Project **CCN/CSI**

Title **Data Structure Description**

Edition **05**

Date **17 August, 1998**

PROJECT IDENTIFICATION		
CONTRACT NUMBER	PROGRAM	WORK PACKAGE
XXI/95/C003	CCN/CSI lot2	WP2
CUSTOMER	CONTRACTUAL	PACKAGE
EC DGXXI/A1	Yes	LWP2.02

	Name, Function	Date	Signature
Written by :	Olivier ROTH Jean-Luc CHARDON Hervé de HALLEUX Franck ROULLEAUX Nicolas A. SPENGOS		
Checked by :	Alain SMIA Development Manager José GUETTA Quality Manager		
Approved by :	Jean-Daniel RUFF Project Manager		
Authorised by :			

SUMMARY :

This document describes the Data Structures associated with scenarios.

KEYWORDS :

N/A

DOCUMENT CHARACTERISTICS			
Number of pages	Copy reference	Dependent documents	Host system
87	N/A	None	PC
Number of figures	Recipient name		Software
N/A	N/A		Word for windows 95 Version 7.0
Archive number	Recipient function		
N/A	N/A		

DOCUMENT CHANGE LOG

Issue	Issue Date	Modified paragraphs	Modifications
00-d1	16-Dec-1996	All	Creation
00	24-Jan-1997	All	Quality Check
01	not issued		Corrections related to QAM018L2
02	23-Dec-1997	6.5	Modifications related to CR21
		All	General upgrade to meet development compatibility
03	16-Mar-98	All	Corrections related to QAM043L2.
04-d1	31-Mar-98	§5.2, §6.6, §6.7	Modifications related to CR26 and CR27.
04-d2	16-Apr-98	All	Modifications related to CR30.
04	22-Apr-98	All	Technical and Quality Review for project Phase 2.
05	17 August, 98	None	Taking into account QAM65

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PURPOSE OF THE DOCUMENT

This document is part of the Component Functional Specifications. It describes data involved in each of the CCN/CSI APIs (Application Programming Interfaces).

Chapter 5 contains the description of the Common Items (see §1.2 below), and of the sub-structures of common use to the Data Structures.

Data Structures description is split into paragraphs, one for each API:

- *CT API (§ 6.1)*

  Data Structures applicable to the Communication Dependent Transmission (CT) API in the Communication layer adaptation modules;
- *GT API (§ 6.2)*

  Data Structures applicable to the Generic Transmission (GT) API in the GT layer core;
- *CSI API (§ 6.3)*

  Data Structures applicable to the Common System Interface (CSI) API in the Low Level Function layer core;
- *SPI API (§ 6.4)*

  Data Structures applicable to the Service Provider Interface (SPI) API in the SPI Function layer core;
- *PRES API (§ 6.5)*

  Data Structures applicable to the Presentation (PRES) API in the Data Presentation modules;
- *GSS API (§ 6.6)*

  Data Structures applicable to the Generic Security Service (GSS) API in the Security modules;
- *HL API (§ 6.7)*

  Data Structures applicable to the High Level (HL) API in the High Level Function layer core;

This document should be used in parallel with the [RD1] document, according to the following rule: for each Data Flow in [RD1], there is an attached Data Structure in the present document.

The present document will also serve as a working base for the creation of the CCN/CSI Reference Manuals.

1.2. FIELD OF APPLICATION

This document applies to Phase 1 and Phase 2 of the CCN/CSI system implementation.

2. DOCUMENTS

2.1. APPLICABLE DOCUMENTS

[AD1]: CCN/CSI Requirement Specifications.
Reference: CCN/CSI-FRS-GEN-01-MARB

[AD2]: CCN/CSI Specification Rules.
Reference: CCN/CSI-USG-SRL-00-MARB.

[AD3]: CCN/CSI: System Overview.
Reference: CCN/CSI-OVW-GEN-01-MARB.

[AD4]: CCN/CSI: Functional System Specifications.
Reference: CCN/CSI-FSS-GEN-01-MARB.

[AD5]: CCN/CSI: Architecture Design.
Reference: CCN/CSI-AD-GEN-01-MARB.

2.2. REFERENCE DOCUMENTS

[RD1]: CCN/CSI: Data Flow Diagrams of Scenarios.
Reference: CCN/CSI-CFS-SCEN-01-MARB.

3. TERMINOLOGY

Please refer to the System Overview document [AD3].

4. HOW TO READ THIS DOCUMENT

Data are described using Common Items or Data Structures:

1. *Common Items* are basic types information, that cannot be separated into lower level description. These Common Items can be:
    - *transient data* (not corresponding to a data store in X.500), present in various APIs (flags, return codes, etc). Such data are named **T*xxx**, where **xxx** can be a generic basic type, such as "ulong" or "char[32]".
    - *data corresponding to an existing information in X.500*; such data are named **X*yyy**, where **X*** is a SilverRun prefix, but **yyy** is a name that corresponds exactly to an information described in the X.500 Directory Schema (and always prefixed with "ccn" due to X.500 constraints). ***Note:*** *X.500-related structures are not described in the present document.*
2. *Data Structures* are upper level data descriptions, made of other Data Structures or Common Items. They are prefixed with D* and contain both upper and lower case characters. When referring to a Data Structure, SilverRun expands its content until reaching the Common Item level.
3. *Common Data Structures* are used as sub-elements for Data Structures; they are not associated to one particular API. Common Data Structures are prefixed with D* and are made up of upper case characters only.

For each API verb, the following sub-structures are presented :

- input parameters
- input/output parameters
- output parameters

Each Data Structure or Common Item is presented as a sequence of elements; these elements can be atomic or composite (that is, themselves a sequence of sub-elements). Appropriate indentation of elements and sub-elements is used to depict the nature of each item.

At the end of this document, you will find appended a Table of Indexes, used to facilitate searching for and retrieving Data Structures. Indexing is done on Data Structure, Common Data Structure and Common Item names.

5. COMMON ELEMENTS

5.1. COMMON ITEMS

Name : T*code_page

Comment : This item defines the code page applied to, or requested for, a typed message, request or response.

It is typed as T*long and takes its values in the set that defines the available code pages within the CCN.

Name : T*gss_cred_id_t

Comment : A credential handle is a caller-opaque atomic datum that identifies a GSS API credential data structure.

Credentials describe a principal (identifier), and they give their holder the ability to act as that principal (including secret keys).

Name : T*gss_ctx_id_t

Comment : The T*gss_ctx_id_t data type contains a caller-opaque atomic value that identifies one end of a GSS API security context.

The security context holds state information about each end of a peer communication, including ciphering state information.

Name : T*hbind

Comment : This item contains the identifier of a bind between an entity (Application, proxy, CSI layer, ...) and an underlying entity givin access to a particular service through an API.

It is typed as T*long.

Name : T*hcall

Comment : This item contains the identifier of a call request issued in a non blocking mode (i.e through the _acall verb of the CSI and HL APIs).

It is typed as T*long.

Name : T*hconn

Comment : This item contains the identifier of a connection established between two peer entities.

It is typed as T*long.

Name : T*host_form

Comment : This item defines the host format applied to, or requested for, a typed message, request or response.

It is typed as T*char[128] and takes its values in the set that defines the available host formats within the CCN.

Name : T*MQHCONN

Comment : The MQHCONN data type represents an MQSeries connection handle, that is, the connection to a particular queue manager.

This item is typed as a T*long.

Name : T*MQHOBJ

Comment : The MQHOBJ data type represents an MQSeries object handle that gives access to an object.

Name : T*msgtype_id

Comment : This item contains the message type identifier to attach to, or retrieved from, a typed message, request or response.

It is typed as T*char[32].

Name : T*prog_ref

Comment : T*prog_ref is a parameter type which is used transmit the reference to a function (function pointer) or to a service (program name) between two entities.

In C language, T*prog_ref is a pointer to a function.

In COBOL language, T*prog_ref is the name of a service program which should be 32 characters or less.

Name : T*reason_code

Comment : This item is used to provide the caller of a verb with a secondary status information. In case of failure, it indicates the reason or the origin of the problem.

In case of normal completion, it is usually meaningless.

T*reason_code is typed as T*long and takes its values in a range according to the entity (HL, SPI, PRES, ...) or third-party software product (Tuxedo, MQ Series,...) where the error occured.

Name : T*return_code

Comment : This item is used to indicate whether the call to a verb has completed successfully, partially or has failed.

It is typed as T*long and can take one of the following values:

+ CSISUCCESS,

+ CSIWARNING,

+CSIFAILED.

Name : T*service_return

Comment : This item contains an Application specific value that is attached to the response to

a service request.

It is typed as T*long.

5.2. COMMON DATA STRUCTURES

Name : D*CSISECINFO

Composition :

authentication provider id	T*long
security information	T*char []

Comment : This structure is a generic structure used to exchange security mechanism dependent parameters that are unknown to the HL API. It is composed of an authentication mechanism identifier (GSS provider identifier) followed by the dependent part represented as a "flat" memory area.

Using this data structure in HL API and GSS API verbs allows to implement a CSI stack that is independent from the security provider that the GSS API uses.

Name : D*DATA_DESCRIPTOR

Composition :

structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

Comment : This structure is used to pass a message either by memory, or by file.

The 'flags' field supports multiple values indicating:

- how the information is given (by memory or by file),

- if the data must persist after processing or not.

According to the data storage mode chosen, only one of the following two fields is significant:

- 'file name' which specifies the name of the file containing the data;

- 'data buffer address' which provides the address of the memory area containing the data.

The 'data length' field specifies the length of the data (in the memory area or in the file, according to the storage mode).

If data is stored by memory, the reference to this memory area is presented differently according to the language being used:

- with the C language, this Data Descriptor contains a pointer to the data stored in memory;

- with the COBOL language, the data to be sent is concatenated to its associated 'Data Descriptor' structure when it is given as parameter of a HL, CSI, PRES, GSS, SPI or GT call.

Name : D*GSS_BUFFER_T

Composition :

length	T*long
value	T*octet [ ]

Comment : Many of the GSS API routines take arguments and return values that describe contiguous multi-byte data. All such data is passed between the GSS API and the caller using the D*GSS_BUFFER_T data type.

This data structure is also used to pass opaque data types, such as security tokens, between the GSS API and the application.

Name : D*GSS_CHAN_BINDINGS

Composition :
opaque structure

Comment : This Data Structure is defined to conform to the X/open C441 specification.

The CCN/CSI implementation of the GSS API doesn't support the use of user-specified tags to identify a given context to the peer Application. These tags, called Channel Bindings, are usually used to identify the particular communication channel that carries the context.

When this data structure appears as input data, the value GSS_C_NO_CHANNEL_BINDINGS shall be used in order to indicate that channel bindings are unused.

Name : D*GSS_NAME_T

Composition :

user_id	X*ccnUserName
application_id	X*ccnApplicationName
user_password	X*ccnUserPassword
application_key	X*ccnApplicationSecurityKey

Comment : The D*GSS_NAME_T Data Structure is used to identify an entity.

It contains the user name and the application identifier.

Optionally, this structure also contains the user password and the application key, when it is used to initalise the credential on the initiator side.

Name : D*GSS_OID

Composition :

length	T*ulong
elements	T*octet [ ]

Comment : This Data Structure is a type containing ISO-defined tree-structured values and is used by the GSS API caller to select an underlying security mechanism.

If only one mechanism is available, the caller may replace the GSS_OID by the value GSS_C_NULL_OID.

Name : D*GSS_OID_SET

Composition :

count	T*int
elements	D*GSS_OID
length	T*ulong
elements	T*octet [ ]

Comment : Certain GSS API procedures take arguments of the type gss_OID_set. This type represents one or more object identifiers.

Name : D*MQGMO

Composition :

structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

Comment : The MQGMO structure contains options for get-message operations.

It is used as an input/output parameter for the _mqget, _mq_browse, and _mq_delete calls of the CSI and HL APIs.

All the fields of the MQGMO structure are described in the MQSeries Technical Reference.

Name : D*MQMD

Composition :

structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long

format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]

Comment : The MQMD structure contains the control information that accompanies the application data when a message travels between the sending and receiving applications.

All the fields of the MQMD structure are described in the MQSeries Technical Reference.

Name : D*MQOD

Composition :

structure identifier	T*char [4]
structure version number	T*long
object type	T*long
object name	T*char [48]
object queue manager name	T*char [48]
dynamic queue name	T*char [48]
alternate user identifier	T*char [12]

Comment : The MQOD structure is used to specify an object by name. The following types of object are valid:

+ queue,

+ process definition,

+ queue manager.

This structure is an input/output parameter for the _mq_open and _mq_put1 calls of the CSI and HL APIs.

All the fields of the MQOD structure are described in the MQSeries Technical Reference.

Name : D*MQPMO

Composition :

Structure identifier	T*char [4]

Structure version number	T*long
Options	T*long
Timeout	T*long
Context	T*MQHOBJ
KnownDescCount	T*long
UnknownDescCount	T*long
resolved queue name	T*char [48]
resolved queue manager	T*char [48]

Comment : The MQPMO structure contains options for put-message operations.

It is used as an input/output parameter for the _mq_put and _mq_put1 calls of the CSI and HL APIs.

All the fields of the MQPMO structure are described in the MQSeries Technical Reference.

Name : D*QOS

Composition :

structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

Comment : The QOS structure contains the items defining the Quality of Service that applies to, or is specified for, a message, a service request or a service response.

Name : D*SERVICE_INFO

Composition :

structure identifier	T*char [4]
version number	T*long
service name	T*char [48]
service data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong

file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
requested quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long
conversation connect descriptor	T*hconn
application key	T*long
client identifier	T*long

Comment : The SERVICE_INFO structure contains the items and sub-structures that describe a synchronous service request.

6. API INPUT AND OUTPUT DATA STRUCTURES

6.1. CT API

Name : D*ct_accept

Composition :

ct_accept input data

CT bind handle	T*hbind

ct_accept output data

CT connect handle	T*hconn
CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_accept verb.

This call waits for and accepts a pending incoming connection request. The caller gets back a new CT connection handle.

Name : D*ct_bind

Composition :

ct_bind input data

bind information	T*char [256]

ct_bind output data

CT bind handle	T*hbind
CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_bind verb.

This call creates a communication channel according to the bind information provided by the caller i.e the activation mode, the protocol used and the network address. The caller receives back a unique CT bind handle that shall be used for the subsequent CT_connect or CT_accept call.

Name : D*ct_connect

Composition :

ct_connect input data

CT bind handle	T*hbind

ct_connect output data

CT connection handle	T*hconn
CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_connect verb.

This call establishes a connection with a server entity. Characteristics of the connection are retrieved from the CT profile table through the CT bind handle received; only "initiator" activation mode is valid. The caller

receives back a unique CT connection handle that must be used for all subsequent CT_send and CT_recv calls to and from the same remote peer.

Name : D*ct_disconnect

Composition :

ct_disconnect input data

CT connection handle	T*hconn

ct_disconnect output data

CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_disconnect verb.

The caller requests a disconnection from the remote peer, on the communication channel referenced by the CT connection handle. It is the responsibility of the underlying Communication Services to ensure that no pending data is lost during disconnection. Successful completion of this call must include the removal of the corresponding C layer connection handle from the associated entry in the CT profile table.

Name : D*ct_recv

Composition :

ct_recv input data

CT connection handle	T*hconn

ct_recv input/output data

data length	T*ulong

ct_recv output data

data buffer	T*octet [ ]
CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_recv verb.

This call waits for and retrieves data sent by the remote peer over the communication channel identified by the received CT connection handle. It is the responsibility of the receiver-side GT layer to repeat calls to CT_recv if the received data contains an indication that the sender-side GT layer has segmented data.

Name : D*ct_send

Composition :

ct_send input data

CT connection handle	T*hconn
data length	T*ulong
data buffer	T*octet [ ]

ct_send output data

CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_send verb.

This call uses the underlying Communication Services to send data to the remote peer, through the channel identified by the CT connection handle. It is the responsibility of the caller to repeat CT_send calls when total data length is greater than the maximum size authorised for the CT data buffers..

Name : D*ct_unbind

Composition :

ct_unbind input data

CT bind handle	T*hbind

ct_unbind output data

CT return code	T*return_code
CT reason code	T*reason_code

Comment : This structure describes the data associated with the CT_unbind verb.

This call releases the communication channel identified by the CT bind handle. Note that this call must be preceded by a CT_disconnect call. Successful completion of CT_unbind includes the removal of the CT bind handle and the corresponding C bind handle from the associated entry in the CT profile table.

6.2. GT API

Name : D*t_bind

Composition :

t_bind input data

application name	X*ccnApplicationName
proxy name	X*ccnApplicationName

t_bind output data

T bind handle	T*hbind
T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_bind verb.

This call binds the caller CSI or SPI layer to the underlying Generic Transmission layer, and establishes a connection to a remote peer either as an initiator or as an acceptor. It conveys a proxy name and an application name that shall be used within the underlying Communication Dependent Transmission layer to extract all the profile information needed to establish the actual connection. The caller gets back a bind handle that has to be used for all subsequent operations to and from that communication channel.

Name : D*t_exec

Composition :

t_exec input data

T bind handle	T*hbind
verb to execute	T*long
data in buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

t_exec output data

data out buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_exec verb.

This call is used to perform a synchronous (Tuxedo) or asynchronous (MQSeries) API call identified by 'verb to execute', using the T bind handle obtained by a previous T_bind call.

Name : D*t_getmsgin

Composition :

t_getmsgin input data

T bind handle	T*hbind

t_getmsgin input/output data

message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

t_getmsgin output data

T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_getmsgin verb.

This call waits for and retrieves a message (either in buffer or file format) from a remote peer through the National Network. The originator channel is identified through the T bind handle. If needed, data reassembly has been performed by the GT layer when the result is returned to the caller.

Name : D*t_putmsgout

Composition :

t_putmsgout input data

T bind handle	T*hbind
message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

t_putmsgout output data

T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_putmsgout verb.

This call sends a message (either in buffer or file format) to a remote peer through the National Network. The recipient channel is identified through the T bind handle. If required, data segmentation is performed by the GT layer.

Name : D*t_rexec

Composition :

t_rexec input data

T bind handle	T*hbind
verb to execute	T*long
data in buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

t_rexec output data

data out buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
receive/send flags	T*long
T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_rexec verb.

This call is used to perform the remote execution of a synchronous (Tuxedo) or asynchronous (MQSeries) API call identified by 'verb to execute'. This call is solely used by Applications on the Application Platform and is transmitted, through the National Network, to the associated Gateway for actual processing.

The 'receive/send flags' parameter determines the behaviour of the GT layer for the correct execution of the call:

- send the request to the peer entity;
- send the request and wait for the result;
- send nothing and wait for incoming data.

Name : D*t_unbind

Composition :

t_unbind input data

T bind handle	T*hbind

t_unbind output data

T return code	T*return_code
T reason code	T*reason_code

Comment : This structure describes the data associated with the T_unbind verb.

This call requests an unbind from the communication channel identified by a T bind handle, or an unbind from a Tuxedo session (on the Gateway). No further message transfer operations are forthcoming on this channel.

6.3. CSI API

Name : D*csi_acall

Composition :

csi_acall input data

CSI service name	X*ccnServiceName
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
flags	T*long
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long

ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

csi_acall output data

request/response call descriptor	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_acall verb.

This call allows a service request to be sent without waiting for the service response (see getrply).

Input data:

+ 'CSI service name': This is the name of the service to be called.

+ 'message to send': The message contains the parameters of the service request. It may be passed by memory or by file. It is the result of a PRES_encode call.

+ 'flags': Same as the 'flags' parameter of the 'tpcall' verb in the XATMI specification. For instance, it is possible to indicate that a reply is not expected.

+ 'quality of service': The quality of service indicates information such as:

- compression, integrity and confidentiality level;

- switch to degraded mode;

- VAS script activation;

- class of traffic.

Output data:

+ 'request/response call descriptor': allows the service response associated with this service request to be retrieved latter (see D*csi_getrply).

+ 'CSI return code': indicates whether the operation is successful or not.

+ 'CSI reason code': indicates the reason of the failure.

Name : D*csi_advertise

Composition :

csi_advertise input data

CSI service name	X*ccnServiceName
function address/prog_name	T*prog_ref

csi_advertise output data

CSI return code	T*return_code

CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_advertise verb.

This call allows a server to advertise the CSI services that it offers.

Refer to the tpadvertise in the XATMI specification for the parameters description.

Name : D*csi_bind

Composition :

csi_bind input data

application name	X*ccnApplicationName
proxy name	X*ccnApplicationName

csi_bind output data

default QOS	X*ccnDefaultQualityOfService
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_bind verb.

The 'application name' and the 'proxy name' parameters are transmitted to the CSI stack.

The default Quality of Service, which is associated with the Application service, is returned when the CSI_bind succeeds.

Name : D*csi_call

Composition :

csi_call input data

CSI service name	X*ccnServiceName
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
flags	T*long

csi_call input/output data

quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]

integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

csi_call output data

service response	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
service return code	T*service_return
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_call verb, which allows an application to send a service request and await its reply.

At the CSI interface, the data is passed through a 'Data Descriptor' structure, which allows to send and receive data either by memory or by file.

As input parameter, 'quality of service' indicates the requested Quality of Service, and as output parameter, what has been effectively applied.

The possible values for the 'flags' parameter are mapped onto the 'flags' parameter of the tpcall verb in the XATMI specification. For instance, it is possible to indicated that the call is indefinitely blocking or not.

Name : D*csi_cancel

Composition :

csi_cancel input data

request/response call descriptor	T*hcall

csi_cancel output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_cancel verb.

This call allows an application to cancel a call descriptor for an outstanding reply corresponding to a request previously issued through a CSI_acall verb.

Name : D*csi_connect

Composition :

csi_connect input data

service name	X*ccnServiceName

application data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long

csi_connect output data

conversation connect descriptor	T*hconn
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_connect verb.

This call allows an Application to set up an half-duplex connection with a conversational CSI service.

The 'application data' is an optional parameter that the Application can use to pass information to the service during the connection establishment. The 'quality of service' parameter describes the particular handling to apply on this data.

The possible values for the 'flags' parameter are mapped onto the 'flags' parameter of the tpconnect verb in the XATMI specification. For instance, it is possible to indicate that the caller uses the connection only for sending, receiving or both.

Name : D*csi_delete_sec_context

Composition :

csi_delete_sec_context input data

token of the context to release	D*GSS_BUFFER_T
length	T*long

value T*octet [ ]

csi_delete_sec_context output data

CSI error code T*return_code

CSI reason code T*reason_code

Comment : This structure describes the data associated with the CSI_delete_sec_context verb.

The token associated with the context to release is given as input parameter.

Name : D*csi_discon

Composition :

csi_discon input data

conversation connect descriptor T*hconn

csi_discon output data

CSI return code T*return_code

CSI reason code T*reason_code

Comment : This structure describes the data associated with the CSI_discon verb.

This call allows an Application to immediately abort an existing connection with a conversational CSI service. This routine is only available for the initiator of the conversation.

Name : D*csi_getrply

Composition :

csi_getrply input data

request/response call descriptor T*hcall

request/response flags T*long

csi_getrply output data

service response D*DATA_DESCRIPTOR

- structure identifier T*char [4]
- structure version number T*long
- flags T*ulong
- file name T*char [256]
- data length T*ulong
- data buffer address T*octet [ ]

quality of service D*QOS

- structure identifier T*char [4]
- version number T*long
- specified QOS T*long
- applied QOS T*long
- urgency T*long
- report option T*long
- ReplyTo queue name T*char [48]
- ReplyTo queue manager T*char [48]
- correlative identifier T*octet [24]
- integrity required T*long

confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
service return code	T*service_return
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_getrply verb.

This call allows a service response (corresponding to a previous service request; see D*csi_acall) to be retrieved.

The 'quality of service' parameter, which is returned as output parameter, indicates whether the data compression and the security options (sealing, deciphering) have been applied on message data or not.

The 'service return code' output parameter corresponds to the 'rcode' parameter of the tpreturn verb, which is available for the receiver in the 'tpurcode' variable.

The possible values for the 'flags' parameter are mapped onto the 'flags' parameter of the tpgetrply verb in the XATMI specification. For instance, it is possible to indicate that the call is indefinitely blocking or not.

Name : D*csi_init_sec_context

Composition :

csi_init_sec_context input data

token to send	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]

csi_init_sec_context output data

return token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
CSI return code	T*return_code
CSI reason code	T*reason_code
security protocol status	T*long

Comment : This structure describes the data associated with the CSI_init_sec_context verb, which initiates a security context with a peer application.

The output 'security protocol status' parameter is a flag, which indicates whether the security context initialisation is achieved (COMPLETE) or not (CONTINUE).

The 'return token' is provided by the peer entity. It is expected only if the initialisation is to be continued.

Name : D*csi_mq_browse

Composition :

csi_mq_browse input data

CSI queue connection handle	T*MQHCONN
CSI queue handle	T*MQHOBJ

csi_mq_browse input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue browse message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

csi_mq_browse output data

received message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
message length	T*long

quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_browse verb, which retrieves messages from an opened queue non-destructively. It is mapped onto the MQGET call at the MQI interface (with BROWSE options).

Name : D*csi_mq_close

Composition :

csi_mq_close input data

CSI queue connection handle	T*MQHCONN
CSI queue options	T*long

csi_mq_close input/output data

CSI queue object handle	T*MQHOBJ

csi_mq_close output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_close verb.

This call releases the access of the caller to the CSI queue specified by the input parameters. It is mapped onto the MQCLOSE call at the MQI interface.

Name : D*csi_mq_conn

Composition :

csi_mq_conn input data

CSI queue manager name	T*char [48]

csi_mq_conn output data

CSI queue connection handle	T*MQHCONN

CSI reason code T*reason_code

Comment : This structure describes the data associated with the CSI_mq_conn verb.

This call connects an application to a queue manager.

The call is mapped onto a MQCONN call at the MQI interface. Refer to the MQCONN call (described in the MQSeries Technical Reference).

Name : D*csi_mq_delete

Composition :

csi_mq_delete input data

CSI queue connection handle	T*MQHCONN
CSI queue handle	T*MQHOBJ
required message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue delete message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

csi_mq_delete output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_delete verb.

This call deletes a fetched message from an opened queue. It is mapped onto the MQGET call at the MQI interface (without BROWSE options), and it does not return the message to be deleted.

Name : D*csi_mq_disc

Composition :

csi_mq_disc input/output data

CSI queue connection handle	T*MQHCONN

csi_mq_disc output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_disc verb.

This call breaks the connection between an Application and a queue manager.

The call is mapped onto a MQDISC call at the MQI interface. Refer to the MQDISC call (described in the MQSeries Technical Reference).

Name : D*csi_mq_get

Composition :

csi_mq_get input data

CSI queue connection handle	T*MQHCONN
CSI queue handle	T*MQHOBJ

csi_mq_get input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]

accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue get message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

csi_mq_get output data

received message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
message length	T*long
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_get verb.

This call gets a message from an opened queue. It can be used repeatedly to get many messages from the same queue. A retrieved message is deleted from the queue.

At the CSI interface, a message is passed through a 'DataDescriptor' structure, which allows to receive data either by memory or by file.

The 'message descriptor' structure indicates, as input parameter, the attributes awaited for the message and as output parameter, the attributes the message effectively has.

The 'quality of service' parameter, which is returned as output parameter, indicates whether the data compression and the security options (sealing, ciphering) have been applied on message data or not.

The call is mapped onto the MQGET call at the MQI interface. Refer to the MQGET call (described in the MQSeries Technical Reference).

Name : D*csi_mq_open

Composition :

csi_mq_open input data

CSI queue connection handle	T*MQHCONN
CSI queue open options	T*long

csi_mq_open input/output data

object descriptor	D*MQOD
structure identifier	T*char [4]
structure version number	T*long
object type	T*long
object name	T*char [48]
object queue manager name	T*char [48]
dynamic queue name	T*char [48]
alternate user identifier	T*char [12]

csi_mq_open output data

CSI queue handle	T*MQHOBJ
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_open verb.

This call establishes access to a message queue.

'CSI queue open options' indicates the access rights of the caller to the queue (input/output, shared/exclusive, browsing capability).

The call is mapped onto a MQOPEN call at the MQI interface. Refer to the MQOPEN call (described in the MQSeries Technical Reference).

Name : D*csi_mq_put

Composition :

csi_mq_put input data

CSI queue connection handle	T*MQHCONN
CSI queue object handle	T*MQHOBJ
message to send	D*DATA_DESCRIPTOR

structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

csi_mq_put input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]

Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue put message options	D*MQPMO
Structure identifier	T*char [4]
Structure version number	T*long
Options	T*long
Timeout	T*long
Context	T*MQHOBJ
KnownDescCount	T*long
UnknownDescCount	T*long
resolved queue name	T*char [48]
resolved queue manager	T*char [48]

csi_mq_put output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_put verb.

This call puts a message in an opened queue. It is mapped onto the MQPUT call at the MQI interface. It is used when multiple messages need to be put on the same queue.

At the CSI interface, a message is described through a 'DataDescriptor' structure, which allows to send data either by memory or by file.

The 'quality of service' parameter, which is given as input parameter, indicates the Quality of Service asked by the Application.

Name : D*csi_mq_put1

Composition :

csi_mq_put1 input data

CSI queue connection handle	T*MQHCONN
object descriptor	D*MQOD
structure identifier	T*char [4]
structure version number	T*long
object type	T*long
object name	T*char [48]
object queue manager name	T*char [48]
dynamic queue name	T*char [48]
alternate user identifier	T*char [12]
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS

structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

csi_mq_put1 input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue put message options	D*MQPMO
Structure identifier	T*char [4]
Structure version number	T*long
Options	T*long

Timeout	T*long
Context	T*MQHOBJ
KnownDescCount	T*long
UnknownDescCount	T*long
resolved queue name	T*char [48]
resolved queue manager	T*char [48]

csi_mq_put1 output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_mq_put1 verb.

This call puts a message in a queue; the queue does not need to be open. It is mapped onto the MQPUT1 call at the MQI interface. This call gives better performance for a server replying to different queues.

It has the same description as the CSI_mq_put except for the 'object descriptor' input parameter which replaces the 'CSI queue object handle'.

Name : D*csi_process_context_token

Composition :

csi_process_context_token input data

token to send	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]

csi_process_context_token output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_process_context_token verb. This call sends a security token, involved in a password update process, to a peer application .

Name : D*csi_recv

Composition :

csi_recv input data

conversation connect descriptor	T*hconn
flags	T*long

csi_recv output data

received message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long

specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
event type	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code
service return code	T*service_return

Comment : This structure describes the data associated with the CSI_recv verb. This call is used to receive data sent through a conversational connection.

The 'quality of service' parameter indicates the particular handling applied on the received data during the call execution.

The 'event type' parameter indicates if a particular event occured in the peer entity (service abortion, conversation control exchange).

The 'service return code' is a specific value provided by the peer entity if it has executed a CSI_return call.

Name : D*csi_return

Composition :

csi_return input data

CSI return code	T*return_code
application defined return code	T*service_return
service response information	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long

report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
request/response flags	T*long

Comment : This structure describes the data associated with the CSI_return verb.

This call is used by a service routine to indicate that its service is completed and to send the service's reply message.

A service routine does not return any value to its caller, thus no output parameters are returned by the CSI_return verb.

The 'quality of service' parameter is given to indicate the Quality of Service of the service response message.

The service response message is transmitted in a 'DataDescriptor' structure.

Other parameters are the same as the 'tpreturn' verb described in the XATMI specification.

Name : D*csi_send

Composition :

csi_send input data

conversation connect descriptor	T*hconn
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]

integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long

csi_send output data

event type	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_send verb. This call sends a message to a peer entity through a conversational connection.

The 'quality of service' parameter indicates the particular handling applied on the received data during the call execution.

The 'event type' parameter indicates if a particular event occured in the peer entity (abnormal service termination).

Name : D*csi_service

Composition :

csi_service input data

CSI service information	D*SERVICE_INFO
structure identifier	T*char [4]
version number	T*long
service name	T*char [48]
service data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
requested quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long

compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long
conversation connect descriptor	T*hconn
application key	T*long
client identifier	T*long

Comment : This structure describes the data associated with the CSI_service verb.

This call is a template for CSI service routines which is invoked by the Low Level Function layer core during the execution of its own 'tpservice' function.

Like for the 'tpservice' verb, no output parameters are returned.

Name : D*csi_svcstart

Composition :

csi_svcstart output data

CSI service type	T*long
CSI service information	D*SERVICE_INFO
structure identifier	T*char [4]
version number	T*long
service name	T*char [48]
service data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
requested quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]

VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long
conversation connect descriptor	T*hconn
application key	T*long
client identifier	T*long
CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_scvstart verb.

This call allows a server application to inform its dedicated Remote API Proxy that it is ready to process incoming service requests. The output parameters give back the request service type (svrinit, svrdone or service request) and information (name, request parameters).

Name : D*csi_svrdone

Composition :
empty structure

Comment : This structure is associated with the CSI_svrdone verb.

This call is a CSI server termination routine which is called by the CSI Low Level Function layer during the execution of its own 'tpsvrdone' function. The 'tpsvrdone' function is itself activated by the Tuxedo TP Monitor.

Like the 'tpsvrdone' verb, CSI_svrdone has neither input, nor output parameters.

Name : D*csi_svrinit

Composition :
csi_svrinit input data

CSI server arguments	T*char [256]

csi_svrinit output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_svrinit verb.

This call is a CSI server initialisation routine which is called by the CSI Low Level Function layer during the execution of its own 'tpsvrinit' function. The 'tpsvrinit' function is itself activated by the Tuxedo TP Monitor.

The 'CSI server arguments' parameter contains application specific arguments usually separated by blank characters.

Name : D*csi_unadvertise

Composition :
csi_unadvertise input data

CSI service name	X*ccnServiceName

csi_unadvertise output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_unadvertise verb.

This call allows a server to unadvertise a service that it offers.

Name : D*csi_unbind

Composition :

csi_unbind output data

CSI return code	T*return_code
CSI reason code	T*reason_code

Comment : This structure describes the data associated with the CSI_unbind verb.

As an Application can not have more than one connection in the same time, no input parameter is required to identify the connection to be closed.

6.4. SPI API

Name : D*spi_bind

Composition :

spi_bind input data

application name	X*ccnApplicationName
proxy name	X*ccnApplicationName

spi_bind output data

SPI bind handle	T*hbind
SPI return code	T*return_code
SPI reason code	T*reason_code

Comment : This structure describes the data associated with the SPI_bind verb.

This call allows a proxy running on a Gateway to bind with a server Application running on a peer Application Platform. The call conveys a proxy name and an application name that shall be used by the communication layers for establishing the actual connection. The caller gets back a bind handle that has to be used for all subsequent operations on that connection.

Name : D*spi_call

Composition :

spi_call input data

SPI bind handle	T*hbind
request message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]

data length	T*ulong
data buffer address	T*octet [ ]

spi_call input/output data

verb identifier	T*long

spi_call output data

resulting message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
SPI return code	T*return_code
SPI reason code	T*reason_code

Comment : On reception of a request for a remote CSI verb execution, the SPI layer invokes a RAP function. This RAP function was previously declared with the SPI_setcallback call, and the D*spi_call structure describes the data associated with this RAP function.

The 'request message' parameter contains the input data of the callback function. The 'verb identifier' represents, as input parameter, the request verb and, as output parameter, the response verb. The result of the callback function is enclosed in the 'resulting message' structure.

Name : D*spi_loop

Composition :

spi_loop input data

SPI bind handle	T*hbind
anticipated message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
anticipated verb identifier	T*long

spi_loop output data

SPI return code	T*return_code
SPI reason code	T*reason_code

Comment : This structure describes the data associated with the SPI_loop verb.

This verb is called by the RAP as soon as it is ready to execute remote CSI verbs (SPI_bind and SPI_setcallback calls were successfully performed previously).

Upon reception, the SPI layer sends the data enclosed in the 'anticipated message' parameter (if any). The latter corresponds to the response of the CSI verb indicated by the 'anticipated verb identifier' parameter. For

optimisation reason, this response is anticipated to a CSI request that the Application Platform does not send.

Afterwards, the SPI layer waits for incoming remote CSI requests and process them using a callback function. This loop terminates when the callback function returns a specific code (SPI_LOOP_SHOULD_BREAK).

Name : D*spi_setcallback

Composition :

spi_setcallback input data

SPI bind handle	T*hbind
callback function address	T*prog_ref

spi_setcallback output data

SPI return code	T*return_code
SPI reason code	T*reason_code

Comment : This structure describes the data associated with the SPI_setcallback verb.

This call allows a server RAP to provide the SPI layer with the address of a callback function. This function is in charge of the remote execution of the CSI verbs.

Name : D*spi_unbind

Composition :

spi_unbind input data

SPI bind handle	T*hbind

spi_unbind output data

SPI return code	T*return_code
SPI reason code	T*reason_code

Comment : This structure describes the data associated with the SPI_unbind verb.

This call allows a RAP to unbind from a peer server Application and to close the underlying communication channel between the Gateway and the Application Platform.

6.5. PRES API

Name : D*pres_alloc

Composition :

pres_alloc input data

requested size	T*ulong

pres_alloc output data

data descriptor	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]

data length	T*ulong
data buffer address	T*octet [ ]
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_alloc verb.

This call allocates a data descriptor for the calling entity. The 'requested size' parameter indicates the number of bytes required for the data buffer.

Name : D*pres_bind

Composition :

pres_bind input data

application name	X*ccnApplicationName

pres_bind output data

PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : D*pres_bind is a structured definition of the parameters applying to pres_bind verb.

This call allows an entity (application, proxy, ...) identified by the 'application name' parameter to initialise a session with the Presentation layer.

Name : D*pres_compress

Composition :

pres_compress input data

algorithm	T*char [8]
data to compress	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

pres_compress output data

compressed data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_compress verb.

This call allows an application or a proxy to compress data before sending them to a peer entity over the National network.

Name : D*pres_decode

Composition :

pres_decode input data

message to decode	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
host format	T*host_form
native code page	T*code_page

pres_decode output data

decoded message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_decode verb which is in charge of retrieving a typed message into its native representation.

The native format and code page are provided by the caller as input parameters.

Name : D*pres_encode

Composition :

pres_encode input data

message to encode	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
message type identifier	T*msgtype_id
code page	T*code_page
host format	T*host_form

pres_encode output data

encoded message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_encode verb which is used to type a mesage, and possibly to encode it (i.e. to translate it into the pivot format).

The required information for typing the message is provided by the caller in the 'message type identifier' parameter. The 'host format' and 'code page' items are used in case encoding is processed.

Name : D*pres_free

Composition :

pres_free input/output data

data descriptor to release	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

pres_free output data

PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_free verb.

This call is provided to the Application in order to release buffers or files which were returned to the Application as output parameters, but which were not allocated by the Application itself.

Name : D*pres_gettype

Composition :

pres_gettype input data

message to process	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

pres_gettype output data

message type identifier	T*msgtype_id
code page	T*code_page
host format	T*host_form
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_gettype verb.

This call allows an entity to get the message type identifier, the code page and the host format associated with a typed message without any impact on the message itself (unlike the PRES_decode verb).

Name : D*pres_unbind

Composition :

pres_unbind output data

PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_unbind verb.

This call allows an entity to terminate a session with the Presentation layer.

Name : D*pres_uncompress

Composition :

pres_uncompress input data

data to uncompress	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

pres_uncompress output data

uncompressed data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
PRES return code	T*return_code
PRES reason code	T*reason_code

Comment : This structure describes the data associated with the PRES_uncompress verb.

This call allows an application or a proxy to uncompress data received from a peer entity over the National network.

6.6. GSS API

Name : D*gss_accept_sec_context

Composition :

gss_accept_sec_context input data

verifier_cred_handle	T*gss_cred_id_t
input token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
input_chan_bindings	D*GSS_CHAN_BINDINGS

opaque structure

gss_accept_sec_context input/output data

context handle	T*gss_ctx_id_t

gss_accept_sec_context output data

minor status	T*ulong
src_name	D*GSS_NAME_T
user_id	X*ccnUserName
application_id	X*ccnApplicationName
user_password	X*ccnUserPassword
application_key	X*ccnApplicationSecurityKey
mech_type	D*GSS_OID
length	T*ulong
elements	T*octet [ ]
output token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
ret_flags	T*ulong
time_rec	T*ulong
delegate_cred_handle	T*gss_cred_id_t

Comment : D*gss_accept_sec_context is a structured definition of the parameters applying to the gss_accept_sec_context verb.

This call verifies the incoming token and returns the authenticated internal name as well as the mechanism types used.

Input data:

+ verifier_cred_handle: credential handle obtained by a call to gss_acquire_cred.

+ input token: token obtained from remote application.

+ input_chan_bindings: not used.

Input/output data:

+ context handle: context handle for a new context. (Supply GSS_C_NO_CONTEXT for the first call. Use the value returned in subsequent calls.)

Output data:

+ src_name: authenticated name of context initiator (including user name and application id). This value is returned when the context initialisation completes successfully.

+ mech_type: security mechanism used.

+ output token: token to be passed to the peer application. If the length field of the returned token is 0, no token need to be passed to the peer application.

+ ret_flags: flag set indicating the specific service option supported (mutual authentication, replay attack protection...).

+ time_rec: GSS_C_INDEFINITE, since the implementation does not support credential expiration.

+ delegate_cred_handle: not valid because credential delegation is not supported.

+ minor status: mechanism specific status code.

Name : D*gss_acquire_cred

Composition :

gss_acquire_cred input data

desired_id	D*CSISECINFO
authentication provider id	T*long
security information	T*char []
time_req	T*ulong
desired_mechs	D*GSS_OID_SET
count	T*int
elements	D*GSS_OID
length	T*ulong
elements	T*octet [ ]
cred_usage	T*long

gss_acquire_cred output data

output_cred_handle	T*gss_cred_id_t
actual_mechs	D*GSS_OID_SET
count	T*int
elements	D*GSS_OID
length	T*ulong
elements	T*octet [ ]
time_rec	T*long
minor status	T*ulong

Comment : D*gss_acquire_cred is a structured definition of the parameters applying to the gss_acquire_cred verb.

This call allows an application to acquire a credential so that it can initiate or accept security contexts under the identity represented by the 'desired_id' parameter.

Input data:

+ desired_id: security provider-dependent information identifying the principal whose credential is required.

On the initiator side, this structure contains the user and application id and master keys.

On the acceptor side, this structure contains the RAP identifier.

+ time_req: number of seconds that credential remains valid ; not used.

+ desired_mechs: set of underlying security mechanisms that may be used. On the initiator side, GSS_C_NULL_OID_SET shall be used to obtain the default mechanism.

+ cred_usage: this value shall be set to GSS_C_INITIATE on the initiator side and GSS_C_ACCEPT on the acceptor side. It indicates the desired usage of the credential.

Output data:

+ output_cred_handle: the returned credential handle.

+ actual_mechs: the set of mechanisms for which the credential is valid.

+ time_rec: GSS_C_INDEFINITE. The implementation does not support expiration of credentials.

+ minor status: mechanism specific status code.

Name : D*gss_delete_sec_context

Composition :

gss_delete_sec_context input data

context_handle	T*gss_ctx_id_t

gss_delete_sec_context output data

output token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
minor status	T*ulong

Comment : D*gss_delete_sec_context is a structured definition of the parameters applying to the gss_delete_sec_context verb.

This call allows an entity to delete a security context. It is available for both initiator and acceptor peers.

Input data:

+ context handle: handle identifying the context to delete.

Output data:

+ output token: token to be passed to the peer Application to instruct it to also delete the context.

+ minor status: mechanism specific status code.

Name : D*gss_init_sec_context

Composition :

gss_init_sec_context input data

claimant_cred_handle	T*gss_cred_id_t
target_name	D*CSISECINFO
authentication provider id	T*long

security information	T*char []
mech_type	D*GSS_OID
length	T*ulong
elements	T*octet [ ]
req_flags	T*ulong
time_req	T*ulong
input_chan_bindings	D*GSS_CHAN_BINDINGS
opaque structure	
input token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
gss_init_sec_context input/output data	
context handle	T*gss_ctx_id_t
gss_init_sec_context output data	
actual_mech_type	D*GSS_OID
length	T*ulong
elements	T*octet [ ]
output token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
ret_flags	T*ulong
time_req	T*ulong
minor status	T*ulong

Comment : D*gss_init_sec_context is a structured definition of the parameters applying to the gss_init_sec_context verb.

This call initiates the establishment of a security context between an Application or a RAP and its remote peer.

Input data:

+ claimant_cred_handle: credential handle obtained by a call to gss_acquire_cred.

+ target_name: generic security structure containing the security provider-dependent information identifying the acceptor context.

+ mech_type: security mechanism: supply GSS_C_NULL_OID to obtain the default mechanism.

+ req_flags: flag set. Each flag request that the context support a specific service option.

+ time_req: context validity period. Not used since no lifetime mechanism is supported.

+ input_chan_bindings: not used.

+ input token: token obtained from remote application. Supply GSS_C_NO_BUFFER on initial call.

Input/output data:

+ context handle: handle for a new context. Supply GSS_C_NO_CONTEXT for the first call. Use the value returned in subsequent calls.

Output data:

+ actual_mech_type: security mechanism used.

+ output token: token to be passed to the peer application. If the length field of the returned token is 0, no token need to be passed to the peer Application.

+ ret_flags: flag set indicating the specific service option supported (mutual authentication, replay attack protection...).

+ time_rec: GSS_C_INDEFINITE, since the implementation does not support credential expiration.

+ minor status: mechanism specific status code.

Name : D*gss_password_update

Composition :

gss_password_update input data

context_handle	T*gss_ctx_id_t
type_usage	T*long
password information	D*CSISECINFO
authentication provider id	T*long
security information	T*char []

gss_password_update output data

output token	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]
minor status	T*ulong

Comment : D*gss_password_update is a structured definition of the parameters applying to the gss_password_update verb.

This call formats a token from the security information to be sent to the remote peer.

Input data:

+ context handle: handle of the context on which token is to be processed.

+ type_usage: indicates if the 'name' parameter represents a user password or an application key.

+ password information: generic security structure containing the security provider-dependent information required for updating the password (e.g. user or application identifier, old user password or application key, new user password or application key).

Output data:

+ output token: token to generate.

+ minor status: mechanism specific status code.

Name : D*gss_process_context_token

Composition :

gss_process_context_token input data

context_handle	T*gss_ctx_id_t
token_buffer	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]

gss_process_context_token output data

minor status	T*ulong

Comment : D*gss_process_context_token is a structured definition of the parameters applying to the gss_process_context_token verb.

This call provides a way to pass a token to the underlying security service. Such a token may have been generated during a context deletion or a password update.

Input data:

+ context handle: handle of the context on which token is to be processed.

+ token_buffer: token to be processed.

Output data:

+ minor status: mechanism specific status code.

Name : D*gss_release_buffer

Composition :

gss_release_buffer input data

buffer	D*GSS_BUFFER_T
length	T*long
value	T*octet [ ]

gss_release_buffer output data

minor status	T*ulong

Comment : D*gss_release_buffer is a structured definition of the parameters applying to the gss_release_buffer verb.

This call frees the storage area associated with a buffer previously allocated by a GSS API function.

Input data:

+ buffer: buffer to release.

Output data:

+ minor status: mechanism specific status code.

Name : D*gss_release_cred

Composition :

gss_release_cred input data

cred_handle	T*gss_cred_id_t

gss_release_cred output data

minor status	T*ulong

Comment : D*gss_release_cred is a structured definition of the parameters applying to the gss_release_cred verb.

This call deletes a credential no longer required by a process.

Input data:

+ cred_handle: handle of the credential to delete.

Output data:

+ minor status: mechanism specific status code.

Name : D*gss_release_name

Composition :

gss_release_name input data

name	D*GSS_NAME_T
user_id	X*ccnUserName
application_id	X*ccnApplicationName
user_password	X*ccnUserPassword
application_key	X*ccnApplicationSecurityKey

gss_release_name output data

minor status	T*ulong

Comment : D*gss_release_name is a structured definition of the parameters applying to the gss_release_name verb.

This call frees the storage area associated with a name.

Input data:

+ name: address of the structure to delete.

Output data:

+ minor status: mechanism specific status code.

Name : D*gss_release_oid_set

Composition :

gss_release_oid_set input data

set	D*GSS_OID_SET
count	T*int
elements	D*GSS_OID
length	T*ulong
elements	T*octet [ ]

gss_release_oid_set output data

minor status	T*ulong

Comment : D*gss_release_oid_set is a structured definition of the parameters applying to the gss_release_oid_set verb.

This call frees the storage area associated with a gss_OID_set object.

Input data:

+ set: address of the object to delete.

Output data:

+ minor status: mechanism specific status code.

Name : D*gss_unwrap

Composition :

gss_unwrap input data

context_handle	T*gss_ctx_id_t
input_message_buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

gss_unwrap output data

output_message_buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
conf_state	T*long
qop_state	T*long
minor status	T*ulong

Comment : D*gss_unwrap is a structured definition of the parameters applying to the gss_unwrap verb.

This call allows to convert a wrapped message back to a usable form. The message integrity code is verified and deciphering is performed if needed.

Input data:

+ context handle: handle of the context on which the message arrived.

+ input_message_buffer: wrapped message. This data structure is an extended form of the GSS-BUFFER_T specific to the CCN/CSI project.

Output data:

+ output_message_buffer: data structure to receive the unwrapped message. This data structure is an extended form of the GSS-BUFFER_T specific to the CCN/CSI project.

+ conf_stat: indicates if confidentiality is used or not.

+ qop_stat: quality of protection gained from the message integrity code.

+ minor status: mechanism specific status code.

Name : D*gss_wrap

Composition :

gss_wrap input data

context_handle	T*gss_ctx_id_t
conf_req_flag	T*long
qop_req	T*long
input_message_buffer	D*DATA_DESCRIPTOR

structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

gss_wrap output data

conf_state	T*long
output_message_buffer	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
minor status	T*ulong

Comment : D*gss_wrap is a structured definition of the parameters applying to the gss_wrap verb.

This call allows to sign a message and optionally to cipher it

Input data:

+ context handle: handle of the context on which the message is sent.

+ conf_req_stat: indicates if confidentiality is requested or not.

+ qop_req: quality of protection (shall be set to GSS_C_QOP_DEFAULT).

+ input_message_buffer: message to wrap. This data structure is an extended form of the GSS-BUFFER_T specific to the CCN/CSI project.

Output data:

+ conf_stat: indicates if confidentiality has been applyed or not.

+ output_message_buffer: data structure to receive the wrapped message. This data structure is an extended form of the GSS-BUFFER_T specific to the CCN/CSI project.

+ minor status: mechanism specific status code.

6.7. HL API

Name : D*hl_acall

Composition :

hl_acall input data

HL service name	X*ccnServiceName
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]

data length	T*ulong
data buffer address	T*octet [ ]
flags	T*long
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

hl_acall output data

request/response call handle	T*hcall
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_acall verb.

This call has the same description and parameters as the CSI_acall verb.

It returns a descriptor that can be used by the HL_getrply call to receive the reply of the request.

Name : D*hl_advertise

Composition :

hl_advertise input data

HL service name	X*ccnServiceName
function address/prog_name	T*prog_ref

hl_advertise output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_advertise verb.

This call has the same description and parameters as the CSI_advertise call.

Name : D*hl_alloc

Composition :

hl_alloc input data

requested size	T*ulong

hl_alloc output data

data descriptor	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_alloc verb.

This call has the same description and parameters as the PRES_alloc call.

Name : D*hl_bind

Composition :

hl_bind input data

application name	X*ccnApplicationName
proxy name	X*ccnApplicationName

hl_bind output data

default QOS	X*ccnDefaultQualityOfService
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_bind verb.

This call has the same description and parameters as the CSI_bind verb.

Name : D*hl_call

Composition :

hl_call input data

CSI service name	X*ccnServiceName
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
flags	T*long

hl_call input/output data

quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long

urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

hl_call output data

service response	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
service return code	T*service_return
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_call verb.

This call has the same description and parameters as the CSI_call verb.

Name : D*hl_cancel

Composition :

hl_cancel input data

request/response call descriptor	T*hcall

hl_cancel output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_cancel verb.

This call has the same description and parameters as the CSI_cancel call.

Name : D*hl_connect

Composition :

hl_connect input data

service name	X*ccnServiceName
application data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong

file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long

hl_connect output data

conversation connect handle	T*hconn
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_connect verb.

This call has the same description and parameters as the CSI_connect call.

Name : D*hl_decode

Composition :

hl_decode input data

message to decode	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
expected code page	T*code_page
expected host format	T*host_form

hl_decode output data

decoded message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong

file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_decode verb.

This call has the same description and parameters as the PRES_decode call.

Name : D*hl_delete_sec_context

Composition :

hl_delete_sec_context output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_delete_sec_context verb.

This call deletes a security context with a peer application. It is mapped onto the gss_delete_sec_context and CSI_delete_sec_context calls.

Moreover, if a connection with a queue manager already exists, the HL layer firstly closes it prior processing the context deletion.

This call has no input parameter.

Name : D*hl_discon

Composition :

hl_discon input data

conversation connect handle	T*hconn

hl_discon output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_discon verb.

This call has the same description and parameters as the CSI_discon call.

Name : D*hl_encode

Composition :

hl_encode input data

message to encode	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
Message type identifier	T*msgtype_id

Code Page	T*code_page
host format	T*host_form

hl_encode output data

encoded message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_encode verb.

This call has the same description and parameters as the PRES_encode call.

Name : D*hl_free

Composition :

hl_free input data

data descriptor to release	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

hl_free output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_free verb.

This call has the same description and parameters as the PRES_free call.

Name : D*hl_getrply

Composition :

hl_getrply input data

request/response call descriptor	T*hcall
request/response flags	T*long

hl_getrply output data

service response	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong

data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
service return code	T*service_return
HL return code	T*return_code
HL reason code	T*reason_code
request/response connect descriptor	T*ulong

Comment : This structure describes the data associated with the HL_getrply verb.

This call has the same description and parameters as the CSI_getrply call.

Name : D*hl_gettype

Composition :

hl_gettype input data

message to process	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

hl_gettype output data

message type identifier	T*msgtype_id
code page	T*code_page
host format	T*host_form
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_gettype verb.

This call has the same description and parameters as the PRES_gettype call.

Name : D*hl_init_sec_context

Composition :

hl_init_sec_context input data

credential information	D*CSISECINFO
authentication provider id	T*long
security information	T*char []

hl_init_sec_context output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_init_sec_context verb.

This call initiates a security context with a peer application.

It is mapped onto gss_init_sec_context and CSI_init_sec_context calls.

On input, this call receives, as unique parameter, a generic security structure which provides security information about the User and the Application (e.g. user name and password, application name and key).

Name : D*hl_mq_browse

Composition :

hl_mq_browse input data

CSI queue connection handle	T*MQHCONN
CSI queue handle	T*MQHOBJ

hl_mq_browse input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]

Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue browse message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]
message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
message length	T*long

hl_mq_browse output data

quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_browse verb.

This call has the same description and parameters as the CSI_mq_browse call.

Name : D*hl_mq_close

Composition :

hl_mq_close input data

CSI queue connection handle	T*MQHCONN
CSI queue options	T*long

hl_mq_close input/output data

CSI queue object handle	T*MQHOBJ

hl_mq_close output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_close verb.

This call has the same description and parameters as the CSI_mq_close call.

Name : D*hl_mq_conn

Composition :

hl_mq_conn input data

CSI queue manager name	T*char [48]

hl_mq_conn output data

CSI queue connection handle	T*MQHCONN
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_conn verb.

This call has the same description and parameters as the CSI_mq_conn call. However, it is optional for the default queue manager. If the connection is not established when the first access to one of the default manager queues is performed, the HL layer firstly performs a CSI_mq_conn call prior accessing the queue.

Name : D*hl_mq_delete

Composition :

hl_mq_delete input data

CSI queue connection handle	T*MQHCONN
CSI queue handle	T*MQHOBJ
required message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long

Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue delete message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

hl_mq_delete output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_delete verb.

This call has the same description and parameters as the CSI_mq_delete call.

Name : D*hl_mq_disc

Composition :

hl_mq_disc input/output data

CSI queue connection handle	T*MQHCONN

hl_mq_disc output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_disc verb.

This call has the same description and parameters as the CSI_mq_disc call. However, it is optional. If the connection is still opened when a HL_delete_security_context call or a HL_unbind call is requested, the HL layer firstly disconnects from the queue manager.

Name : D*hl_mq_get

Composition :

hl_mq_get input data

CSI queue connection handle	T*MQHCONN

CSI queue handle	T*MQHOBJ

hl_mq_get input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue get message options	D*MQGMO
structure identifier	T*char [4]
structure version number	T*long
options	T*long
wait interval	T*long
signal1	T*long
signal2	T*long
resolved queue name	T*char [48]

hl_mq_get output data

received message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
message length	T*long
quality of service	D*QOS
structure identifier	T*char [4]

version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_get verb.

This call has the same description and parameters as the CSI_mq_get call.

Name : D*hl_mq_open

Composition :

hl_mq_open input data

CSI queue connection handle	T*MQHCONN
CSI queue open options	T*long

hl_mq_open input/output data

object descriptor	D*MQOD
structure identifier	T*char [4]
structure version number	T*long
object type	T*long
object name	T*char [48]
object queue manager name	T*char [48]
dynamic queue name	T*char [48]
alternate user identifier	T*char [12]

hl_mq_open output data

CSI queue handle	T*MQHOBJ
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_open verb.

This call has the same description and parameters as the CSI_mq_open call.

However, if the connection with the queue manager is not already established, the HL layer firstly issues a CSI_mq_conn call prior opening the queue.

Name : D*hl_mq_put

Composition :

hl_mq_put input data

CSI queue connection handle	T*MQHCONN
CSI queue object handle	T*MQHOBJ
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

hl_mq_put input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]

ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue put message option	D*MQPMO
Structure identifier	T*char [4]
Structure version number	T*long
Options	T*long
Timeout	T*long
Context	T*MQHOBJ
KnownDescCount	T*long
UnknownDescCount	T*long
resolved queue name	T*char [48]
resolved queue manager	T*char [48]

hl_mq_put output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_put verb.

This call has the same description and parameters as the CSI_mq_put call.

Name : D*hl_mq_put1

Composition :

hl_mq_put1 input data

CSI queue connection handle	T*MQHCONN
object descriptor	D*MQOD
structure identifier	T*char [4]
structure version number	T*long
object type	T*long
object name	T*char [48]
object queue manager name	T*char [48]
dynamic queue name	T*char [48]
alternate user identifier	T*char [12]
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS

structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long

hl_mq_put1 input/output data

message descriptor	D*MQMD
structure identifier	T*char [4]
structure version number	T*long
report options	T*long
message type	T*long
expiry time	T*long
feedback/reason code	T*long
data encoding	T*long
coded character set identifier	T*long
format name	T*char [8]
Priority	T*long
Persistence	T*long
Message Identifier	T*octet [24]
Correlation identifier	T*octet [24]
Backout counter	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
User identifier	T*char [12]
accounting token	T*octet [32]
application data relating to identity	T*char [32]
Type of application that put the message	T*long
Name of application that put the message	T*char [28]
Date when message was put	T*char [8]
Time when message was put	T*char [8]
application data relating to origin	T*char [4]
CSI queue put message options	D*MQPMO
Structure identifier	T*char [4]
Structure version number	T*long
Options	T*long

Timeout	T*long
Context	T*MQHOBJ
KnownDescCount	T*long
UnknownDescCount	T*long
resolved queue name	T*char [48]
resolved queue manager	T*char [48]

hl_mq_put1 output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_mq_put1 verb.

This call has the same description and parameters as the CSI_mq_put1 call.

However, if the connection with the queue manager is not already established, the HL layer firstly issues a CSI_mq_conn call prior puting the message in the queue.

Name : D*hl_password_update

Composition :

hl_password_update input data

password information	D*CSISECINFO
authentication provider id	T*long
security information	T*char []

hl_password_update output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_password_update verb.

This call formats a token to be sent to a peer entity for updating a password. It is mapped onto the gss_password_update call.

Name : D*hl_recv

Composition :

hl_recv input data

conversation connect handle	T*hconn
flags	T*long

hl_recv output data

received message	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]

quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
event type	T*long
HL return code	T*return_code
HL reason code	T*reason_code
service return code	T*service_return

Comment : This structure describes the data associated with the HL_recv verb.

This call has the same description and parameters as the CSI_recv call.

Name : D*hl_return

Composition :

hl_return input data

HL return code	T*return_code
application defined return code	T*service_return
service response information	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]

correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
request/response flags	T*long

Comment : This structure describes the data associated with the HL_return verb.

This call has the same description and parameters as the CSI_return call.

Name : D*hl_send

Composition :

hl_send input data

conversation connect handle	T*hconn
message to send	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long

hl_send output data

event type	T*long
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_send verb.

This call has the same description and parameters as the CSI_send call.

Name : D*hl_service

Composition :

hl_service input data

HL service information	D*SERVICE_INFO
structure identifier	T*char [4]
version number	T*long
service name	T*char [48]
service data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
requested quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long
conversation connect descriptor	T*hconn
application key	T*long
client identifier	T*long

Comment : This structure describes the data associated with the HL_service verb.

This call has the same description and parameters as the CSI_service call.

Name : D*hl_svcstart

Composition :

hl_svcstart output data

HL service type	T*long
HL service information	D*SERVICE_INFO
structure identifier	T*char [4]
version number	T*long
service name	T*char [48]
service data	D*DATA_DESCRIPTOR
structure identifier	T*char [4]
structure version number	T*long
flags	T*ulong
file name	T*char [256]
data length	T*ulong
data buffer address	T*octet [ ]
requested quality of service	D*QOS
structure identifier	T*char [4]
version number	T*long
specified QOS	T*long
applied QOS	T*long
urgency	T*long
report option	T*long
ReplyTo queue name	T*char [48]
ReplyTo queue manager	T*char [48]
correlative identifier	T*octet [24]
integrity required	T*long
confidentiality required	T*long
compression option	T*long
compression algorithm identifier	T*char [8]
class of traffic	T*char [16]
VAS script name	T*char [48]
degraded mode flag	T*long
flags	T*long
conversation connect descriptor	T*hconn
application key	T*long
client identifier	T*long
HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_svcstart verb.

This call has the same description and parameters as the CSI_svcstart call.

Name : D*hl_svrdone

Composition :

empty structure

Comment : This structure describes the data associated with the HL_svrdone verb.

This call has the same description and parameters as the CSI_svrdone call.

Name : D*hl_svrinit

Composition :

hl_svrinit input data

HL server arguments	T*char [256]

hl_svrinit output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_svrinit verb.

This call has the same description and parameters as the CSI_svrinit call.

Name : D*hl_unadvertise

Composition :

hl_unadvertise input data

CSI service name	X*ccnServiceName

hl_unadvertise output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_unadvertise verb.

This call has the same description and parameters as the CSI_unadvertise call.

Name : D*hl_unbind

Composition :

hl_unbind output data

HL return code	T*return_code
HL reason code	T*reason_code

Comment : This structure describes the data associated with the HL_unbind verb.

This call has the same description and parameters as the CSI_unbind call.

However, if a connection with a queue manager already exists, the HL layer firstly closes it prior processing the unbind.

7. TABLE OF INDEXES

Page numbers written with bold characters indicate where items or structures are defined, while page numbers written with regular characters indicate where they are used.

END OF DOCUMENT