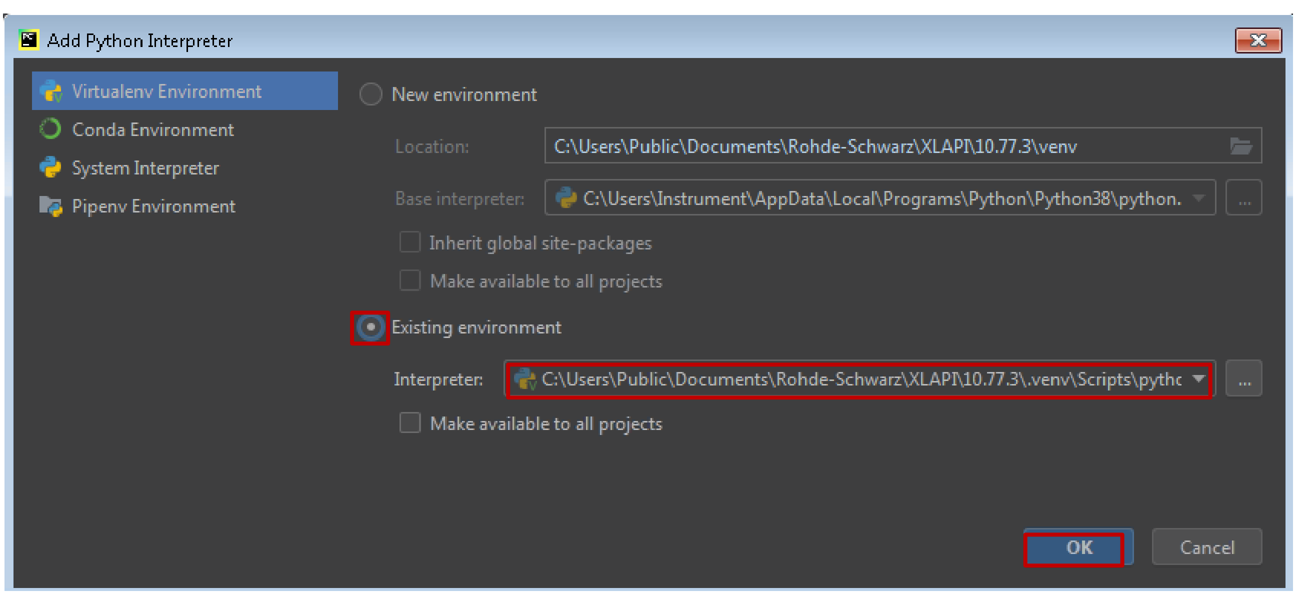


Figure 79 Specify Python interpreter

► Now, the XLAPI python script packages are associated with the Python interpreter. As indication of successful association, the packages are linked with the corresponding versions as shown in Figure 80.

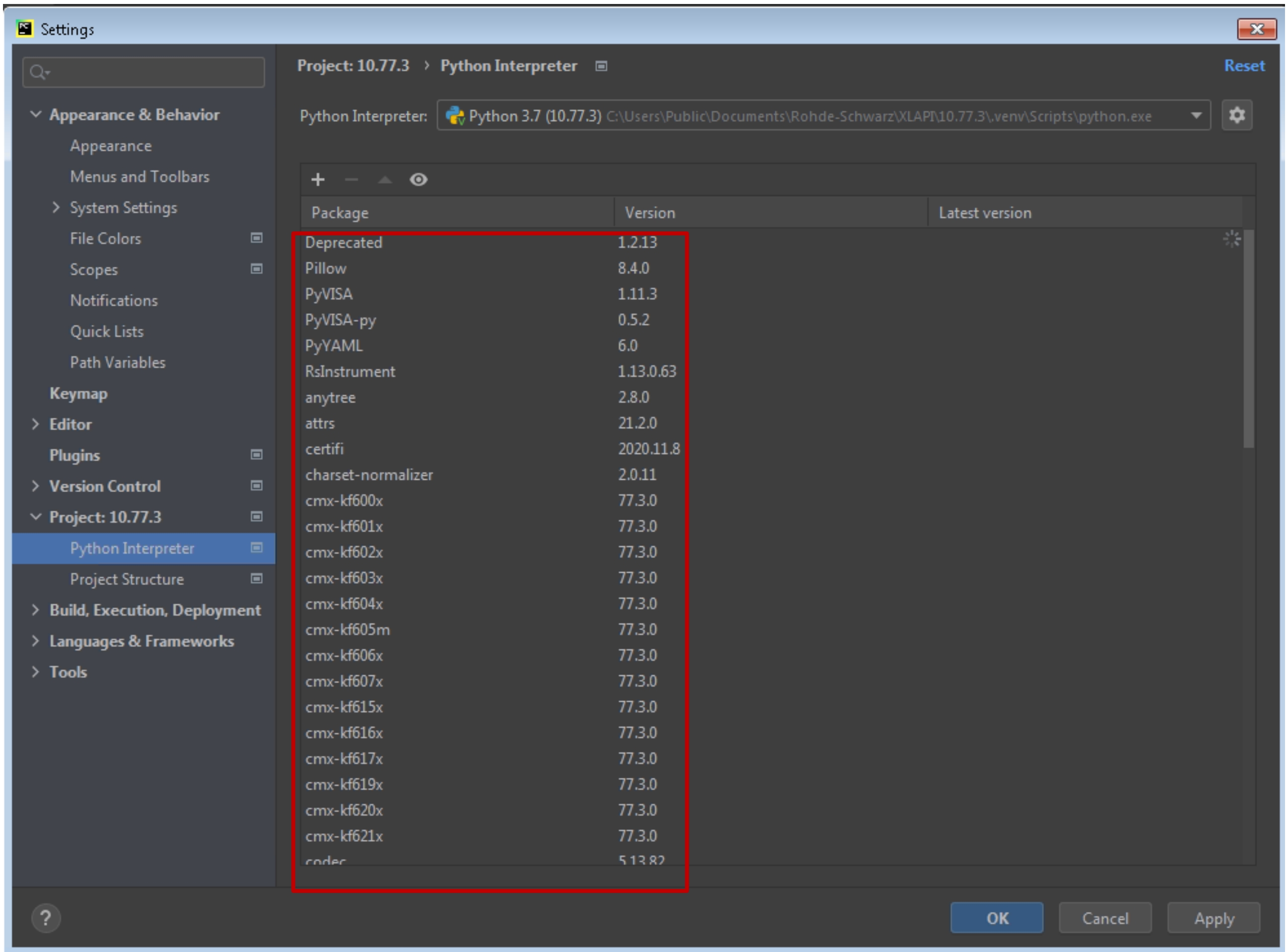


Figure 80 XLAPI packages associated with Python interpreter

3.6.2 Modify Sample Script

After having set up the XLAPI environment properly, XLAPI project tree structure is created as given in Figure 81.

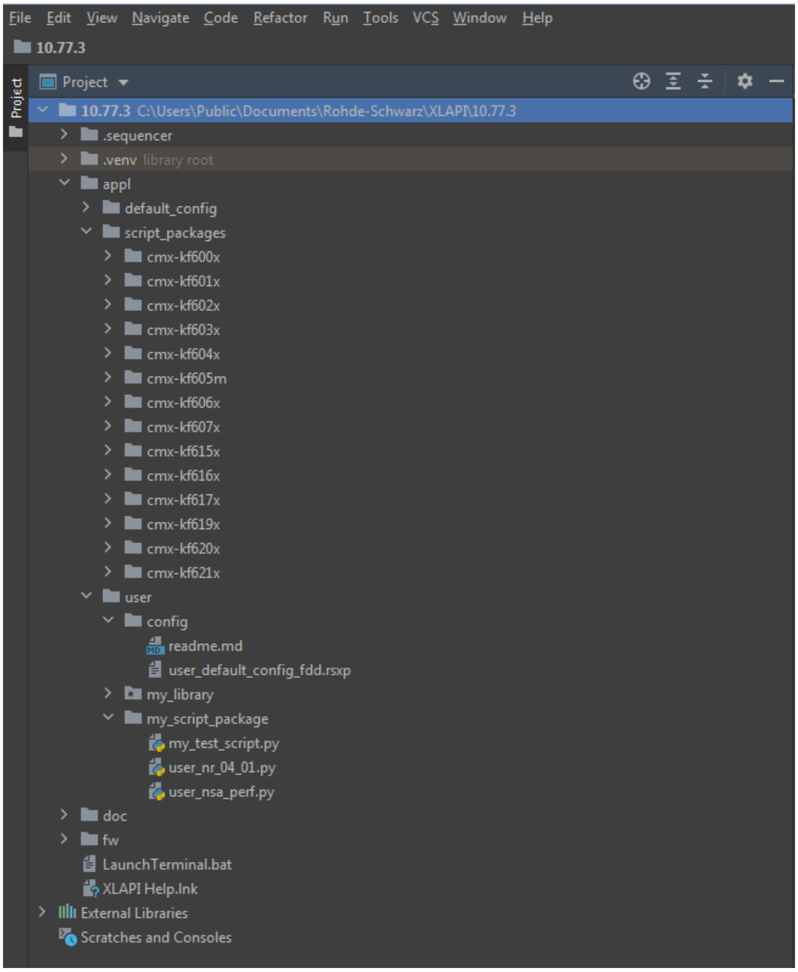


Figure 81 Tree structure of XLAPI project

Table 19 lists few of the important XLAPI project folders.

Project folder	Comment
`appl\default_config`	Original default configuration file if it is specified by the script.
`appl\script_packages`	Original script packages from the XLAPI installation
`appl\user`	This folder should contain user defined configuration files, test scripts and required libraries

Table 19 Important XLAPI project folders

It is highly recommended that user should always work on the copied version of script, library and configuration files that should be located in `user` folder, while keep the original ones untouched.

For example, migrate a sample XLAPI script, e.g. `nr_04_01.py` including its's library to user defined script either in Windows file explorer or directly within the XLAPI project structure in PyCharm.

Step 1: Copy original sample script and its library to user folder and include adaptation

The path given in the following text is relative to the XLAPI installation folder:

`C:\Users\Public\Documents\Rohde-Schwarz\XLAPI\<XLAPI version>`[20]

► Copy sample script and rename it to a new name `user_nr_04_01.py`

Copy `appl\script_packages\cmx_kf600x\nr_04_01.py`

[20] For example, the XLAPI version 10.77.3

to `appl\user\my_script_package\user_nr_04_01.py`

- Copy library `nsa_perf.py` and rename it to a new name `user_ nsa_perf.py`

  Copy `appl\script_packages\cmx_kf600x\nsa_perf.py`

  to `appl\user\my_script_package\user_ nsa_perf.py`

- Adapt the included library name and location in the new script, i.e.

  replace the code

  `from cmx_kf600x.nsa_perf import NsaDlPerfScenario`

  with

  `from appl.user.my_script_package.user_nsa_perf import NsaDlPerfScenario`

Step 2: Add configuration file (optional)

Configuration file can be optionally specified in the script. If otherwise stated in the script, the default configurations are given by the XLAPI environment.

To apply customized configuration file to the script, below steps have to be followed:

- Recommended way is to copy the default configuration `default_config_fdd.rsxp` (a sort of configuration template file) from the folder `appl\default_config\cmx_kf600x` to folder `appl\user\config` under a new name, e.g. `user_default_config_fdd.rsxp`
- Alternatively, create configuration file from a scratch. The syntax of the configuration file can be referred in 'Test Parameters' section of the XLAPI online help document as shown in Figure 90. Place the created configuration files (e.g. `user_default_config_fdd.rsxp`) in folder `appl\user\config`.
- In both ways, make sure to add following two lines in the customized script `user_nr_04_01.py` to apply the configuration file. The configuration file name is given in the parentheses.

```
from xlapi import settings
```

```
settings.session.set_test_param_files("user_default_config_fdd.rsxp")
```

Hereafter the content of an example configuration file `user_default_config_fdd.rsxp` is shown.

```
# Configure LTE cell
- LteCell:
    name: LTE Cell 0
    config:
        - total_dl_power: -50 dBm
        - band: 1
        - bandwidth: MHZ_20


# Configure NR cell
- NrCell:
    name: NR PSCell 0
    config:
        - total_dl_power: -25 dBm
        - dl_ref_a_offset:
            band: 78
            ref_a: MID
        - carrier_bandwidth_and_scs:
            bandwidth: MHZ_100
            scs: KHZ_30
```

Important: make sure the LTE and NR cell name, e.g. 'LTE Cell 0' and 'NR PSCell 0', should match the cell names coded in the script so that the configurations are correlated.

3.6.3 Script Structure and Settings

In a XLAPI throughput test script, e.g. `nr_04_01.py`, it has the structure described in this chapter.

`Scenario` is an unparameterized test definition from which the user can derive concrete scenarios, e.g. `NsaDlPerfScenario` who inherits the attributes of basic class scenario that consists of following components:

Initialization

Specify BLER limit in NR which is checked by the BLER measurement, e.g. limit 0.1%

```
max_nr_bler = 0.1
```

When the `NsaDlPerfScenario` is called, object initializing with respect to the DL MIMO mode configuration, additional DMRS position and security settings is performed.

Example:

Initialize MIMO mode, enable/disable additional DMRS Position and security

```
def __init__(
    self,
    dl_mimo_mode: DownlinkMimoMode.Enum,
    additional_dmrs_pos_enabled: bool,
    security_enabled: bool = False,
):
```

Preamble

Preparations for the test. Typically, the preamble will create cells, configure the scheduler and take care of the signaling that is required to bring the DUT in the correct state to perform the test, e.g. ENDC activation in NSA case.

With below example, optimizations with respect to frequency range, TDD slot format, scheduler are performed so that DL max throughput testing in TDD mode is prepared.

Example:

Create LTE and NR cells

Cell names should match the ones specified in configuration file if it applies

```
lte_cell = network.create_lte_cell(name="LTE Cell 0")
nr_cell = network.create_nr_cell(name="NR PSCell 0")
```

Configure frequency range to LOW

```
nr_cell.set_dl_ref_a_offset(band=nr_cell.get_band(), ref_a=FrequencyRange.LOW)
```

Configure TDD slot format

```
nr_cell.set_tdd_ul_dl_pattern(
    num_dl_slots=8, num_dl_symbols=12, num_ul_slots=1, num_ul_symbols=1
)
```

Configure scheduler

```
nr_scheduler = dut.get_scheduler(nr_cell)
nr_scheduler.configure_ue_scheduler(
    dl_mcs=27,
    dl_rb_alloc=(0, 273),
    dl_mcs_table=McsTable.QAM_256,
    dl_sliv=(1, 13),
    ul_mcs=10,
    ul_rb_alloc=(0, 273),
    ul_sliv=(0, 13),
    dl_mimo_mode=self.dl_mimo_mode,
    dl_dci_format=NrDciFormat.DCI_FORMAT_1_1,
    dl_dci_al=AggregationLevel.AGGREGATION_LEVEL_2,
    ul_dci_format=NrDciFormat.DCI_FORMAT_0_1,
    ul_dci_al=AggregationLevel.AGGREGATION_LEVEL_2,
    configure_dl_partial_slot=True,
)
```

In addition, handling of slot 0 scheduling in NR (see 2.2.4) is automatically considered by NR scheduler.

Testbody

The test purpose of the scenario. For throughput testing, it includes the BLER and iPerf measurements and deliver the verdict based on the given BLER limit (e.g. 1%) defined globally in the library.

In the testbody, DUT control is necessary either in a manual way (user interaction on DUT) or in automatic way (via Automation Manager). Refer to 3.6.4 to know more about the DUT control in conjunction with Automation Manager.

Postamble

Any cleanup that is required after the testbody has been executed, e.g. reset the network emulation, collect and close the logfiles.

Figure 82 shows the output of XLAPI throughput script where the scheduled, actual achieved throughput are presented as well as the BLER measurement after the script execution.

```
[INFO] Powering on the UE
[INFO] [Signaling] Starting LTE attach on LteCell LTE Cell 0
[INFO] [Signaling] Completed LTE attach on LteCell LTE Cell 0
[INFO] [Signaling] Starting NSA dual connectivity on NrCell NR PSCell 0
[INFO] [Signaling] Starting Add EPS dedicated bearer
[INFO] [Signaling] Completed Add EPS dedicated bearer
[INFO] [Signaling] Completed NSA dual connectivity on NrCell NR PSCell 0
[INFO] Starting testbody
[INFO] EPS default bearer ID: 5, UE IP address: 172.22.1.100
[INFO] BLER measurements starting
[INFO] BLER measurements stopped
[INFO] Results for cell NR PSCell 0:
BLER:       0.0%
ACK Count:  12600
NACK Count: 0
DTX Count:  0
Block Count: 12600
Acked data: 566.61 MB
Nacked data: 0 B
Dtxed data: 0 B
Scheduled throughput: 647.55 Mbps
Acked throughput: 647.55 Mbps
Verdict: Verdict.PASS
Confidence verdict: Verdict.NONE

[INFO] [VerdictHandler::set_verdict] PASS: Achieved throughput: 647.55 Mbps, BLER: 0.0% <= 10.0%
[INFO] Nr0401Scenario script completed
[INFO] ################################################################################
[INFO] Final verdict: PASS: Achieved throughput: 647.55 Mbps, BLER: 0.0% <= 10.0%
[INFO] --------------------------------------------------------------------------------
[INFO] C:\Users\Public\Documents\Rohde-Schwarz\XLAPI\10.77.3\.venv\lib\site-packages\xlapi\scenario.py:277: 2022-04-11T15:00:57.969328 - NONE: Initial NONE
[INFO] C:\Users\Public\Documents\Rohde-Schwarz\XLAPI\10.77.3\appl\user\my_script_package\user_nsa_perf.py:222: 2022-04-11T15:03:19.169405 - PASS: Achieved throughput: 647.55 Mbps, BLER: 0.0% <= 10.0%
[INFO] ################################################################################
[INFO] Collecting log files ...
[INFO] [Ims]  IMS Server Started.
[INFO] Message log file:  file://CMW50050-157043/D/Rohde-Schwarz/CMW/Log/4.0.0/2022-04-07_18-21-57/sfa/NetworkEmulation12/signaling.msglog
[INFO] Session log files: file://CMW50050-157043/D/Rohde-Schwarz/CMW/Log/4.0.0/2022-04-07_18-21-57/sfa/NetworkEmulation12
[INFO] XLAPI log file:    file:///D:/Rohde-Schwarz/CMW/Log/4.0.0/2022-04-07_18-21-57/xlapi/user_nr_04_01.2022-04-11_15-00-58/xlapi.log
```

Figure 82 Output of XLAPI throughput script

3.6.4 DUT Control

3.6.4.1 Power On/Off

It is necessary to control DUT's power on/off status in a test script. There are two ways to bring a DUT in such a status, either through airplane mode or normal power on/off the DUT. In either way, XLAPI script communicates with Automation Manager (refer to Chapter 3.1.2.3) who is then translates the operation expression (see unique expressions in Table 20) into a corresponding action, e.g. AT command to switch on/off airplane mode etc. Important here is to make sure that Automation Manager is configured with the exact expressions as given in Table 20. Please be noted that the applied expression differs in both modes.

	Expression	
	Power On	Power Off
Use airplane mode to control DUT	Please switch off the airplane mode.	Please switch on the airplane mode.
Use normal mode to control DUT	Please power on the DUT.	Please power off the DUT.

Table 20 Command expression of DUT control in Automation Manager

In case DUT is controlled by airplane mode using AT commands, the mapping between the applied expression and its' associated action is summarized in Table 12 on page 29. As already mentioned in chapter 3.1.2.3, the action is not only limited to AT command. Any DUT supported control commands can be defined in Automation Manager and utilized for the DUT control. An example of using batch file as defined action is shown in 3.6.4.2.

In XLAPI, functions such as DUT power on, power off and power cycle (power off followed by power on) are provided. In each function, either airplane mode or normal mode is supported.

DUT control of throughput sample scripts in Table 18 is either contained directly in the test script itself or included in the `NsaDlPerfScenario` class within `nsa_perf` module.

Examples shown below use airplane mode for DUT control which is indicated by the argument `airplane_mode=True` when the function is called. If the argument is omitted, the normal mode will be used.

```
dut.control.power_off(airplane_mode=True)
```

```
dut.control.power_on(airplane_mode=True)
```

3.6.4.2 Control of iPerf Application

iPerf application can be handled by the Automation Manager as well.

In the following text, an example is given to show how to include the iPerf in the XLAPI script and through Automation Manager to start and stop the iPerf application on DUT.

Start iPerf

Include the iPerf configuration for CMX in XLAPI script, e.g. set the mode (client or server), protocol (UDP or TCP) and port (5001), to start iPerf.

```
# Set up iPerf on CMX side
meas.delete_all_iperfs()
perf = meas.create_iperf()   perf: [IPerf] Iperf 1
perf.mode = IPerfMode.CLIENT
perf.protocol = IPerfProtocol.UDP
perf.configure_address(dut.state.pdn_connections[0])
perf.port = 5001
```

An output string based on the above settings is created that consists of indication of mode, protocol and port. In the example showing here, the output string turns to be: `Start iPerf on DUT in SERVER mode for Protocol: UDP, Port: 5001`

```
output_string = (
    f"Start iPerf on DUT in SERVER mode for "
    f"Protocol: {perf.protocol.name}, Port: {perf.port}"
)
dut.control.msg_box(output_string)
```

The output string is passed to Automation Manager when the function is called during the script execution, provided that DUT control in CMsquares is set to 'Automatic'. Otherwise, the output string is displayed in the pop-up message box in CMsquares and manual interaction on DUT is expected (see 3.1.2.3).

In Automation Manager, as we learn from 3.1.2.3, the command expression needs to be associated with the corresponding action. A batch file (`adb_start_iperf.bat`) containing ADB/ADB shell commands to execute iPerf on Android device is adopted here. Make sure that adb.exe and iperf2.1 binary for Android OS are located in the folder C:\android_adb before the batch file is called. Figure 83 presents the contents of the batch file.

In case iPerf application has already been installed on the Android DUT, line 27 to 31 of the batch file can be bypassed. Simply use the ADB shell command in line 32 to execute the iPerf application (including the full path of the iPerf binary on the DUT).

adb_start_iperf.bat

```
1  @echo OFF
2  if "%3" == "" goto args_count_wrong
3  if "%4" == "" goto args_count_ok
4
5  :args_count_wrong
6  echo ERROR: Please specify MODE, PROTOCOL and PORT as arguments
7  exit /b 1
8
9  :args_count_ok
10
11 set MODE=%1
12 set PROTOCOL=%2
13 set PORT=%3
14
15 if "%MODE%" equ "SERVER" (
16   set MODE=-s
17 ) else if "%MODE%" equ "CLIENT" (
18   set MODE=-c
19 )
20
21 if "%PROTOCOL%" equ "UDP" (
22   set PROTOCOL=-u
23 )
24
25 REM Load iperf to UE and start it in C:\android_adb
26 @echo ON
27 pushd C:\android_adb
28 adb root
29
30 adb push iperf2.1 /data/local/tmp
31 adb shell chmod 755 /data/local/tmp/iperf2.1
32 adb shell /data/local/tmp/iperf2.1 %MODE% %PROTOCOL% -i 1 -w 8M -g 3000 -p %PORT% > iperf_%1_%PORT%.log
33 popd
```

Figure 83 Batch file (adb_start_iperf.bat) to start iPerf on Andriod OS

We see the batch shown above contains the input arguments. The value of each argument is parsed in the Automation Manager using its regular expression parser feature (for more detailed information about the usage, please refer to [14]) that allows the positions marked as (.+) in the command expression to be parsed as the command line arguments (mode, protocol, port) to the batch file. This operation is shown in Figure 84.

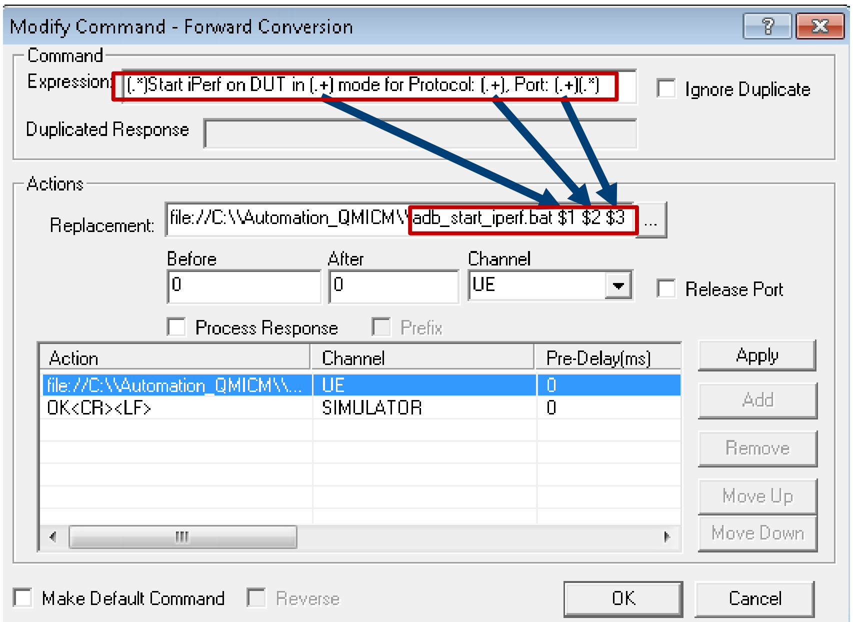


Figure 84 Command expression and parse of the argument value to its associated action to start iPerf

Stop iPerf

To stop iPerf service, include the code line below in the XLAPI script with string 'Stop iPerf'.

```
dut.control.msg_box("Stop iPerf")
```

A batch file (`adb_stop_iperf.bat`) is created that contains the ADB shell command to terminate the iPerf service (see Figure 85).

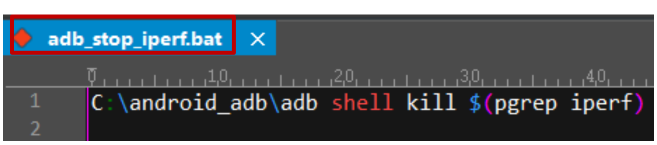


Figure 85 Batch file (adb_stop_iperf.bat) to terminate iPerf on Andriod OS

In the Automation Manager, associate the command expression (Stop iPerf) that matches the string given in the XLAPI script with the batch file as the action to terminate the iPerf on DUT (see Figure 86).

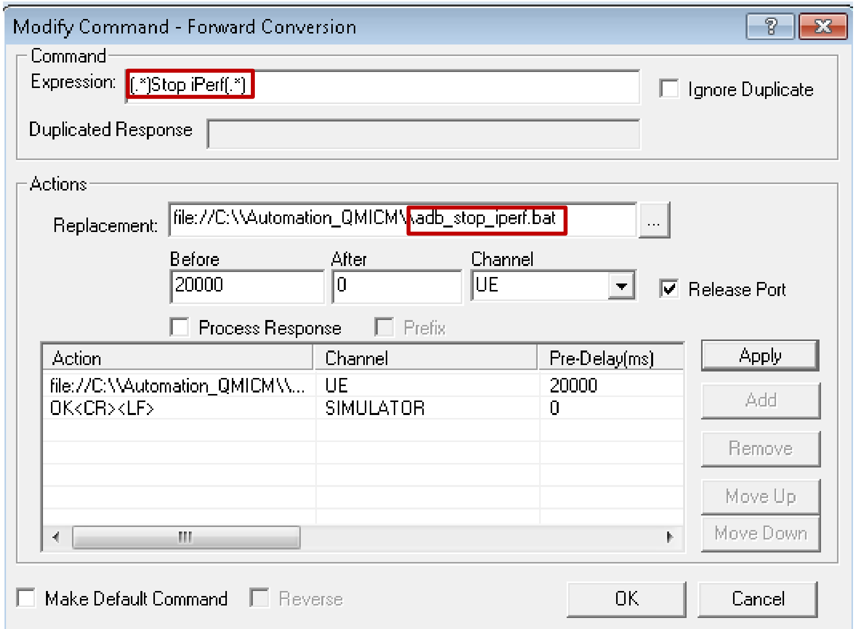


Figure 86 Command expression and its associated action to terminate iPerf

3.6.5 Run the Script

Navigate to the script that intends to be run in PyCharm project tree structure and right mouse click on the script, e.g. '`user_nr_04_01.py`', select drop down option 'Run user_nr_air04_01' as shown in Figure 87.

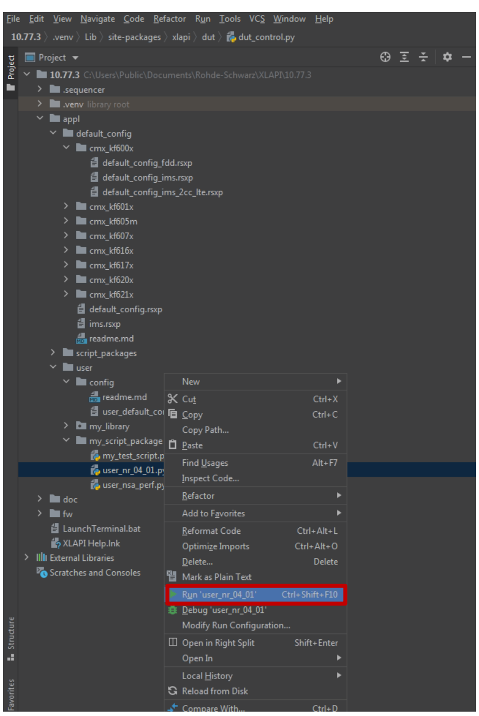


Figure 87 Run a XLAPI script in PyCharm

3.6.6 XLAPI References

Comprehensive XLAPI references are provided on CMX.

XLAPI interactive tutorials is highly recommended for the new comer who starts to XLAPI to do the scripting on CMX. The tutorial is assessible either from CMsquares burger menu > 'XLAPI Interactive Tutorial' or CMsquares home > select application 'XLAPI Interactive Tutorial'. Figure 88 shows the landing page of XLAPI interactive tutorials.

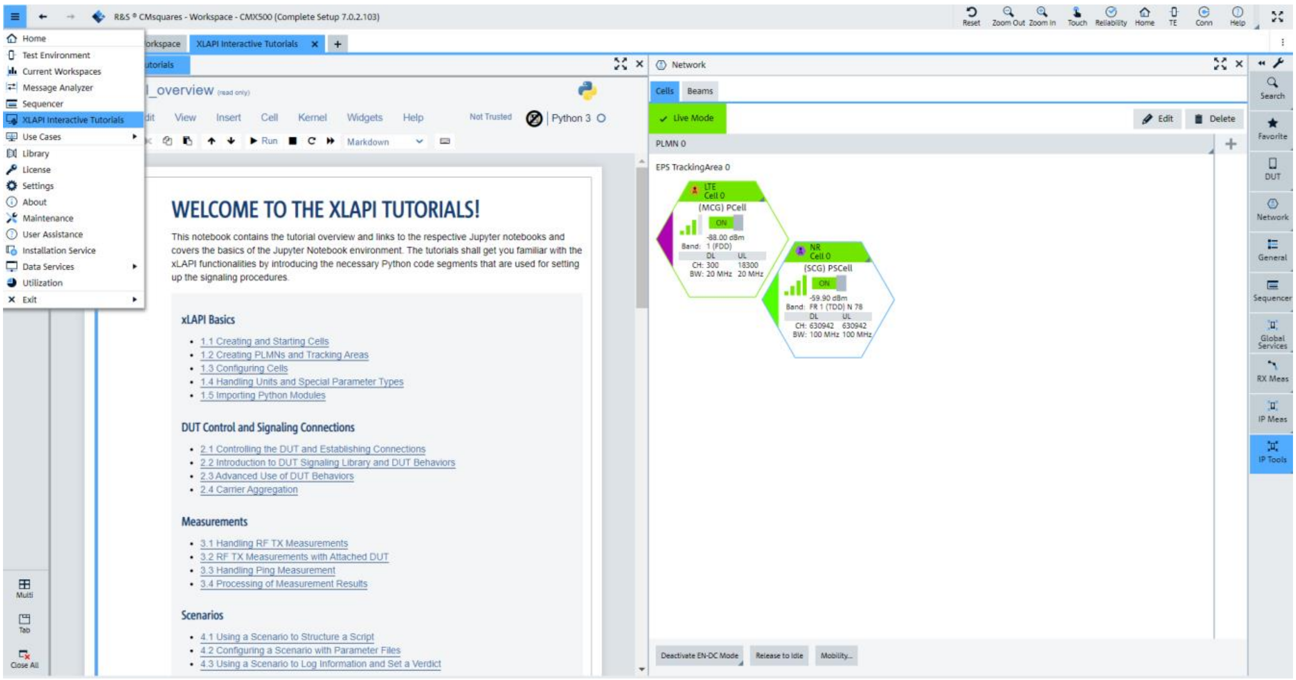


Figure 88 XLAPI interactive tutorials

Further reference can be obtained in XLAPI online document that covers XLAPI usage in conjunction with signaling controls, measurements etc. The online document is accessible from the shortcut 'XLAPI Help' in XLAPI installation folder `'C:\Users\Public\Documents\Rohde-Schwarz\XLAPI\<XLAPI version>`[21]' (see Figure 89).

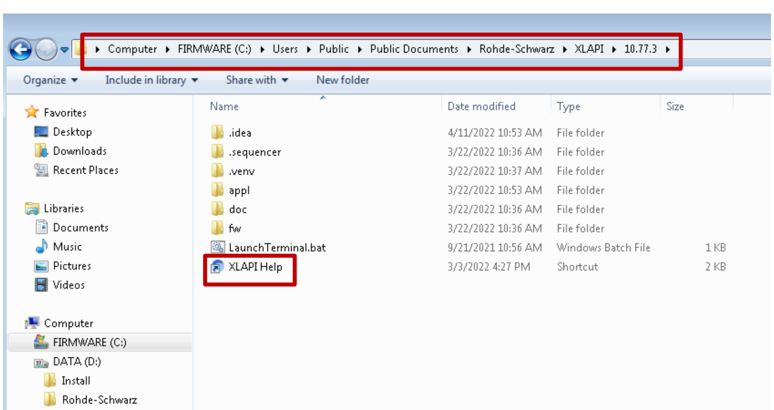


Figure 89 Access to XLAPI online document

Figure 90 below presents the landing page of the XLAPI online document.

[21] Placeholder contains the active XLAPI version, e.g. 10.77.3

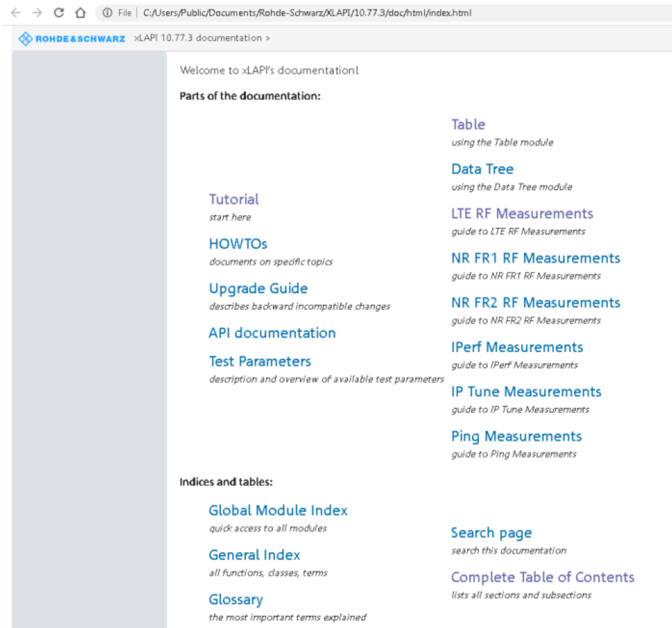


Figure 90 XLAPI online document landing page

3.7 Throughput Testing in CMsequencer

CMsequencer is a sequencer tool integrated in the CMsquares. It supports test script creation, configuration and test automation with flow controls.

High level overview of the test run in CMsequencer is presented in this chapter. For more detailed information, please refer to CMsequencer user manual [16].

CMsequencer is accessible either directly from the home page of CMsquares (see Figure 91)

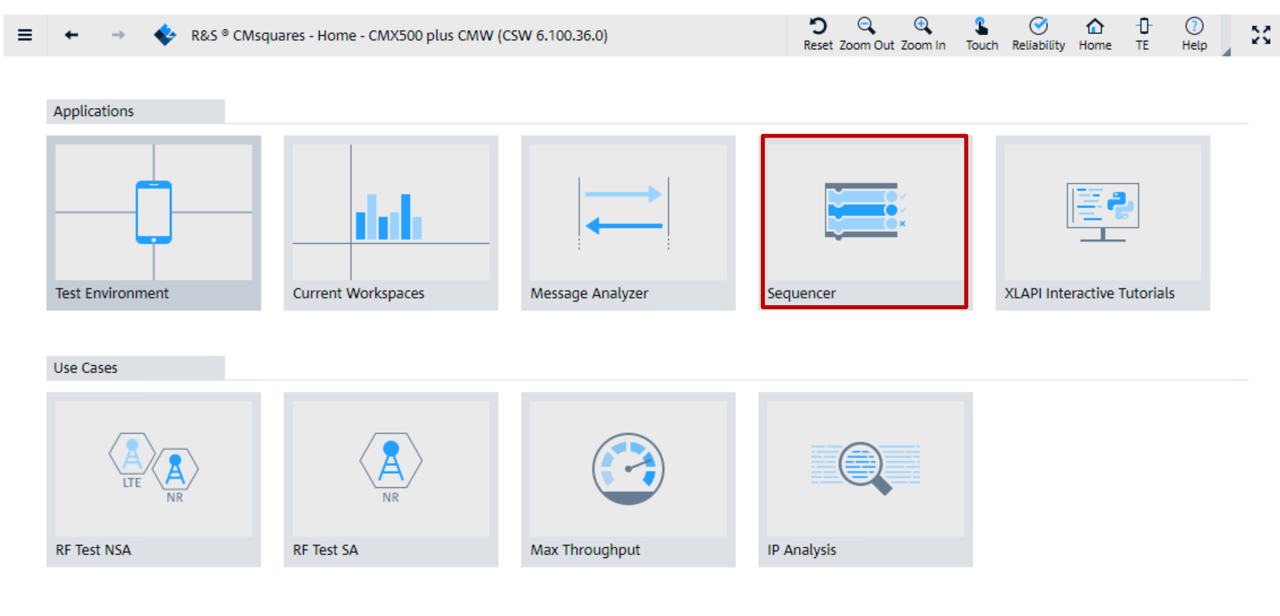


Figure 91 Launch CMsequencer from CMsquares home page

Or, alternatively, from the CMsquares burger menu > 'Sequencer' (see Figure 92)

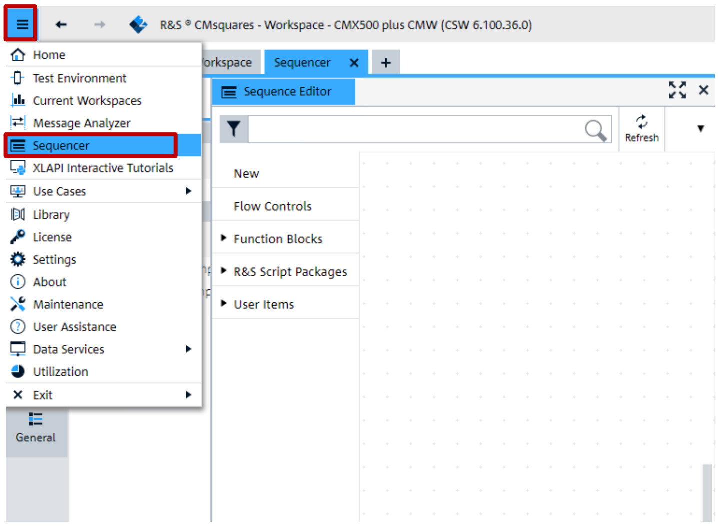


Figure 92 Launch CMsequencer from CMsquares menu

Upon selection, a workspace called 'Sequencer' is created in which the squares like 'Sequence Editor', 'Test Report', 'Recent Test Runs' are populated as shown in Figure 93. Refer to [16] for more details about CMsequencer GUI.

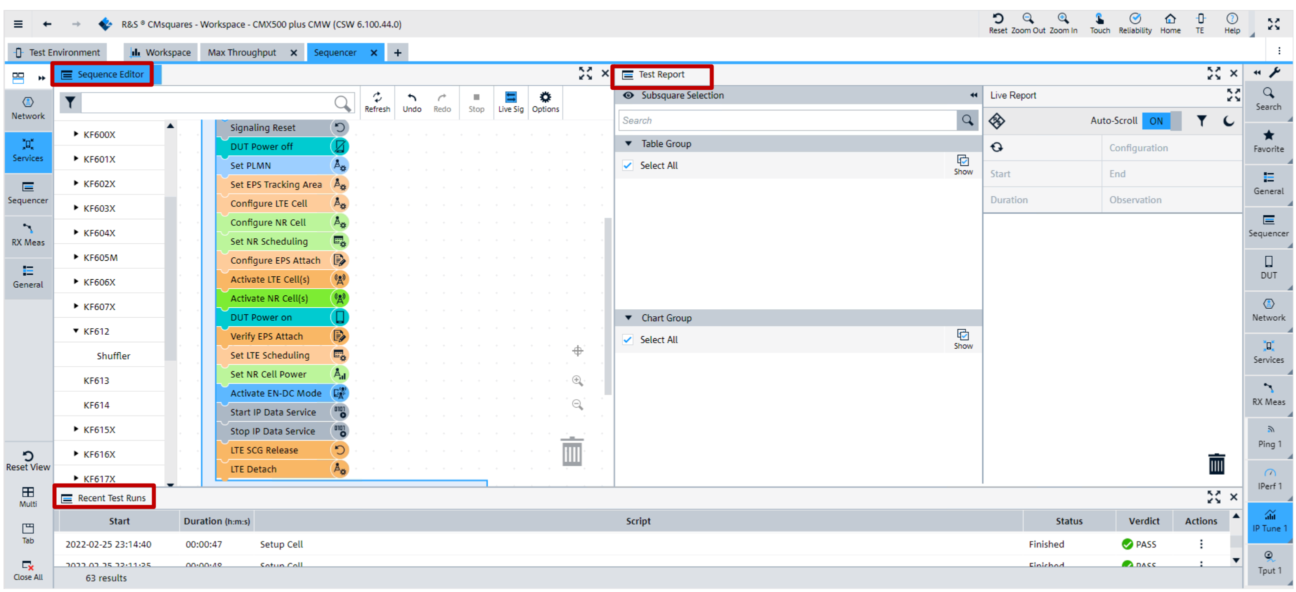


Figure 93 Squares in workspace 'Sequencer'

3.7.1 Test Scripts Configuration and Execution

In general, CMsequencer test scripts can be created by drag and drop the building blocks from 'Function Blocks' area in 'Sequence Editor' square. Multiple test scripts can be included and executed in the arranged order in one test plan.

CMsequencer offers plenty of sample scripts with which user just adjusts the parameter settings of the function block of interest or modifies the sample script with minimal effort to meet the test needs. This saves the time and effort for the user to create a test script from the scratch.

As long as a user test script is created (either modified from an existing script, newly created, or converted from XLAPI etc.), it is saved and located under 'Sequencer Editor' > 'User Items' > 'Test Scripts' as shown in Figure 94.

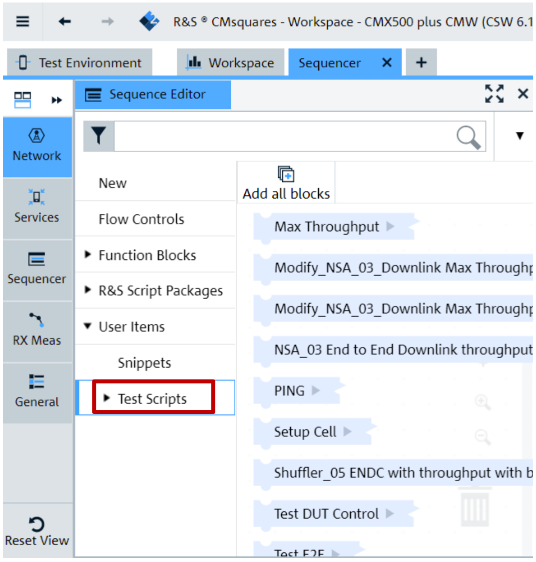


Figure 94 Location of user defined test scripts in CMsequence

Two types of CMsequencer sample scripts are provided for throughput testing contained in R&S script packages (see Figure 95).

- XLAPI based throughput scripts package KF600X as CMsequencer script blocks (these blocks cannot be expanded on function block level, script parameterization is possible via configuration file)
- CMsequencer script package KF612, KF614 (scripts can be expanded and parameterized on function block level)

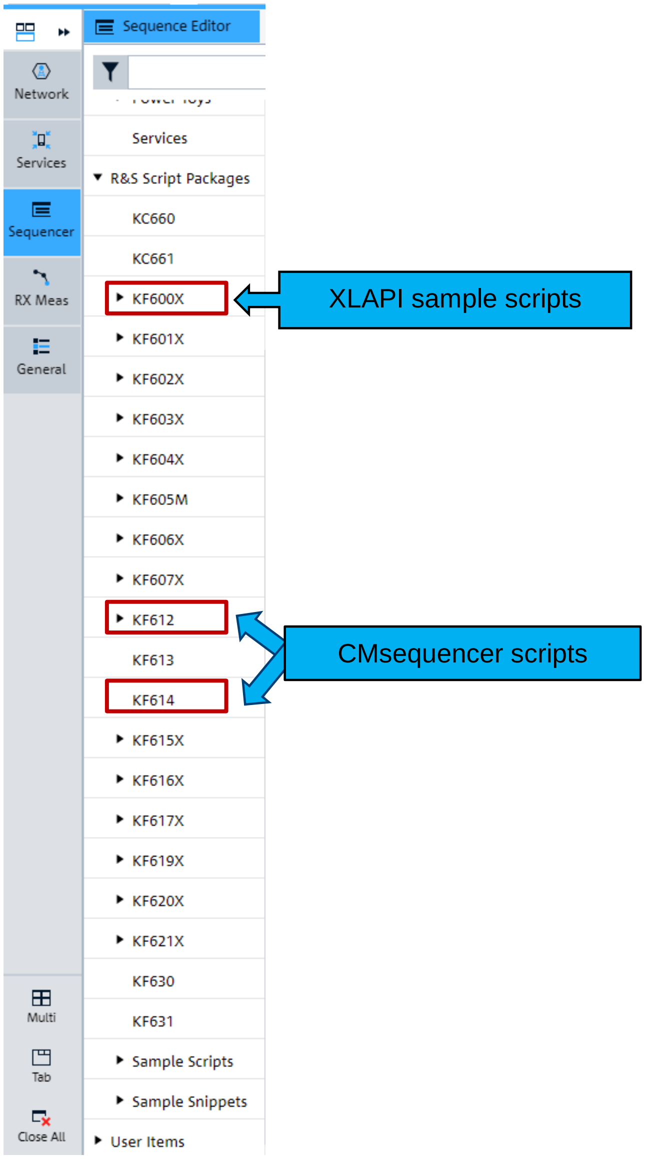


Figure 95 Script packages containing throughput sample scripts

A CMsquencer building block 'Load DUT' should be included at the start of each test script before network emulation starts. As the block name already implies, it loads the active DUT configuration by specifying the active DUT name in its parameter setting. The name should then match the one shown in the DUT configuration square as illustrated in Figure 96 . For more details about how to manage DUT connections, including the activation of a DUT, please refer to 3.2.

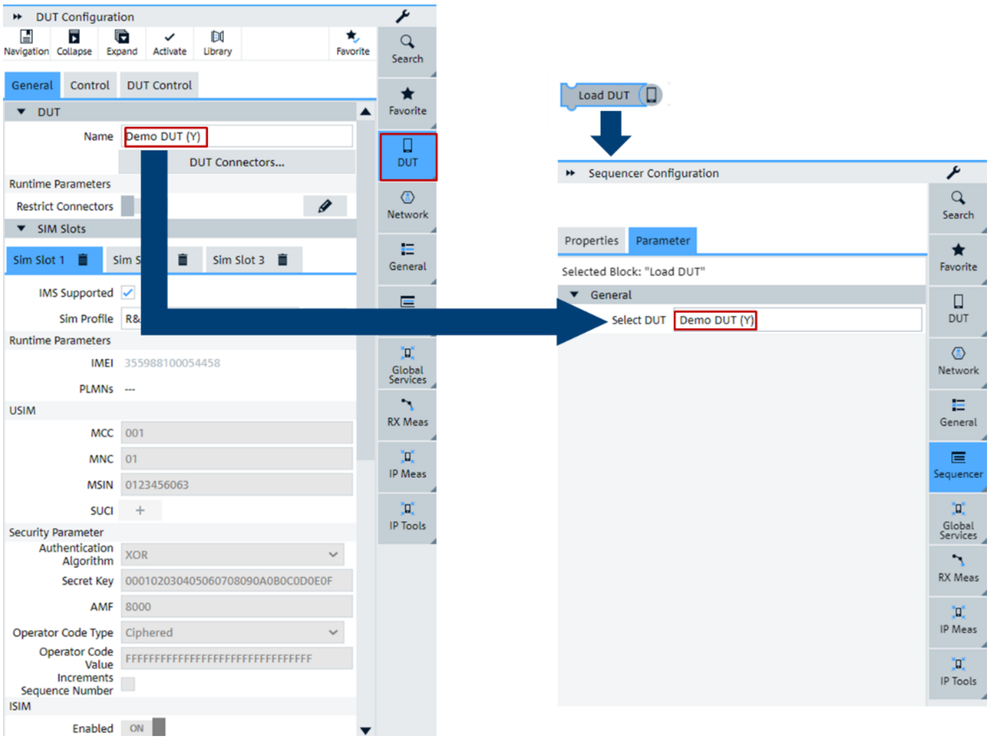


Figure 96 Specifying DUT in 'Load DUT' CMsequencer building block

3.7.1.1 XLAPI Sample Scripts (Script package KF600X)

KF600X package in CMsequencer contains script blocks that are one to one mapped to XLAPI KF600X package containing throughput sample scripts as listed in Table 18 on page 53.

Figure 97 shows the script blocks of KF600X package in CMsequencer.

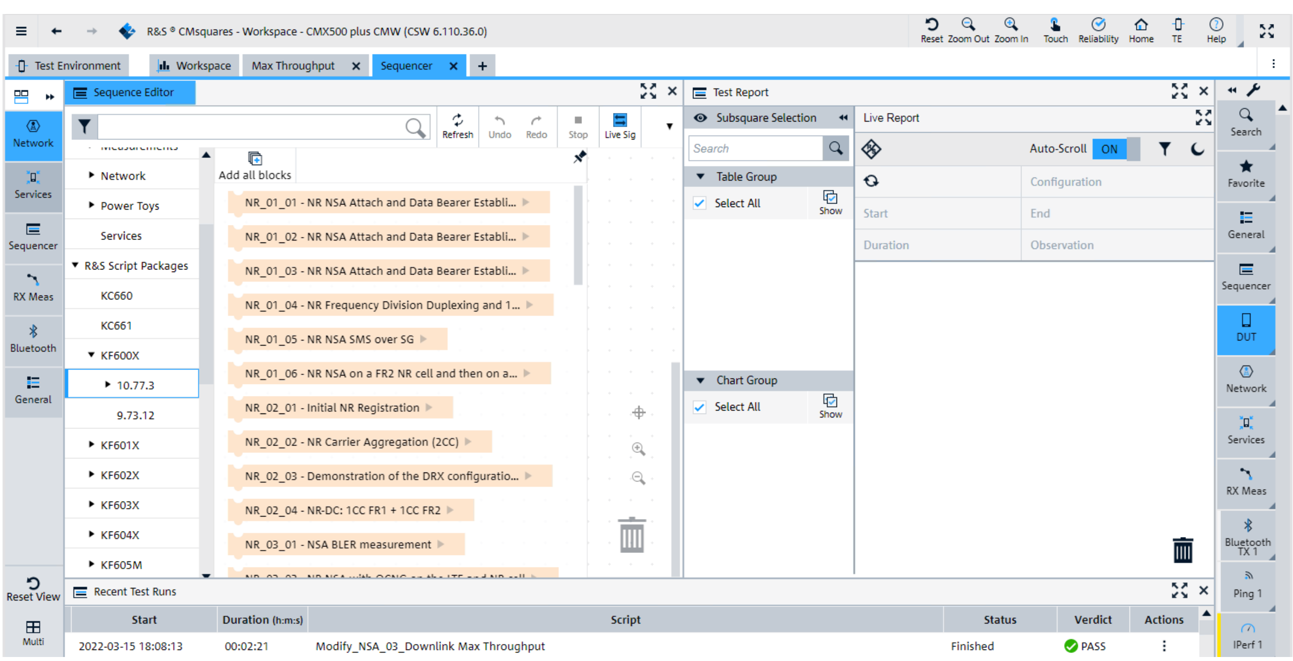


Figure 97 KF600X XLAPI building blocks in CMsequencer

Major benefits of executing XLAPI scripts in CMsequencer are:

► Automated test run

- The scripts can be freely combined with other test scripts or building blocks in the CMsequencer to form a test plan
- Flow control of the XLAPI execution, e.g. if, loop condition etc.

CMsequencer offers two options to parameterize the XLAPI script, by either entering the contents of configuration file `.rsxp` (file format with YAML[22] syntax) in 'input parameters' field directly or parsing the `.rsxp` file including the complete path as execution argument by using `--test-param-files` option in 'execution parameters' field in 'Parameter' page of Sequencer Configuration. An example is given in Figure 98.

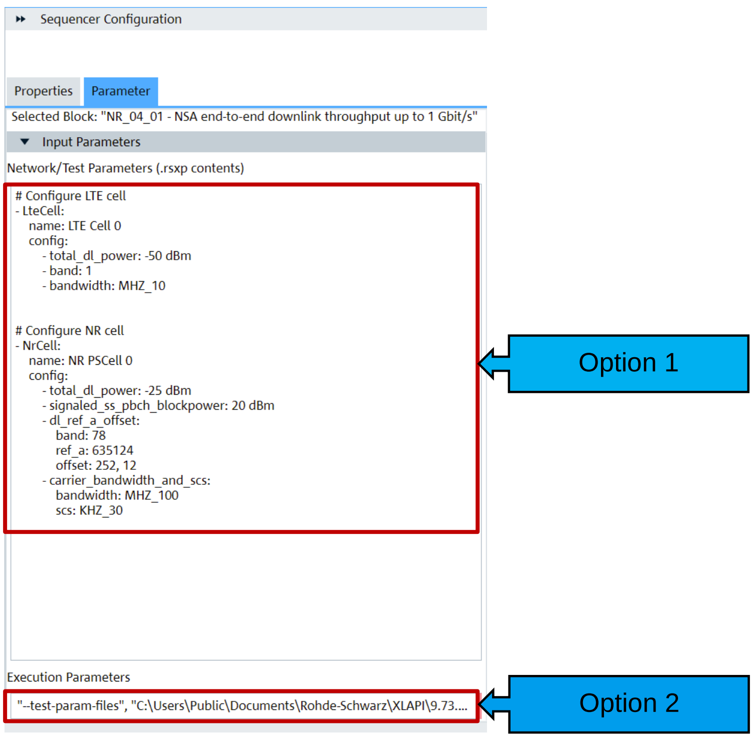


Figure 98 Parameterize XLAPI script in CMsequencer

To get more detailed information about XLAPI script parameterization, read section 'Using Parameter Files' in [15] or visit 'Test Parameters' area in the XLAPI online help document `XLAPI Help` (see also Chapter 3.6.6).

In principle, any XLAPI scripts can be converted into executable CMsequencer blocks via 'Send to CMsequencer' tool in PyCharm. More information about executing a XLAPI script on CMX via CMsequencer, refer to Chapter 4.6 of [15].

3.7.1.2 CMsequencer Sample Scripts (Script package KF612 and KF614)

CMsequencer sample scripts for throughput testing are included in KF612 [17] and KF614 package [18]. These can be served as an entry point to create own test script in CMsequencer environment.

Throughput relevant sample scripts of KF612 package are listed below in Table 21.

Script	Title
NSA_03	End-to-End Downlink Throughput, 700 Mbit/s

[22] YAML stands for 'Yet Another Markup Language'. Like JavaScript Object Notation (JSON), it is a human readable markup language commonly used for configuration files.

Script	Title
SA_09	End-to-End UL Throughput, 350 Mbit/s with UL MIMO
SA_10	End-to-End UL Max Throughput, 950 Mbit/s with UL MIMO
Shuffler_05	ENDC with throughput with band combinations from CSV
Shuffler_06	ENDC with 4x4 Max throughput with band combinations from CSV

Table 21 CMsequencer throughput sample scripts in KF612 package

As a special remark, the scripts with shuffler function are iterated by the permutated band combinations presented in the CSV file (see Figure 99). The CSV configuration file is placed in folder `C:\Users\Public\Documents\Rohde-Schwarz\CMsequencer\conf`. This function relieves the effort to explore the throughput among the various UE supported band combinations.

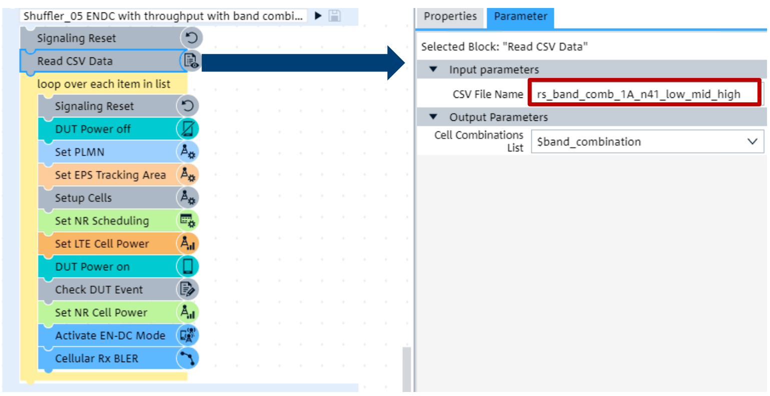


Figure 99 Script with shuffler function and its input from CSV file

Following sample test scripts out of KF614 package given in Table 22 can be used and modified to verify the UE throughput during the handover procedure.

Script	Title
CONN_07	NSA E2E Throughput with Handover in Loop
CONN_08	SA E2E Throughput with Handover in Loop

Table 22 CMsequencer throughput sample scripts in KF614 package

Each sample script can be expanded on function block level that can be parameterized individually. In CMsequencer, right mouse click on the selected script and choose 'Expand Block' option to see the belonging function blocks of the script (see Figure 100).

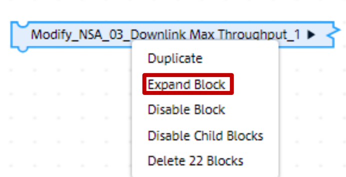
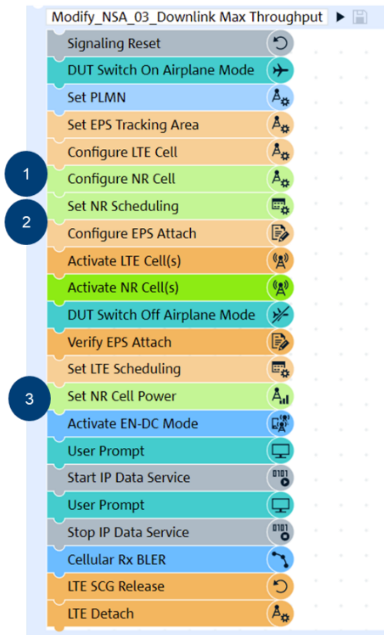


Figure 100 Expand block of CMsequencer script

In Figure 101, it shows as an example a modified script from the sample script 'NSA_03 End to End Downlink throughput 700 Mbps' and major parameter configurations of the selected function blocks for DL maximum throughput testing in TDD mode.

CMsequencer Script



Function Block Configurations

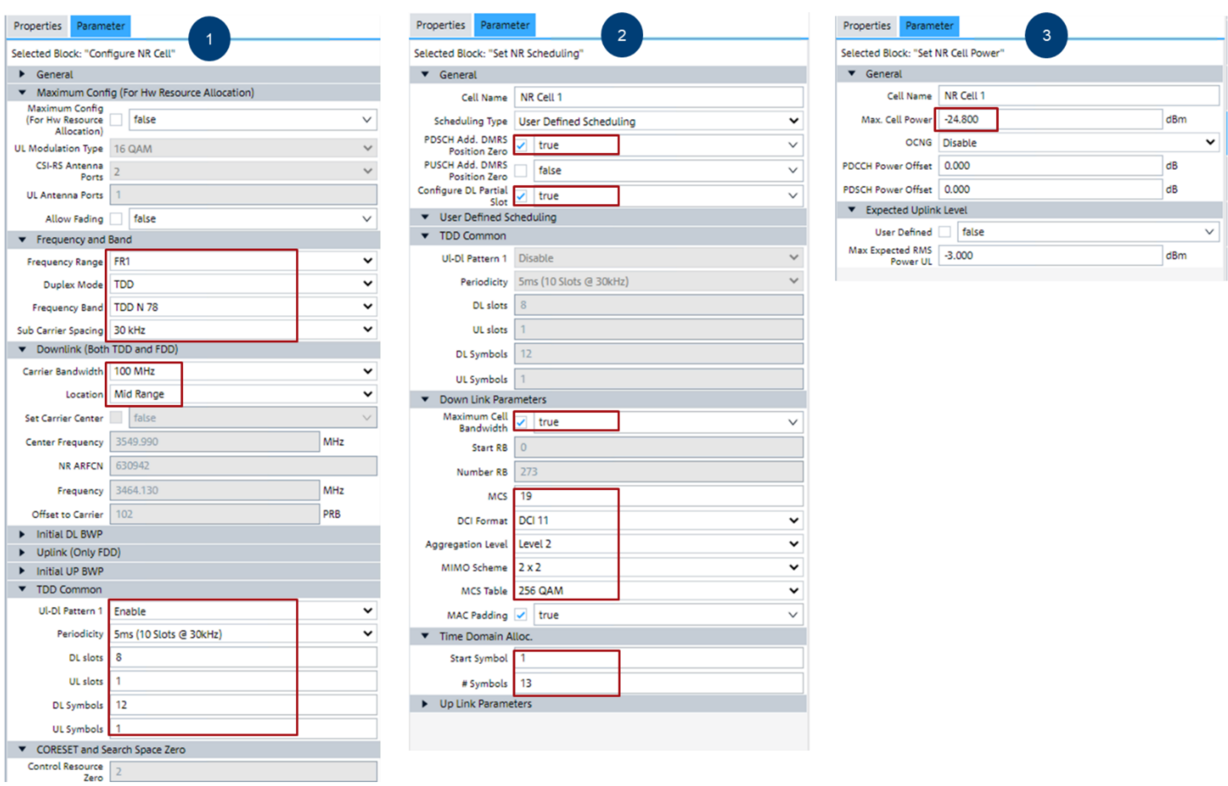


Figure 101 CMsequencer script based on sample script 'NSA_03 End to End Downlink throughput 700 Mbps' and configurations of the selected function blocks

To run the script, press ▶ button next to the script name as indicated in Figure 102.

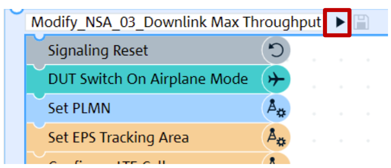


Figure 102 Run CMsequencer script

3.7.2 Reports

As shown in Figure 103, live test report is presented in 'Test Report' square and it will be kept until the new test run reflashes the live report square.

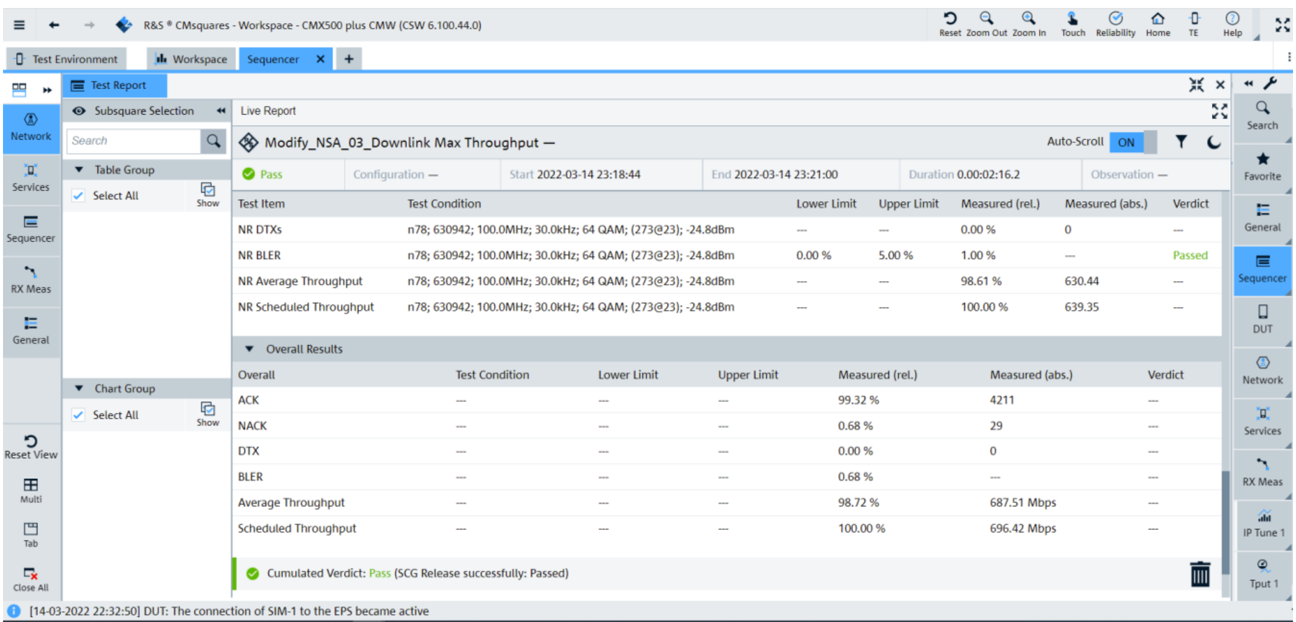


Figure 103 CMsequencer live test report

Test reports and logfiles of the previous test runs can be accessed easily from 'Recent Test Runs' square (see Figure 104). Click on icon ⋮ next to the test script of interest, more options are provided to enable the download of logfiles in different test report formats (.csv / .html / .pdf / .xml).

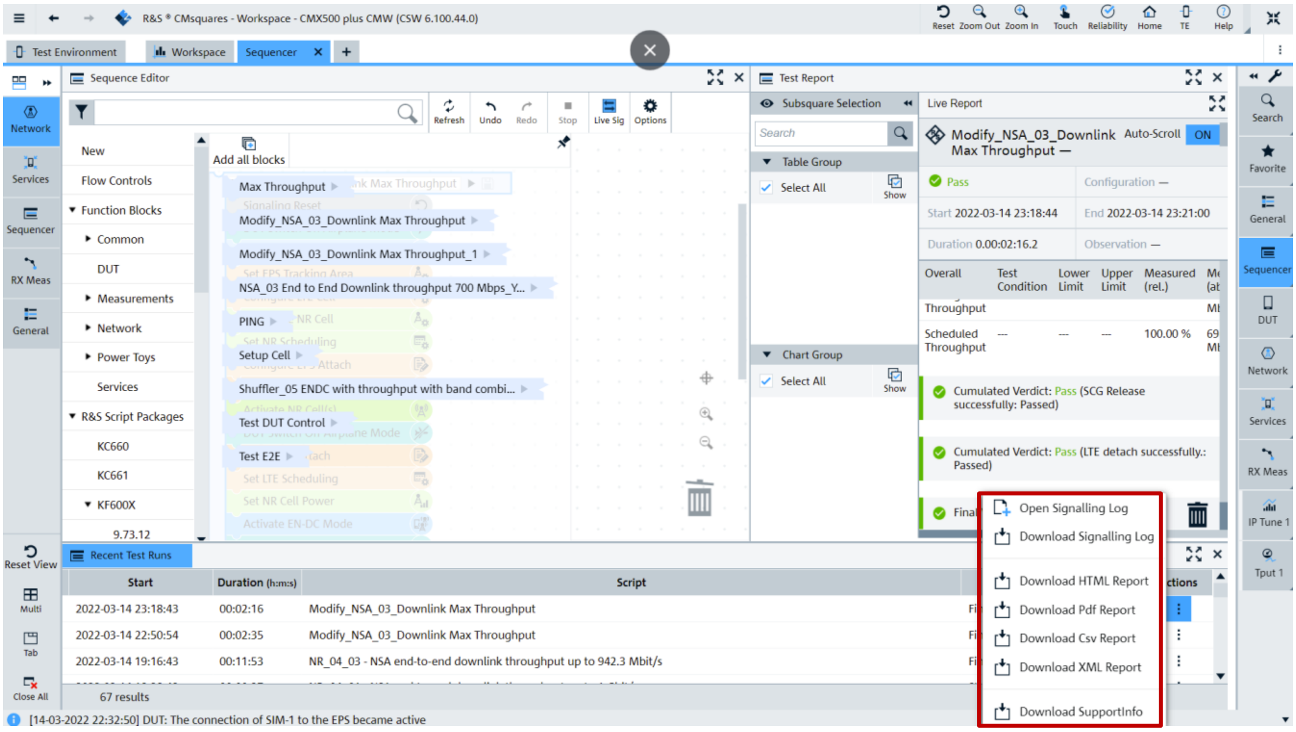


Figure 104 Easy access to logfiles and test reports in 'Recent Test Runs' square

4 Summary

This application note tackles the background information of the optimization policies for verifying UE's DL maximum throughput in TDD mode for NR FR1. It includes the optimal positioning of the SSB/CORESET0, PDSCH/PDCCH/PUCCH/PUSCH optimization in the resource grid, MCS, HARQ scheme, DRB establishment and MIMO etc. It is all about how to increase the transmission bandwidth, reduce the control overhead, assign more resources for the data transmission in the resource grid in DL direction.

In further course of the document, R&S test and measurement (T&M) solutions based on CMX platform are presented. Table 23 summarizes the available throughput measurements in CMsquares and/or applications.

Measurement	Access of the Measurement	Typical Use Case
BLER	CMsquares (Rx Meas)	DL/UL throughput (scheduled and actual measured throughput) on MAC layer.
RTT	CMsquares (Ping)	Test the responsiveness and stability of the data connection between DUT and CMX
iPerf	CMsquares (iPerf) + iPerf on DUT	End-to-end throughput measurement on IP level in both UL and DL direction with both UDP and TCP
Throughput	CMsquares (Tput)	Throughput on different layers (PHY, MAC, RLC, PDCP etc.) in both UL and DL direction
Traffic Generator	Web application hosted on DUT running in DUT's Web browser	Quick check of the DL and UL throughput of the DUT
IP Tune	CMsquares (IP tune) + R&S Throughput App 2 on DUT	Fine tune the throughput on IP level in both UL and DL direction with both UDP and TCP including MCS tuning

Table 23 Services on CMX for throughput measurements

For ease of cell configurations tailored for max throughput testing, a max throughput wizard in CMsquares has been highlighted. Furthermore, guidelines and tips of performing max throughput by adopting XLAPI script which stands for more configuration and control flexibility, as well as the CMsequencer, a sequencer guarantees better test script management, are elaborated.

5 Literature

[1] Rohde & Schwarz, "5G NR ebook," [Online]. Available: https://www.rohde-schwarz.com/5G-ebook.

[2] R&S, "R&S®CMX500 Radio Communication Tester User Manual".

[3] 3GPP, "3GPP TS38.306 V16.4.0 (2021-03); NR User Equipment (UE) radio access capabilities (Release 16)".

[4] 3GPP, "3GPP TS38.101-1 V17.2.0 (2021-06); NR User Equipment (UE) radio transmission and reception Part1: Range 1 Standalone (Release 17)".

[5] 3GPP, "3GPP TS38.101-2 V17.2.0 (2021-06) NR User Equipment (UE) radio transmissiion and reception Part 2: Range 2 Standalone (Release 17)".

[6] 3GPP, "3GPP TS38.211 V16.6.0 (2021-06); NR Physical channels and modulation (Release 16)".

[7] 3GPP TSG RAN WG1 Meeting #92, "Discussion on NR UE peak data rate," in *R1-1801352*.

[8] 3GPP, "3GPP TS38.214 V16.6.0 (2021-06); NR Physical layer procedures for data (Release 16)".

[9] 3GPP TS38.213 V16.2.0 (2020-06), "NR; Physical layer procedures for control (Release 16)".

[10] 3GPP TS38.508-1 V16.4.0 (2020-06), "5GS; User Equipment (UE) conformance specification; Part 1: Common test environment (Release 16)".

[11] R&S, "CMX One-Box-Tester press release," 10 Febuary 2022. [Online]. Available: https://www.rohde-schwarz.com/about/news-press/all-news/rohde-schwarz-presents-the-new-r-s-cmx500-one-box-tester-a-powerful-5g-test-platform-for-simplified-device-testing-press-release-detailpage_229356-1175685.html?change_c=true.

[12] "iPerf User Manuals," [Online]. Available: https://iperf2.sourceforge.io/iperf-manpage.html.

[13] R&S, "R&S®CMX500 Application Tests User Manual".

[14] R&S, "R&S®CMW-KT014 Automation Manager Software Manual".

[15] R&S, "R&S®CMX500 XLAPI Scripting Interface User Manual".

[16] R&S, "R&S®CMsequencer User Manual".

[17] R&S, "R&S®CMX-KF612M CMsequencer Script Package User Manual".

[18] R&S, "R&S®CMX-KF614M CMsequencer Script Package User Manual".

[19] 3GPP, "3GPP TS38.104 V15.11.0 (2020-09); NR Base Station (BS) radio transmission and reception (Release 15)".

[20] R&S, "R&S®CMX-KF600X NR XLAPI Test Scripts Release Notes".

[21] R&S, "R&S®CMX500 Signaling Applications User Manual".

[22] "R&S®CMW500 Base Software User Manual," [Online]. Available: https://gloris.rohde-schwarz.com.

Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, monitoring and network testing. Founded more than 80 years ago, the independent company which is headquartered in Munich, Germany, has an extensive sales and service network with locations in more than 70 countries.

www.rohde-schwarz.com

Certified Quality Management
ISO 9001

Rohde & Schwarz training

www.training.rohde-schwarz.com

Rohde & Schwarz customer support

www.rohde-schwarz.com/support

PAD-T-M: 3572.7186.00/01.00/EN

1SL379 | Version 2e | 08.2022
Application Note | Measuring Data Throughput with The CMX500
Data without tolerance limits is not binding | Subject to change

www.rohde-schwarz.com