|  |  |
| --- | --- |
|  |  |

XTDP Protocol Module Function Specification

[1 General 2](#_Toc283634886)

[1.1 Revision history 2](#_Toc283634887)

[1.2 How to Read this Document 2](#_Toc283634888)

[1.3 Scope 2](#_Toc283634889)

[1.4 Abbreviations 2](#_Toc283634890)

[1.5 Terminology 3](#_Toc283634891)

[1.6 References 3](#_Toc283634892)

[2 Version and conformance 4](#_Toc283634893)

[2.1 Version 4](#_Toc283634894)

[2.2 Conformance 4](#_Toc283634895)

[3 Data structure 4](#_Toc283634896)

[3.1 XTDP 4](#_Toc283634897)

[3.2 XTDL 5](#_Toc283634898)

[3.3 Generating the TTCN-3 modules from the XSD documents 5](#_Toc283634899)

[3.4 Encoding, decoding and length calculation 5](#_Toc283634900)

[4 Procedures 6](#_Toc283634901)

[4.1 General 6](#_Toc283634902)

[4.2 Handshake 6](#_Toc283634903)

[4.3 Handling of unsupported procedures 9](#_Toc283634904)

[4.4 Authentication 10](#_Toc283634905)

[4.5 Layout information 10](#_Toc283634906)

[4.6 Acknowledged action requests 11](#_Toc283634907)

[4.7 Unacknowledged action requests 14](#_Toc283634908)

[4.8 Add requests 14](#_Toc283634909)

[4.9 Remove requests 16](#_Toc283634910)

[4.10 User data 17](#_Toc283634911)

[4.11 Closing the session 18](#_Toc283634912)

[4.12 Underlying connection lost 20](#_Toc283634913)

[4.13 Timer and counter values 20](#_Toc283634914)

[5 Other 21](#_Toc283634915)

[5.1 Supported platforms 21](#_Toc283634916)

[5.2 Limitations 21](#_Toc283634917)

# Abstract

This document specifies the eXtensible Titan Display Protocol (XTDP). XTDP is a simple transfer protocol that can be used for communication between running TTCN-3 test cases and any external program or entity, like a GUI, any controlling or database servers etc. XTDP provides built-in support for the eXtensible Titan Display Language (XTDL), which is used e.g. between running TitanSim applications and the Runtime GUI (CNL 113 437) product of TCC.

# General

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2011-01-21 | PA1 | First version | ELSZSKU |
|  |  |  |  |

## How to Read this Document

This is the Function Specification for the XTDP protocol module, which is developed for the TTCN-3 Toolset with TITAN. This document should be read together with Product Revision Information ‎[1].

## Scope

The purpose of this document is to specify the functionality of the XTDP protocol, especially the dynamic behaviour of the protocol. XTDP is designed for two purposes:

* to support a communication between a running TTCN-3 test case and an external XTDP/XTDL-enabled GUI to present information about test execution on the GUI and to convey commands and data to the running test case from the GUI. As an example, the Runtime GUI product of TCC (CNL 113 437) supports this feature;
* can be used as a transport protocol for any user-specified textual or binary protocols between a running TTCN-3 test case and any external entity (no XTDP is required for the communication between running TTCN-3 test components as this capability is provided by Titan).

To read this document (which is referring to the static data structure specification) requires basic knowledge of XSD. To use the protocol, basic knowledge of TTCN-3 ‎[8], the XML to TTCN-3 mapping ‎[9] and the TITAN TTCN-3 Test Executor ‎[7] is valuable.

## Abbreviations

ETSI European Telecommunications Standards Institute

GUI Graphical User Interface

TCC (Ericsson R&D) Test Competence Center

TTCN-3 Testing and Test Control Notation version 3

XML Extensible Markup Language

XSD XML Schema Definition

XTDL eXtensible Titan Display Language

XTDP eXtensible Titan Display Protocol

## Terminology

Titan Ericsson’s TTCN-3 Test environment

TitanSim Ericsson’s Load Test Framework

Widgets Description of a screen configuration

## References

1. 109 21-CNL 113 663-1 Uen  
   XTDP Protocol Module, Product Revision Information
2. World Wide Web Consortium W3C Recommendation: "Extensible Markup Language (XML) 1.1"  
   NOTE: Available at <http://www.w3.org/TR/xml11>.
3. World Wide Web Consortium W3C Recommendation (2006): "Namespaces in XML 1.0"  
   NOTE: Available at <http://www.w3.org/TR/REC-xml-names/>
4. World Wide Web Consortium W3C Recommendation (2004): "XML Schema Part 0: Primer"  
   NOTE: Available at <http://www.w3.org/TR/xmlschema-0>.
5. World Wide Web Consortium W3C Recommendation (2004): "XML Schema Part 1: Structures"  
   NOTE: Available at <http://www.w3.org/TR/xmlschema-1>.
6. World Wide Web Consortium W3C Recommendation (2004): "XML Schema Part 2: Datatypes"  
   NOTE: Available at <http://www.w3.org/TR/xmlschema-2>.
7. 1/1553-CRL 113 200 Uen  
   User Documentation for the TITAN TTCN-3 Test Executor
8. ETSI ES 201 873-1  
   The Testing and Test Control Notation version 3. Part 1: Core Language
9. ETSI ES 201 873-9  
   ETSI Standard, Methods for Testing and Specification (MTS);The Testing and Test Control Notation version 3; Part 9: Using XML schema with TTCN-3

# Version and conformance

## Version

The protocol described in this document has the versions:

XTDP version: 1.0

XTDL version: 1.0.

## Conformance

It is not mandatory for an implementation to be able to initiate all procedures specified in this document. The designer of an application using XTDP may decide which procedures needs to be supported by the given application (and in cases of some procedures, also further features, see clauses ‎4.4, **Error! Reference source not found.** and **Error! Reference source not found.**). Also, an application designer may limit the allowed directions of the individual procedures, except of handling unsupported procedures, which shall be supported by all entities.

However, all implementations claiming conformance to this version of the protocol shall support at least the client, the server or both roles of the Handshake procedure (see clause ‎4.2) and shall be able to handle all non‑supported procedures initiated by the peer entity according to clauses ‎4.1 and ‎4.3.

Implementations shall declare the following:

# Data structure

## XTDP

XTDP messages shall be valid XML documents (see ‎[1]); the data structure and content are specified by using XSD (see ‎[4], ‎[5], ‎[6]) and is available in TCC’s ClearCase in the file /vobs/ttcn/TCC\_Releases/ProtocolModules/XTDP\_CNL113663/src/XTDP-Message.xsd. The receiving entity shall validate the received messages against this XSD specification.

## XTDL

The data structure of XTDL information shall be valid XML element (see ‎[1]); the data structure and content are specified by using XSD (see ‎[4], ‎[5], ‎[6]) and is available in TCC’s ClearCase, in the file /vobs/ttcn/TCC\_Releases/ProtocolModules/XTDP\_CNL113663/src/XTDL.xsd. The receiving entity shall validate the received information against this XSD specification.

## Generating the TTCN-3 modules from the XSD documents

The TTCN-3 modules, that can be imported to the user’s TTCN-3 code shall be generated from the above XSD files using the xsd2ttcn converter which is part of the TITAN TTCN-3 Test Executor toolset.

## Encoding, decoding and length calculation

This product contains encoding/decoding functions. The functions for encoding TTCN-3 instances of XTDP messages to XML documents and decoding XML documents containing XTDP-Message to TTCN-3 values are provided by TITAN. The same is true for instances of the XTDP Window type. These functions can be used in TTCN-3 by invoking the external function declarations provided in the module XTDP\_EncDecFunctions ( see in the file XTDP\_EncDecFunctions.ttcn). This module also contains the function f\_XTDP\_getMsgLen that may be useful e.g. in configuring the IPL4asp CNL 113 531 testport to assist in the message delineation the function. See description of these functions in the table below.

| **Name** | **Type of formal parameters** | **Type of return value** | **Description** |
| --- | --- | --- | --- |
| enc\_XTDP\_Message | in XTDP\_Message | octetstring | Encode a complete XTDP message; encoded message is returned in the |
| dec\_XTDP\_Message | in octetstring  out XTDP\_Message | integer | Decode a complete XTDP message; decoded message is returned in the out parameter; the return value is 0 if decoding was successful, 1 if it was unsuccessful |
| enc\_XTDL\_Window | in Window | octetstring | Encode the XTDL Window type |
| dec\_XTDL\_Window | in octetstring  out Window | integer | Decode an XTDL Window instance; decoded value is returned in the out parameter; the return value is 0 if decoding was successful, 1 if it was unsuccessful |
| f\_XTDP\_getMsgLen | in octetstring  *inout ro\_integer* | integer | Returns the length of the encoded XTDP message or -1 if the input message was too short (less than 4 octets) |
| f\_XTDP\_getRawMsg | in octetstring | octetstring | Returns the encoded XTDP message extracted from the whole sent message (conteining also the length) |

# Procedures

## General

XTDP connections shall be set-up in a server-client manner, i.e. one of the entities shall passively wait for an incoming connection request, while (the) other entit(ies) shall initiate the connection establishment. XTDP entities can use either connection oriented (e.g. TCP) or connectionless (e.g. UDP) underlying transport services, depending on the application area. The designer of an application using XTDP may also decide to use both connection‑oriented and connectionless transports, but the type of the transport shall not be changed during an XTDP session, i.e. between the *XTDP‑HandshakeRequest* and the *XTDP‑Bye* messages.

Procedures in this document describe the possible types of interactions between the communicating entities. The concrete transactions initiated by one or another entities are the instances of procedures. Once an XTDP session has been set-up by establishing the underlying transport connection and by completing the Handshake procedure, XTDP becomes a simple transaction-oriented protocol, i.e. one or several transactions may be carried out at the same time and the order of the transactions is not specified, except the limitation in clause ‎4.2 Handshake (at the XTDP level, of course, but the order may be determined at the application logic level). Several procedures are allowed to be started both by the client or the server entities; this is identified by the “peer-to-peer” direction in the procedures’ description.

In case of all acknowledged procedures the initiating entity shall include a unique *transactionID* attribute (field) into the sent XTDP message. The *transactionID* value shall be unique on the given transport connection, among the active transactions of the initiating entity; i.e. transactions started by different entities or transactions started on different transport connections may use identical *transactionID*s. The receiving entity shall include the received *transactionID* into the responding XTDP message. If it responds to one received initiating message by several responses, it shall include the *transactionID* of the initiating message into each responding message.

In case of unacknowledged procedures, the initiating entity may include a *transactionID* attribute (field) into the sent XTDP message, but in this case the receiving entity shall silently ignore it.

NOTE: In this way, if an unacknowledged procedure becomes acknowledged in a later version of this protocol, the initiating entity still will be able to interwork with elder implementations if the use case also allows unacknowledged communication.

## Handshake

Direction: client-server

Procedure type: acknowledged

The purpose of the handshake procedure is to assure that the peer entities have compatible capabilities both in terms of supported XTDP messages and procedures and supported XTDL information.

The handshake procedure shall be the first procedure of an XTDP connectivity session. It shall be initiated by the entity initiating the session (i.e. the client entity). In specific applications, if the application designer decides differently, the handshake procedure may be started by any of the entities; however, in this case the application designer shall assure seamless initiation of this procedure.

No other procedure shall be initiated by any of the peers until the handshake procedure has been finished.



Figure **1**: Handshake procedure

The *XTDP‑HandshakeRequest* message shall contain at least the XTDP protocol version minimally required from the receiving entity, in the *xtdpRequiredVersion* element (field). If the initiating entity will send any XTDL information (i.e. will use any other procedures except Handshake and User data transport) during the session, it shall also include, in the *xtdlRequiredVersion* element (field) the XTDL version minimally required from the receiving entity.

NOTE 1: If the initiating entity has no information about the protocol information and procedures that will be used during the session, it shall include its own versions into the above element (fields).

Optionally, the initiating entity may include the highest supported XTDP and XTDL versions (its own version) into *xtdpMyVersion* and *xtdlMyVersion*.

NOTE 2: Though this information is used for logging/debugging purposes only, it is recommended to always send this information, unless bandwidth or other concerns preventing this.

The initiating entity, on sending the *XTDP‑HandshakeRequest* message, shall start timer Thandsh. If the corresponding *XTDP‑Outcome* message is received before Thandsh expiry, it shall cancel timer Thandsh. If the *XTDP‑Outcome* message contains the “*error”* value in the *responseCode* element (field), the entity shall invoke closing the XTDP session (see clause Closing the session) and shall report a version compatibility check failure to the user and/or to a management entity, including also the protocol(s) which failed the check (XTDP or XTDL or both). If Thandsh expires, the initiating entity shall shall invoke closing the XTDP session (see clause Closing the session) and report a communication error result to the user and/or to a management entity.

On receiving an *XTDP‑HandshakeRequest* message, the receiving entity shall carry out an XTDP version compatibility checking: it shall decide if it can support the version identified in *xtdpRequiredVersion* of the received *XTDP‑HandshakeRequest* message. If it can support the version identified by *xtdpRequiredVersion*, the XTDP version check is successful, otherwise it is failed.

NOTE 3: In both XTDP and XTDL version numbers, a minor number change in the protocol version indicates backward compatible extension, while a major version number change indicates non‑backward compatible change.

If the received message contains an *xtdlRequiredVersion* element (field), the receiving entity shall also make an XTDL version compatibility checking as well. In this case, if the receiving entity does not support XTDL or the version of the supported XTDL is lower than the version identified by *xtdlRequiredVersion*, the XTDL version checking fails; if the version of the supported XTDL is higher than the version identified by *xtdlRequiredVersion* and the identified version is acceptable for the receiving entity, the version checking is successful.

NOTE 4: I.e., it shall be decided by the application designer if the fallback to the given lower version is acceptable or not.

The actual communication during the session shall be carried out at the protocol version(s) agreed between the entities in the handshake procedure.

Optionally, if the receiving entity knows apriori that the user of XTDP services will use XTDP protocol data and procedures and XTDL information complying with a lower version of XTDP and/or XTDL respectively, it may use these version information, instead of its own versions, in the above XTDP and XTDL version compatibility checking.

Any other information (i.e. future extensions of the handshake procedure) in the received *XTDP-HandshakeRequest* message shall silently be ignored by the receiving entity.

On completing the version checking(s), the receiving entity shall send an *XTDP-Outcome* message containing elements (fields) as follows:

* If both of the XTDP and XTDL version compatibility checks are successful, the value of *responseCode* shall be set to *“success”* and neither *errorCode*, nor *errorMessage shall be included*;
* If the received message did not contain an *xtdlRequiredVersion* element (field) and the XDTP version checking is successful, the value of *responseCode* shall be set to *“success”* and neither *errorCode, nor errorMessage shall be included.*
* If any or both of the above version compatibility checks fail(s), the value of *responseCode* shall be set to *“error”* and one *errorCode* element (field) shall be included for each failed version check; a failed XTDP version check shall be identified by the *xtdpIncompatibleVersion* value of the *errorCode* element (field) and a failed XTDL version check shall be identified by the *xtdlIncompatibleVersion* value. Including the *errorMessage* element (field) is optional and the receiving entity may include any useful additional information into it (e.g. reporting its own version).

## Handling of unsupported procedures

Direction: both (peer-to-peer)

Procedure type: n/a

This procedure is not designed to handle version incompatibility, i.e. to handle unknown procedure requests, as the handshake procedure assures that the peers are supporting compatible versions of the protocol, before this procedure can be invoked. It allows graceful handling of known, but unsupported procedures and cases, when an application supports only the initiating side of a peer-to-peer procedure.



Figure : Handling of unsupported procedures

When an entity is receiving an XTDP message, which is initiating a procedure, not supported by it, the receiving entity shall send an *XTDP‑Outcome* message in which the value of the *responseCode* is set to *error,* and containing one *errorCode* element (field) with the value *procedureUnsupported* or *directionUnsupported*, whichever is appropriate. Including the *errorMessage* element (field) is optional and the receiving entity may append any further useful information into it.

NOTE: The Unacknowledged action requests**Error! Reference source not found.** and the Acknowledged action requests**Error! Reference source not found.** are different procedures, therefore this clause also apply if the *XTDP‑Requests* message itself is supported but the required acknowledgement form is not.

## Authentication

Direction: server-client

Procedure type: acknowledged

The procedure allows the server side to ask authentication from the connecting client.



Figure : Authentication procedure

The procedure is invoked by the server on sending an *XTDP‑AuthChallenge* message stating a simple string challenge question. On sending the message, the entity shall start timer Tauth. If the corresponding *XTDP‑AuthResponse* message is received before Tauth expiry, it shall cancel timer Tauth. If Tauth expires, the initiating entity shall invoke closing the XTDP session (see clause Closing the session) and report the authentication failure to the user and/or to a management entity.

The receiving entity shall respond with an *XTDP‑AuthResponse* message containing a simple string challenge response.

The challenge question and the challenge response are beyond the scope of this document, the authentication algorithm shall be specified by the application designers.

## Layout information

Direction: both (peer-to-peer)

Procedure type: acknowledged

The procedure allows to an entity to request the sub-tree belonging to a given parent widget.



Figure : Layout change procedure

The initiating entity shall invoke the procedure by sending an *XTDP‑LayoutRequest* message. The message may contain the identifier of the parent widget, included into the *widgetId* element (field). If *widgetId* is missing from the message, the layout information of the whole display is requested.

The initiating entity, on sending the *XTDP‑LayoutRequest* message, shall start timer Tlout. If the corresponding *XTDP‑LayoutResponse* message is received before Tlout expiry, it shall cancel timer Tlout. If Tlout expires, the initiating entity shall invoke closing the XTDP session (see clause Closing the session) and report the error to the user and/or to a management entity.

The receiving entity, on receipt of an *XTDP‑LayoutRequest* message, shall respond with an *XTDP‑LayoutResponse* message, containing the widget-children, i.e., the XTDL tree belonging to the requested parent widget.

## Acknowledged action requests

Direction: both (peer-to-peer)

Procedure type: acknowledged

This procedure is related to manipulating existing elements of one or several widget(s). The initiating entity demands reporting the results of execution for each requested actions.



Figure : Unacknowledged information change procedure

To invoke this procedure the initiating entity shall send an *XTDP‑Requests* message with the *ackRequested* attribute set to the value *true*. Each message shall contain at least one *XTDP‑Request* element (field) but may contain more. Though, the protocol allows bunching several requests into a single *XTDP‑Requests* message, the strategy of grouping the requests is application dependent (i.e. an application designer may decide to send each request in separate messages but may also decide e.g. to collect all requests for a given time period into one message).

Each *XTDP‑Request* element (field) in the message shall contain

* A mandatory *requestId* element (field), which shall contain a value, unique within the given message*;*
* *a mandatory* *action* element (field) that shall define the action requested;

The following *action*s are defined:

* *get* the requestor wants to query the current value or status of a widget; this action shall be used with the Acknowledged action requests procedure only
* *put* the requestor wants to update the current value or status associated to a widget
* *disable* the requestor wants a widget to be disabled, i.e., it is rendered as such (greyed out and/or set to be non-editable
* *enable* the opposite of disable, e.g. the widget is un-greyed and/or set to be editable
* *increase* the requestor wants to increase the value stored in the widget, i.e., it is meaningful only for widgets supporting it (e.g., numerical widget with a spinner
* *decrease* the requestor wants to decrease the value stored in the widget, i.e., it is meaningful only for widgets supporting it (e.g., numerical widget with a spinner
* *getfocus* the requestor asks for the ID of the widget, which is associated with the focus (or has the focus on the GUI); the requested widgetID shall be returned in the response (in the *argument* element(field));
* *setfocus* the requestor wants the associated widget to get focus, i.e., if it is hidden then it shall be raised to be visible, it shall receive keyboard inputs, etc.
* *gettooltip* the requestor asks the tooltip that is associated to the widget with the identified ID; the requested tooltip shall be returned in the response
* *settooltip* the requestor wants indicate associated widget to get a new tooltip value associated
* *replaceimage* the requestor wants to replace the image rendered on the associated widget (such as a button pixmap).

In actions, where the *widgetID* is optional, an omitted *widgetID* identifies the whole display area. Where the *widgetID* is not allowed (no *widgetID* element is included into the XSD definition of the given action), a widget ID is not applicable for the given action.

Please note that this document does not specify, which actions should be supported by the applications, nor the allowed action directions, applicability of actions to the different kind of widgets and arguments. Application developers shall unambiguously define this information in the documentation of the given application.

The initiating entity, on sending the *XTDP‑Requests* message, shall start timers Tactm and Tactp. If all outstanding requests have been acknowledged with any of the result codes (i.e. some requests may be in the *pending* state, see the values of the *responseCode* element (field)) in one or more *XTDP‑Responses* message(s) before **Error! Reference source not found.** expiry, it shall cancel timer **Error! Reference source not found.**. If no *XTDP‑Responses* message has been received or not all outstanding requests has been acknowledged before the expiry of **Error! Reference source not found.**, the entity shall cancel timer Tactp and invoke closing the XTDP session according to clause‎4.11 Closing the session and report the error to the user and/or to a management entity. If all outstanding requests have been acknowledged with the *success* or *error* result before the expiry of Tactp, Tactp shall be cancelled. If not all outstanding requests have been acknowledged with the *success* or *error* result before the expiry of Tactp, the action taken is application dependent.

When the receiving entity receives an *XTDP‑Requests* message with the *ackRequested* attribute set to *true*, it shall send the result of each requested action in one or more *XTDP‑Responses* message. Each result shall be reported in one *XTDP‑Response* element (field) with the following content:

* *requestId* mandatory, shall have the same value as the *requestId* of the action request, the result of which is being reported by the given *XTDP‑Response*
* *responseCode* mandatory, shall be one of the *success,* *pending* or *error* values, depending on the status of executing the given action
* *argument* optional, included if the given action specifies a return value.
* *errorSource* conditional, shall be used, if the value of *responseCode* is *error*; it shall specify the location of the source of the error
* *errorCode* conditional, shall be used, if the value of *responseCode* is *error*; it shall specify the reason of the error
* *errorMessage* optional, may be included to report any further useful information regarding the cause of the error

## Unacknowledged action requests

Direction: both (peer-to-peer)

Procedure type: unacknowledged

This procedure has the same purpose as the Acknowledged action requestsAcknowledged action requestsAcknowledged action requests procedure with the difference, that the initiating entity does not request reporting the results of the requested actions.



Figure : Unacknowledged information change procedure

To invoke the procedure the initiating entity shall send an *XTDP‑Requests* message with the *ackRequested* attribute set to the value *false*. The *XTDP‑Requests* message shall contain information as specified in clause ‎4.6 Acknowledged action requests with one exception: the *get* action shall not be used with the unacknowledged procedure.

## Add requests

Direction: both (peer-to-peer)

Procedure type: acknowledged

This procedure allows to set the widget-children, i.e. the XTDL tree to be associated to a parent widget, dynamically; in one transaction several such additions may be requested.



Figure : The Add procedure

To invoke the procedure the initiating entity shall send an *XTDP‑AddRequests* message. Each *XTDP‑AddRequest* element (field) in the message shall contain

* *requested* mandatory,shall contain a value, unique within the given message
* *parentWidgetId* optional, if present shall identify the parent widget; a missing *parentWidgetId* identifies the whole display area.
* a mandatory widget description (an XTDL full or partial tree)

The initiating entity, on sending the *XTDP‑AddRequests* message, shall start timers Taddm and Taddp. If all outstanding requests have been acknowledged with any of the result codes (see the values of the *responseCode* element (field)) by one or more corresponding *XTDP‑ARResponses* message(s) before Taddm expiry, it shall cancel timer Taddm. If all outstanding requests have been acknowledged with a *success* or *error* result before the expiry of Tactp, timer Tactp shall be cancelled. If no *XTDP‑ARResponses* message has been received or not all outstanding requests has been acknowledged before the expiry of Taddm, , the entity shall invoke closing the XTDP session (see clause Closing the session) and report the error to the user and/or to a management entity. If not all outstanding requests have been acknowledged with the *success* or *error* result before the expiry of Taddp, the action taken is application dependent.

When the receiving entity receives an *XTDP‑AddRequests* message, it shall send the result of each requested addition in one or more *XTDP‑ARResponses* message. Each result shall be reported in one *XTDP‑ARResponse element (field)* with the following content:

* *requestId* mandatory, shall have the same value as the requested addition, the result of which is being reported by the given *XTDP‑ARResponse*
* *responseCode* mandatory, shall be one of the *success,* *pending* or *error* values, depending on the status of executing the given addition
* *errorSource* conditional, shall be used if the value of *responseCode* is *error*; it shall specify the location of the source of the error
* *errorCode* conditional, shall be used if the value of *responseCode* is *error*; it shall specify the reason of the error
* *errorMessage* optional, may be included to report any further useful information regarding the cause of the error

## Remove requests

Direction: both (peer-to-peer)

Procedure type: acknowledged

This procedure allows deleting the widget-children, i.e. the XTDL tree belonging to a parent widget, dynamically; in one transaction several such removal may be requested.



Figure : The Remove procedure

To invoke the procedure the initiating entity shall send an *XTDP‑RemoveRequests* message. Each *XTDP‑RemoveRequest* element (field) in the message shall contain

* *requested* mandatory, shall contain a value, unique within the given message
* w*idgetId* optional, if present shall identify the parent widget; a missing w*idgetId* identifies the whole display area.

The initiating entity, on sending the *XTDP‑RemoveRequests* message, shall start timers Tremm and Tremp. If all outstanding requests have been acknowledged with any of the result codes (see the values of the *responseCode* element (field)) by one or more corresponding *XTDP‑ARResponses* message(s) before Tremm expiry, it shall cancel timer Tremm. If all outstanding requests have been acknowledged with a *success* or *error* result before the expiry of Tremp, timer Tremp shall be cancelled. If no *XTDP‑ARResponses* message has been received or not all outstanding requests has been acknowledged before the expiry of Tremm, the entity shall invoke closing the XTDP session (see clause Closing the session) and report the error to the user and/or to a management entity. If not all outstanding requests have been acknowledged with the *success* or *error* result before the expiry of Tremp, the action taken is application dependent.

When the receiving entity receives an *XTDP‑RemoveRequests* message, it shall send the result of each requested removal in one or more *XTDP‑ARResponses* message. Each result shall be reported in one *XTDP‑ARResponse element (field)* with the following content:

* *requestId* mandatory, shall have the same value as the requested removal, the result of which is being reported by the given *XTDP‑ARResponse*
* *responseCode* mandatory, shall be one of the *success,* *pending* or *error* values, depending on the status of executing the given removal
* *errorSource* conditional, shall be used if the value of *responseCode* is *error*; it shall specify the location of the source of the error
* *errorCode* conditional, shall be used if the value of *responseCode* is *error*; it shall specify the reason of the error
* *errorMessage* optional, may be included to report any further useful information regarding the cause of the error

## User data

Direction: both (peer-to-peer)

Procedure type: unacknowledged

This procedure allows transparent transport any textual or binary information to the peer, not specified by XTDP and XTDL. It allows using XTDP as the transport protocol for any user-defined protocols.



Figure : User data transport procedure

The sending entity shall send the *XTDP-Container* message to use this service. The sending entity may include the *contentType* element (field) into the message to help processing the received message at the receiving side: it may identify the TTCN-3 type of the content as well as e.g. the name and/or the version of the user-defined protocol or the name of the message of the user-defined protocol (e.g. when there are several top level messages) to allow correct decoding at the receiving side etc.

The *XTDP-Container* message shall include either the *textual* element (field), if the encapsulated information is of charstring or universal charstring TTCN-3 types or the *binary* element (field), if it is of bitstring, hexstring or octetstring TTCN-3 types.

Note: the user data transport procedure does not provide acknowledge of receipt or successful processing at the XTDP level. These services, if needed, shall be included into the user-defined protocol.

## Closing the session

Direction: both (peer-to-peer)

Procedure type: acknowledged

This procedure allows closing the XTDP session.



Figure : Closing procedure

To close the XTDP session, the initiating entity shall cancel all running procedures and timers, if any, then send an *XTDP-Bye* message, start timer Tbye and enter the *Pending Closing* state. The message may contain a textual information on the reason of closing. In the *Pending Closing* state the initiating entity shall ignore all other incoming messages then *XTDP-Bye*. If no concurrent closing is initiated by the remote entity, and the responding *XTDP-Bye* message is received, the initiating entity shall cancel timer Tbye and shall initiate closing of the underlying transport connection. If Tbye expires, the entity shall re‑send the *XTDP-Bye* message Nbye-1 times and re-start Tbye at each re‑sending. If no responding *XTDP-Bye* message is received after sending the initiating *XTDP-Bye* message Nbye times (including the one starting the procedure), the entity shall close the underlying transport connection and report a communication failure to the user and/or to a management entity.

NOTE: It is proposed to set Nbye to 1 in case of an acknowledged underlying transport connection (e.g. TCP) and 3 for unacknowledged underlying transport means (e.g. UDP).

The receiving entity, if not in the *Pending Closing* state, on receipt of an *XTDP-Bye* message shall not initiate any further procedure (including session closing), shall stop all running procedures, cancel all running timers and send an empty *XTDP-Bye* message. In case of a connectionless underlying network transport (e.g. UDP) it shall close the transport association after sending the *XTDP-Bye* message (e.g by closing the UDP socket).



Figure : Concurrent closing procedures

In case of concurrent closing procedures, i.e. when the initiating entity in the *Pending Closing* state receives and *XTDP‑Bye* message with a *transactionID* different from the *transactionID* of the *XTDP‑Bye* message sent by it before, shall restart timer Tbye, and respond to the incoming *XTDP‑Bye* message with an empty *XTDP‑Bye* message. However, it shall not close the underlying transport connection before Tbye expires.

NOTE: Pls. note that in the concurrent closing procedure re‑sending of the initiating *XTDP‑Bye* message does not apply.

When Tbye expires:

* if the entity has received an *XTDP‑Bye* message with a *transactionID* identical to the value in the *XTDP‑Bye* message sent by it before, it shall close the underlying transport connection, if yet exists;
* if the entity has not received an *XTDP‑Bye* message corresponding to the *XTDP‑Bye* message sent by it before, it shall close the underlying transport connection, if yet exists, and report a communication failure to the user and/or to a management entity.

When Tbye is running and the underlying connection is closed by the other entity:

* if the entity has received an *XTDP‑Bye* message with a *transactionID* identical to the value in the *XTDP‑Bye* message sent by it before, it shall cancel timer Tbye;
* if the entity has not received an *XTDP‑Bye* message corresponding to the *XTDP‑Bye* message sent by it before, it shall cancel timer Tbye, and report a communication failure to the user and/or to a management entity.

## Underlying connection lost

If the underlaying connection is lost (e.g. a TCP connection is lost or the UDP socket used is closed) without the XTDP session closing procedure (see clause ‎4.11 Closing the session), the entity shall stop all running procedures and timers and report a communication failure to the user and/or to a management entity.

## Timer and counter values

The default values in Table 1 shall be used for the timers and counters of the XTDP protocol. Depending on the application in which the protocol is used, the default values may be changed by the application designer (and, if needed based on an a-priori mutual agreement between the peers).

| **Counter/timer** | **Default value** | **Reference clause** | **Note** |
| --- | --- | --- | --- |
| Thandsh | 1 sec. | ‎4.2 |  |
| Tauth | 1 sec. | ‎4.4 |  |
| Tlout | 1 sec. | ‎4.5 |  |
| **Error! Reference source not found.** | 2 sec. | ‎4.6 |  |
| Tactp | 10 sec. | ‎4.6 | Basically, the timer value is application-dependent. It may also be changed dynamically, based on the number of requests bunched into the *XTDP-Requests* message, hovewer, in this case the algorithm shall be unambiguously known by both sides. |
| Taddm | 2 sec. | ‎4.8 |  |
| Taddp | 10 sec. | ‎4.8 | Basically, the timer value is application-dependent. |
| Tremm | 2 sec. | ‎4.9 |  |
| Tremp | 10 sec. | ‎4.9 | Basically, the timer value is application-dependent. |
| Tbye | 1 sec. | ‎4.11 |  |
| Nbye | 1..3 | ‎4.11 | It is proposed to set Nbye to 1 in case of an acknowledged underlying transport connection (e.g. TCP) and 3 for unacknowledged underlying transport means (e.g. UDP). |

Table 1: Counter and timer default values

# Other

## Supported platforms

Since the used functions are provided by the TITAN the protocol module supports all the platforms supported by the TITAN.

## Limitations

The protocol module requires TITAN version R8A or later. When used in conjunction with the Runtime GUI CNL 113 437, the RT GUI shall be of version R9A or later. Compatibility with other products is not guaranteed.