DSM-CC User-Network Guide

PRELIMINARY DRAFT

2003 01 29

BIONIC BUFFALO CORPORATION

Copyright and License

Author and Publisher

Bionic Buffalo Corporation
2533 North Carson Street, Suite 1884
Carson City, Nevada 89706-0147
United States of America

telephone +1 775 882 1842
fax +1 775 882 6047
e-mail *query@tatanka.com*
web *http://www.tatanka.com*

Related Documents and Products

This document pertains to products licensed by Bionic Buffalo Corporation. For other documents and for additional information, see *http://www.tatanka.com/prod/info/dsmcc.html*, or send e-mail to *query@tatanka.com.*

Document Revision History

Original release, January 29 2003
This document supersedes all previous documents
Update, 2003.01. 31, to correct minor errors and reformat.

Update, 2003.02.03

Last modified 15:32:25, Saturday, 05 April, 2003.

Updates and revisions of this document are announced on Bionic Buffalo's DSM-CC e-mail announcement list. See *http://www.tatanka.com/bbc/lists.html* for information.

Additional Document Information

Project name: italy
Document file name: italyap1 * *italyapi3*

Written and illustrated using StarOffice and OpenOffice.

CONTENTS

PREFACE

This is a guide for those using Bionic Buffalo's Italy software to implement applications acting as DSM-CC User-Network clients and servers.

THIS IS A PRELIMINARY VERSION OF THE FIRST RELEASE OF THIS DOCUMENT.

UPDATED VERSIONS TO FOLLOW REGULARLY.

OVERVIEW

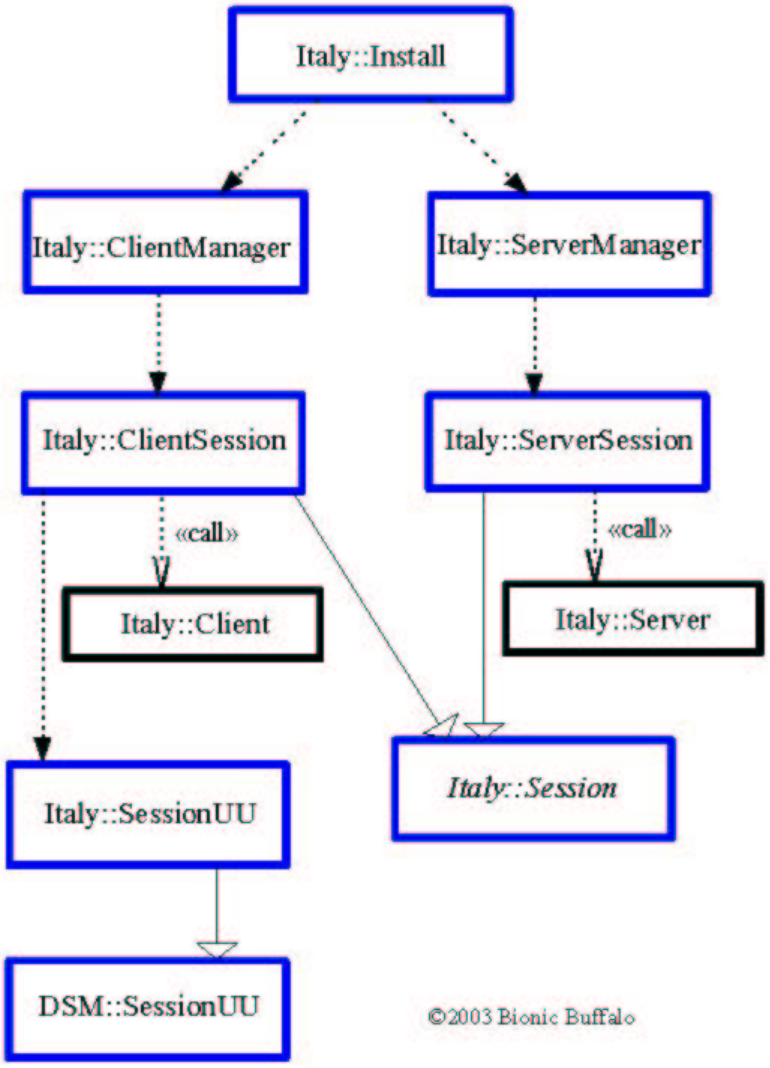
This document provides information for use of the DSM-CC User-Network client and server (Italy) library. It includes a description of the API available to applications, and also a description of objects to be implemented by the applications themselves.

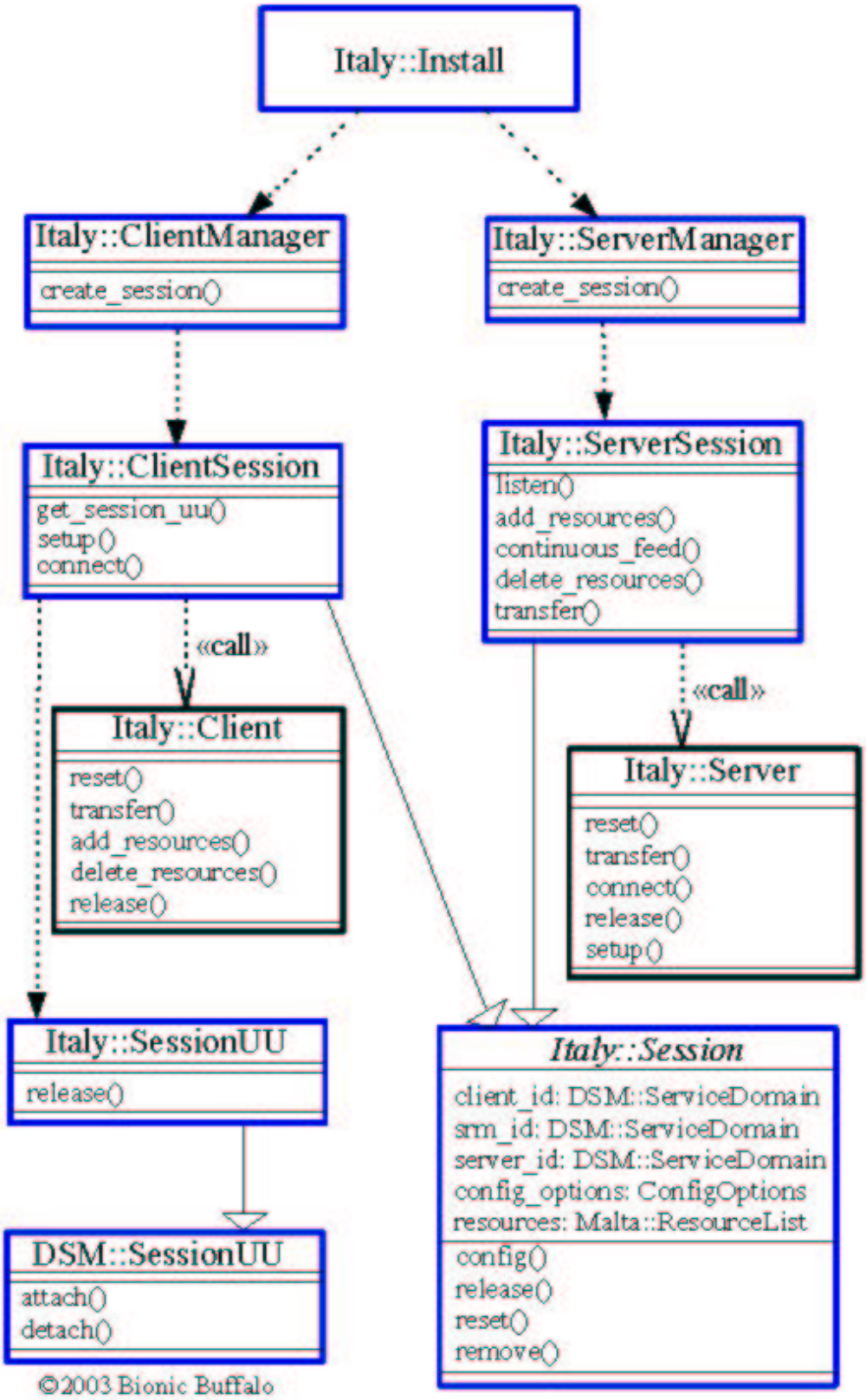
Bionic Buffalo has created various versions of DSM-CC software since 1996. This document is the first released preliminary draft relating to the A-series software versions of the User-Network

This is a **preliminary** release of this document. There is still a list of corrections and additions, so please use with caution.

Enjoy.

An implementation is constructed from various classes of object. Most of them are implemented in the Italy and Malta libraries (provided by Bionic Buffalo), though two classes (`::Italy::Client` and `::Italy::Server`) are to be implemented by the customer. The software is constructed on a CORBA base, so the interface implementations may be distributed over, or invoked from, multiple host platforms.





APPLICATION PROGRAM INTERFACE

This chapter describes the object interfaces defined for DSM-CC User-Network clients and servers. It includes descriptions of interfaces implemented by Bionic Buffalo in the Italy and Malta libraries, as well as interfaces to objects to be implemented by the customer's applications.

Pages 9 - 61

::Italy::Client

Synopsis

```
typedef CORBA_Object          Italy_Client ;
```

Description

An `Italy::Client` object is created by an application using `Italy` Services. The Operations of an `Italy::Client` object provide a mechanism for the `Italy` library routines to notify the application of asynchronous events such as when a server performs a `reset` or adds resources to a session.

Notes

::Italy::Client::add_resources

Synopsis

```
void Italy_Client_add_resources
( Italy_Client                    o,
/* in */ Malta_ResourceList       * resources,
/* in-out */ CORBA_OctetSeq       * uu_data,
/* in-out */ CORBA_OctetSeq       * private_data,
CORBA_Environment                 * ev ) ;
```

Arguments

`resources` (in) the resources being added

`uu_data` (in) user-user data from the server
(out) user-user data returned to the server

`private_data` (in) private data from the server
(out) private data returned to the server

Returns

`void`

Errors

Description

This operation is invoked by the `Italy` library when an incoming `add resource` indication is received. The response message is built using the output values of `uu_data` and `private_data`.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::Client::delete_resources

Synopsis

```
void Italy_Client_delete_resources
     ( Italy_Client                    o,
     /* in */ Malta_ResourceList       * resources,
     /* in-out */ CORBA_OctetSeq       * uu_data,
     /* in-out */ CORBA_OctetSeq       * private_data,
     CORBA_Environment                 * ev ) ;
```

Arguments

`resources` (in) the resources being deleted

`uu_data` (in) user-user data from the server
(out) user-use data to be returned to the server

`private_data` (in) private data from the server
(out) private data to be retruned to the server

Returns

`void`

Errors

Description

This operation is invoked by the `Italy` library when an incoming `delete resource` indication is received. The response indication is received. The response message is built using the output values of a `uu_data` and `private_data`.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::ClientManager

Synopsis

```
typedef CORBA_Object          Italy_ClientManager ;
```

Description

This class is a factory for ClientSession objects.

Notes

1. All `ClientSession` objects created by `Client Manager` share the same address. However, each may communicate with a different SRM and server.

::Italy::ClientManager::create_session

Synopsis

```
void Italy_ClientManager_create_session
     ( Italy_ClientManager                    o,
     /* in */ Italy_Client                    client,
     /* out */ Italy_ClientSession            * session,
     CORBA_Environment                        * ev ) ;
```

Arguments

`client` (in) the applications `Italy::Client` object corresponding to the session

session (out) and `Italy::ClientSession` object to be used by the application

Returns

`void`

Errors

Description

This operation creates a new `Italy::ClientSession` object. It is associated with the specified `Italy::Client` object.

Notes

::Italy::Client::release

Synopsis

```
void Italy_Client_release
     ( Italy_Client                o,
     CORBA_Environment              * ev ) ;
```

Arguments

Returns

`void`

Errors

Description

This operation is invoked by the `Italy` library when and incoming `release` indication is received. The library will send the response message after this operation completes.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::Client::reset

Synopsis

```
void Italy_Client_reset
     ( Italy_Client                o,
     CORBA_Environment             * ev ) ;
```

Arguments

Returns

`void`

Errors

Description

This operation is invoked by the `Italy` library when a relevant reset indication is received. The response message will be sent after all Client::reset operations have completed.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::ClientSession

Synopsis

```
typedef CORBA_Object          Italy_ClientSession ;
```

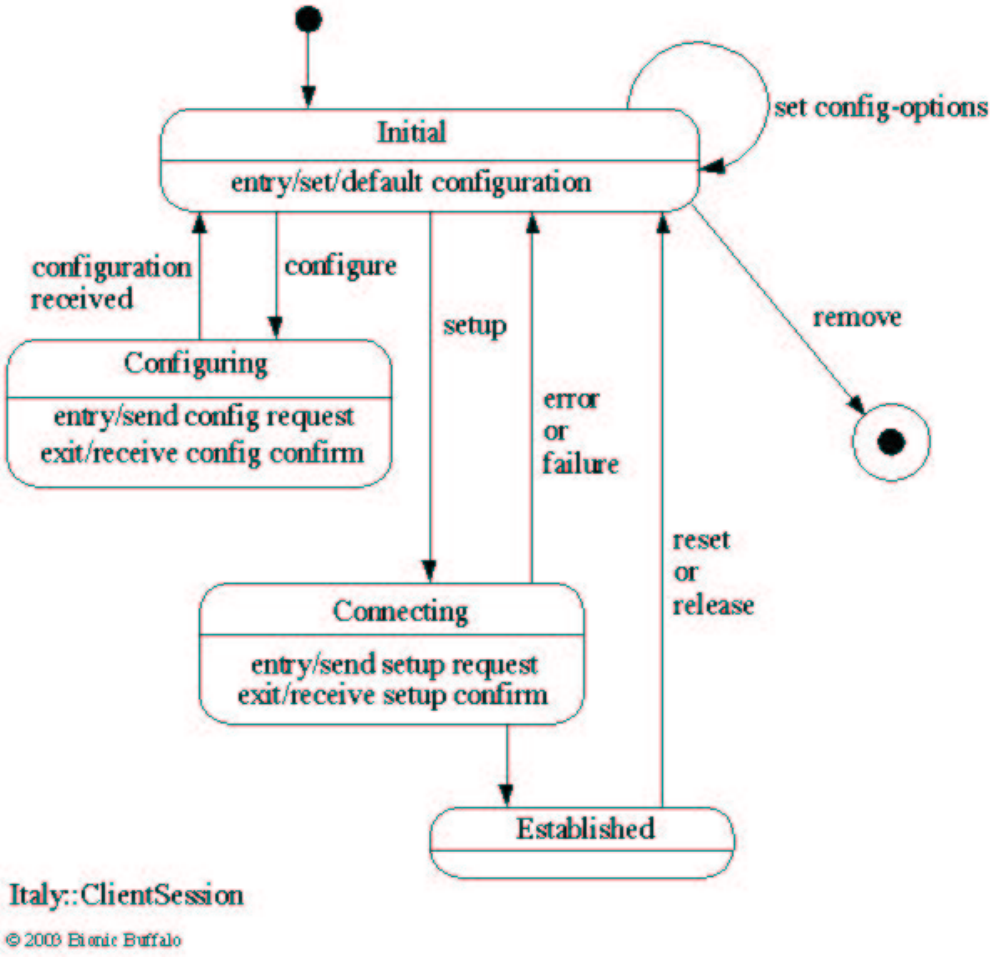
Description

An object of this class corresponds to an individual session.

Notes

1. After the `ClientSession` object is created, the application should configure the object, then call `ClientSession::setup` to start the session.

2. ::Italy::ClientSession Diagram - following page (17)



::Italy::ClientSession::connect

Synopsis

```
void Italy_ClientSession_connect
     ( Italy_ClientSession              o,
     /* in */ CORBA_OctetSeq            * uu_data,
     /* in */ CORBA_OctetSeq            * private_data,
     CORBA_Environment                  * ev ) ;
```

Arguments

`uu_data` (in) user-user data to be passed to the server

`private_data` (in) private date to be passed to the server

Returns

`void`

Errors

Description

This operation is used to notify the server that the client is ready to proceed with user-user communications.

Notes

::Italy::ClientSession::get_session_uu

Synopsis

```
void Italy_ClientSession_get_session_uu
     ( Italy_ClientSession                  o,
     /* out */ Italy_SessionUU              * session_uu,
     CORBA_Environment                      * ev ) ;
```

Arguments

`session_uu` (out) the associated `SessionUU` object

Returns

`void`

Errors

Description

This operation is used to obtain an object which inherits the `DSM::SessionUU` interface and which can be used by the user-user application.

Notes

::Italy::ClientSession::setup

Synopsis

```
void Italy_ClientSession_setup
     ( Italy_ClientSession              o,
     /* in-out */ CORBA_OctetSeq        * uu_data,
     /* in-out */ CORBA_OctetSeq        * private_data,
     /* out */ Malta_ResourceList       ** resources,
     CORBA_Environment                  * ev ) ;
```

Arguments

`uu_data` (in) user-user data to be passed to the server
(out) user-user data returned by the server

`private_data` (in) private data to be passed to the server
(out) private data returned by the server

`resources` (out) resources to be used for this session

Returns

`void`

Errors

Description

This operation begins a session, after the `ClientSession` object has been configured.

Notes

1. If the requested server has forwarded the session to an alternate server, the `ClientSession::server_id` will reflect the new server id value after this operation returns.

::Italy::Client::transfer

Synopsis

```
void Italy_Client_transfer
     ( Italy_Client o,
     /* in */ DSM_ServiceDomain          * old_server,
     /* in */ DSM_ServiceDomain          * new_server,
     /* in */ Malta_ResourceList         * resources,
     /* in-out */ CORBA_OctetSeq         * uu_data,
     /* in-out */ CORBA_OctetSeq         * private_data,
     CORBA_Environment                   * ev ) ;
```

Arguments

`old_server`	(in) the server previously used this session
`new_server`	(in) the new server to be used for this session
`resources`	(in) an updated session resource listed
`uu_data`	(in) user-user data from the new server
`private data`	(in) private data from the new server
	(out) private data to be returned to the new server

Returns

`void`

Errors

Description

This operation is called by the `Italy` library when the session is being transferred to as new server.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::ConfigOptions

Synopsis

```
typedef struct Italy_ConfigOptions
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_unsigned_long      session_msg_size_limit ;
     CORBA_unsigned_long      session_client_port ;
     CORBA_unsigned_long      session_srm_port ;
     CORBA_unsigned_long      session_server_port ;
     CORBA_unsigned_long      config_client_port ;
     CORBA_unsigned_long      config_srm_port ;
     CORBA_unsigned_long      config_server_port ;
     CORBA_unsigned_short     forward_count_limit ;
     CORBA_unsigned_short     message_retry_limit ;
     TimeBase_TimeT           message_retry_interval ;
     TimeBase_TimeT           session_progress_interval ;
     TimeBase_TimeT           timer_resolution ;
     CORBA_octet              session_id_assignor ;
     CORBA_octet              resource_id_assignor ;
     Malta_ProtocolDialect    protocol_dialect ;
     }
     Italy_ConfigOptions ;
```

Members

`session_msg_size_limit ;` largest allowable message (in octlets); used to allocate receive buffers

`session_client_port ;` port number used by client for user-network session protocol (TCP/IP or UDP/IP connections)

`session_srm_port ;` port number used by SRM for user-network session protocol (TCP/IP or UDP/IP connections)

`session_server_port ;` port number used by server for user-network session protocol (TCP/IP or UDP/IP connections)

`config_client_port ;` port number used by client for user-network config protocol (TCP/IP or UDP/IP connections)

`config_srm_port ;` port number used by SRM for user-network config protocol (TCP/IP or UDP/IP connections)

`config_server_port ;` port number used by server for user-network config protocol (TCP/IP or UDP/IP connections)

`forward_count_limit ;` maximum number of times session may be forwarded

`message_retry_limit ;` maximum number of times a message will be resent if no reply is received

`message_retry_interval ;` time without reply before a message is resent

`session_progress_interval ;` time between session progress messages

`timer_resolution ;` resolution of timer used for `message_retry_interval` and `session_progress_interval`

`session_id_assignor ;` who assigns the session identifier

`resource_id_assignor ;` who asigns resource identifiers

`protocol_dialect ;` dialect of protocol used during the sesion

Description

Configuration parameters for an `Italy::Session` object.

Notes

::Italy::Server

Synopsis

```
typedef CORBA_Object          Italy_Server ;
```

Description

An object of this class is created by the application for each session. The operations of the `Italy::Server` object are used by the `Italy` library to notify the application of asynchronous events such as session setup messages and client release requests.

Notes

::Italy::Server::connect

Synopsis

```
void Italy_Server_connect
     ( Italy_Server                o,
     /* in */ CORBA_OctetSeq       * uu_data,
     /* in */ CORBA_OctetSeq       * private_data,
     CORBA_Environment             * ev ) ;
```

Arguments

`uu_data`	(in) user-user data sent by the client
`private_data`	(in) private data sent by the client

Returns

`void`

Errors

Description

The `Italy` library invokes this operation on the `Server` object corresponding to a connected `ServerSession`, to notify the application that a client is ready to begin user-user communication.

Notes

::Italy::ServerManager

Synopsis

```
typedef CORBA_Object          Italy_ServerManager ;
```

Description

`ServerManager` objects are factories for `ServerSession` objects. All of the `ServerSession` objects created by a given `ServerManager` share the same network address, but each may communicate with a different client.

Notes

::Italy::ServerManager::create_session

Synopsis

```
void Italy_ServerManager_create_session
     ( Italy_ServerManager                  o,
     /* in */ Italy_Server                  server,
     /* out */ Italy_ServerSession          * session,
     CORBA_Environment                      * ev ) ;
```

Arguments

`server` (in) the applications `Italy::Server` to be associated with the new `ServerSession`

`session` (out) the newly-created ServerSession

Returns

`void`

Errors

Description

Creates a new `ServerSession` object. The application should configure the new object, then use `ServerSession::listen` to wait for client connections.

Notes

::Italy::Server::release

Synopsis

```
void Italy_Server_release
     ( Italy_Server                o,
     CORBA_Environment             * ev ) ;
```

Arguments

Returns

```
void
```

Errors

Description

This operation is invoked by the `Italy` library to notify the application that a client has requested release of the session.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::Server::reset

Synopsis

```
void Italy_Server_reset
     ( Italy_Server           o,
     CORBA_Environment        * ev ) ;
```

Arguments

Returns

```
void
```

Errors

Description

This operation is used to notify the server application that all sessions are being cleared.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::ServerSession

Synopsis

```
typedef CORBA_Object          Italy_ServerSession ;
```

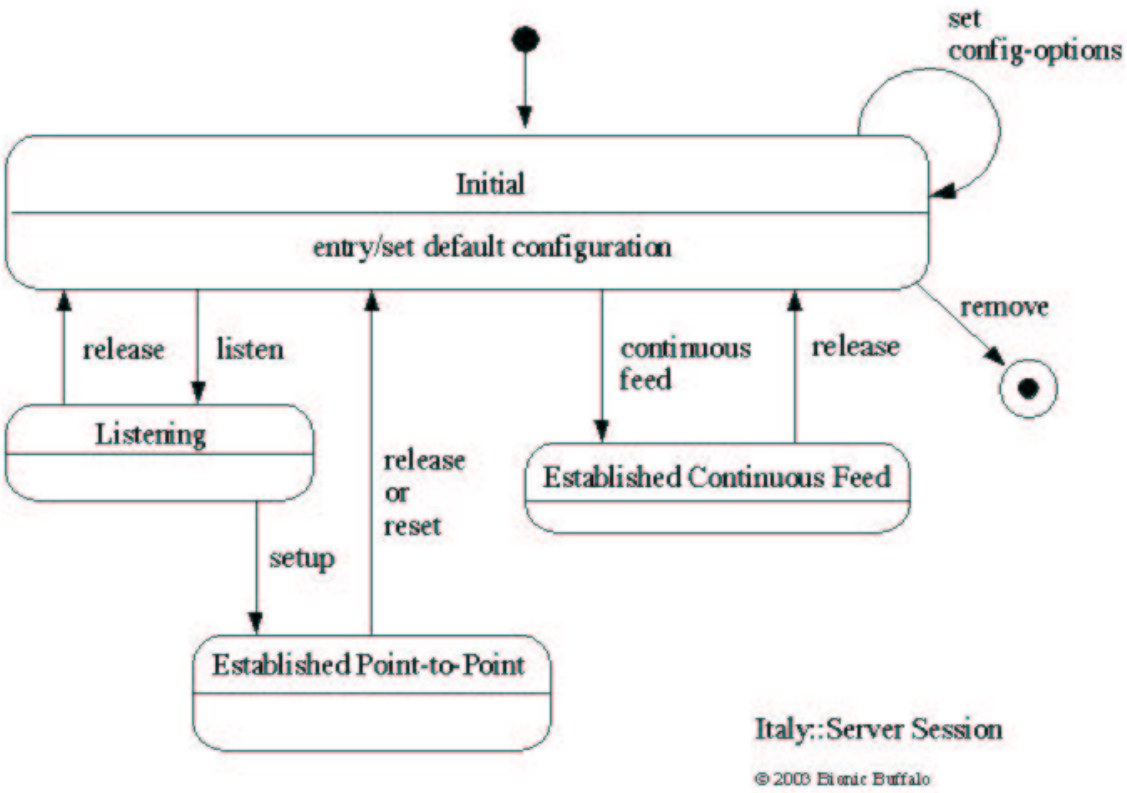
Description

This class represents a session on the server. It inherits `Italy::Session`..

Notes

1. An application should instantiate as many server objects as are needed for the number of simultaneous sessions permitted.

2. ::Italy::ServerSession Diagram on following page (31)



::Italy::ServerSession::add_resources

Synopsis

```
void Italy_ServerSession_add_resources
     ( Italy_ServerSession              o,
     /* in */ Malta_ResourceList        * resources,
     /* in-out */ CORBA_OctetSeq        * uu_data,
     /* in-out */ CORBA_OctetSeq        * private_data,
     CORBA_Environment                  * ev ) ;
```

Arguments

`resources`	(in) resources to be added
`uu_data`	(in) user-user data to be sent to the client
	(out) user-user data returned by the client
`private_data`	(in) private data to be sent to the client
	(out) private data returned by the client

Returns

Errors

Description

The server application uses this operation to add resources to a session.

Notes

::Italy::ServerSession::continuous_feed

Synopsis

```
void Italy_ServerSession_continuous_feed
     ( Italy_ServerSession                  o,
     /* in-out */ Malta_ResourceList        * resources,
     CORBA_Environment                      * ev ) ;
```

Arguments

`resources` *(in) the resources requested for the sessions*

(out) the resources actually assigned to the session

Returns

`void`

Errors

Description

This operation is used by a server application to establish a continuous feed session

Notes

1. This is an alternative to using `ServerSession::listen` to establish point-to-point sessions
2. All of the clients share the resources of a given continuous feed session

::Italy::ServerSession::delete_resources

Synopsis

```
void Italy_ServerSession_delete_resources
     ( Italy_ServerSession                    o,
     /* in */ Malta_ResourceList              * resources,
     /* in-out */ CORBA_OctetSeq              * uu_data,
     /* in-out */ CORBA_OctetSeq              * private_data,
     CORBA_Environment                        * ev ) ;
```

Arguments

`resources`	(in) the resources to be deleted
`uu_data`	(in) user-user data to be passed to the client
	(out) user-user data returned by the client
`private_data`	(in) private data to be passed to the client
	(out) private data returned by the client

Returns

Errors

Description

This operation is called by the server application to delete resources from a session.

Notes

::Italy::ServerSession::listen

Synopsis

```
void Italy_ServerSession_listen
     ( Italy_ServerSession              o,
     CORBA_Environment                  * ev ) ;
```

Arguments

Returns

`void`

Errors

Description

This operation puts the `ServerSession` into listening state, wherein it waits for connection requests from clients. It is called after the `ServerSession` has been configured.

Notes

::Italy::ServerSession::transfer

Synopsis

```
void Italy_ServerSession_transfer
     ( Italy_ServerSession              o,
     /* in */ DSM_ServiceDomain         * dest_server,
     /* in */ DSM_ServiceDomain         * base_server,
     /* in-out */ CORBA_OctetSeq        * uu_data,
     /* in-out */ CORBA_OctetSeq        * private_data,
     CORBA_Environment                  * ev ) ;
```

Arguments

`dest_server`	(in) the server to which the session is to be transferred
`base_server`	(in) the server which originally owned this session
`uu_data`	(in) user-user data to be sent to client
	(out) user-user data returned by client
`private_data`	(in) private data to be sent to the client
	(out) private data returned by the client

Returns

Errors

Description

This operation is used when the server application wants to transfer the session to another server.

Notes

::Italy::Server::setup

Synopsis

```
void Italy_Server_setup
     ( Italy_Server                    o,
     /* in-out */ CORBA_OctetSeq       * uu_data,
     /* in-out */ CORBA_OctetSeq       * private_data,
     /* out */ Malta_ResourceList      ** resources,
     CORBA_Environment                 * ev ) ;
```

Arguments

`uu_data`	(in) user-user data from the client
	(out) user-user data to be returned to the client
`private_data`	(in) private data from the client
	(out) private data to be returned to the client
`resources`	(out) the resources to be used in the session

Returns

`void`

Errors

Description

The `Italy` library calls this routine to notify the server application that a client has attempted to connect to a listening session.

Notes

1. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::Server::transfer

Synopsis

```
void Italy_Server_transfer
     ( Italy_Server                    o,
     /* in */ DSM_ServiceDomain        * source_server,
     /* in */ DSM_ServiceDomain        * base_server,
     /* in-out */ Malta_ResourceList   * resources,
     /* in-out */ CORBA_OctetSeq       * uu_data,
     /* in-out */ CORBA_OctetSeq       * private_data,
     CORBA_Environment                 * ev ) ;
```

Arguments

`source_server`	(in) the server from which the session is being transferred
`base_server`	(in) the first server to have owned the session
`resources`	(in) resources being transferred to the new server
	(out) resources used in this session

Returns

Errors

Description

This operation is called by the `Italy` library to notify a server application that another server has requested a session to be transferred to this server. The session must be in the listening state, and this operation may be called in lieu of a call to `Server::setup`.

Notes

1. The outgoing resources parameter includes those resources transferred through the incoming resources parameter, plus any newly-negotiated resources to be used in the session.

2. `Italy` creates a new thread to call this operation. The thread is terminated after the method returns.

::Italy::Session

Synopsis

```
typedef CORBA_Object          Italy_Session ;
```

Description

This abstract interface is inherited by `Italy::ClientSession` and by `Italy::ServerSession.` It defines operations and attributes common to both session types.

Notes

::Italy::Session::client_id

Synopsis

```
DSM_ServiceDomain             * Italy_Session__get_client_id
     ( Italy_Session          o,
     CORBA_Environment        * ev );

void Italy_Session__set_client_id

     ( Italy_Session          o,
     DSM_ServiceDomain        * client_id,
     CORBA_Environment        * ev );
```

Arguments

`client_id` (set) the client NSAPA

(get) the client NSAPA

Returns

Errors

Description

The client NSAP address used in the session.

Notes

1. This attribute cannot be set for server sessions.

::Italy::Session::config

Synopsis

```
void Italy_Session_config
     ( Italy_Session              o,
     CORBA_Environment            * ev ) ;
```

Arguments

Returns

```
void
```

Errors

Description

Used to request the network to configure the client or server. This activates an exchange using the User-Network config protocol.

Notes

::Italy::Session::config_options

Synopsis

```
Italy_ConfigOptions Italy_Session__get_config_options
     ( Italy_Session                         o,
     CORBA_Environment                       * ev );

void Italy_Session__set_config_options
     ( Italy_Session                         o,
     Italy_ConfigOptions                     * config_options,
     CORBA_Environment                       * ev );
```

Arguments

`config_options`	(set) configuration parameters
	(get) configuration parameters

Returns

Errors

Description

Interrogates or sets various configuration parameters for a session.

Notes

1. Some configuration parameters may be set by the user-network config protocol.
2. Normally, the application will `get` the parameters, modify those that are to be changed, then `set` the modified values.

::Italy::Session::private_data

Synopsis

```
CORBA_OctetSeq                * Italy_Session__get_private_data
     ( Italy_Session                          o,
     CORBA_Environment                        * ev );

void Italy_Session__set_private_data
     ( Italy_Session                          o,
     CORBA_OctetSeq                           * private_data,
     CORBA_Environment                        * ev );
```

Arguments

Returns

Errors

Description

Notes

::Italy::Session::release

Synopsis

```
void Italy_Session_release
     ( Italy_Session              o,
     CORBA_Environment            * ev ) ;
```

Arguments

Returns

Errors

Description

This operation requests termination of the session.

Notes

::Italy::Session::remove

Synopsis

```
void Italy_Session_remove
     ( Italy_Session              o,
     CORBA_Environment            * ev ) ;
```

Arguments

Returns

Errors

Description

This operation destroys the `ServerSession` or `ClientSession` object.

Notes

::Italy::Session::reset

Synopsis

```
void Italy_Session_reset
     ( Italy_Session              o,
     CORBA_Environment            * ev ) ;
```

Arguments

Returns

```
void
```

Errors

Description

This operation requests all sessions to be released.

Notes

::Italy::Session::resources

Synopsis

```
Malta_ResourceList              * Italy_Session__get_resources
     ( Italy_Session            o,
     CORBA_Environment          * ev );

void Italy_Session__set_resources
     ( Italy_Session            o,
     Malta_ResourceList         * resources,
     CORBA_Environment          * ev );
```

Arguments

Returns

Errors

Description

Notes

::Italy::Session::server_id

Synopsis

```
DSM_ServiceDomain             * Italy_Session__get_server_id
     ( Italy_Session          o,
     CORBA_Environment        * ev );

void Italy_Session__set_server_id
     ( Italy_Session          o,
     DSM_ServiceDomain        * server_id,
     CORBA_Environment        * ev );
```

Arguments

`server_id`	(set) the server NSAPA
	(get) the server NSAPA

Returns

Errors

Description

The server NSAP address used in the session.

Notes

::Italy::Session::srm_id

Synopsis

```
DSM_ServiceDomain                   * Italy_Session__get_srm_id
     ( Italy_Session                o,
     CORBA_Environment              * ev );

void Italy_Session__set_srm_id
     ( Italy_Session                o,
     DSM_ServiceDomain              * srm_id,
     CORBA_Environment              * ev );
```

Arguments

`srm_id`	(set) the SRM NSAPA
	(get) the SRM NSAPA

Returns

Errors

Description

The client NSAP address used in the session.

Notes

::Italy::SessionUU

Synopsis

```
typedef CORBA_Object          Italy_SessionUU ;
```

Description

This interface inherits `DSM::SessionUU`, and is used to extend the definition of the base interface.

Notes

::Italy::Session::uu_data

Synopsis

```
CORBA_OctetSeq                      * Italy_Session__get_uu_data
     ( Italy_Session                o,
     CORBA_Environment              * ev );

void Italy_Session__set_uu_data
     ( Italy_Session                o,
     CORBA_OctetSeq                 * uu_data,
     CORBA_Environment              * ev );
```

Arguments

Returns

`void`

Errors

Description

This operation releases the user-network session associated with the `SessionUU` object.

Notes

::Italy::SessionUU::release

Synopsis

```
void Italy_SessionUU_release
     ( Italy_SessionUU             o,
     CORBA_Environment             * ev ) ;
```

Arguments

Returns

Errors

Description

Notes

::Malta::ResourceList

Synopsis

```
typedef CORBA_sequence_any    Malta_ResourceList ;

typedef struct Malta_ResourceAttribute
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAllocator       allocator ;
     CORBA_boolean                 negotiable ;
     CORBA_boolean                 mandatory ;
     Malta_ResourceNumber          number ;
     Malta_AssociationTag          association_tag ;
     CORBA_unsigned_short          status ;
     CORBA_unsigned_short          request_id ;
     }
     Malta_ResourceAttribute ;


typedef struct Malta_Resource_ContinuousFeedSession
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_SessionId               session_id ;
     Malta_ResourceNumberSeq       resource_numbers ;
     }
     Malta_Resource_ContinuousFeedSession ;

typedef struct Malta_Resource_AtmConnection
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_AtmConnectionSeq        connections ;
     }
     Malta_Resource_AtmConnection ;

typedef struct Malta_Resource_MpegProgram
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_MpegProgramNumber       program_number ;
     Malta_MpegPid                 map_table_pid ;
     Malta_MpegPid                 conditional_access_pid ;
     Malta_StreamInfoSeq elementary_streams ;
     }
     Malta_Resource_MpegProgram ;

typedef struct Malta_Resource_PhysicalChannel
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_unsigned_short          direction ;
     CORBA_boolean                 sequence_is_range ;
```

```
     CORBA_ULongSeq               channel_ids ;
     }
     Malta_Resource_PhysicalChannel ;

typedef struct Malta_Resource_TransportStreamBandwidth
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
     CORBA_unsigned_short         direction ;
     CORBA_ULongSeq bandwidths ;
     CORBA_boolean                bandwidths_are_range ;
     CORBA_ULongSeq               transport_ids ;
     }
     Malta_Resource_TransportStreamBandwidth ;

typedef struct Malta_Resource_AtmSvcConnection
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
     CORBA_OctetSeq               setup_parameters ;
     }
     Malta_Resource_AtmSvcConnection ;

typedef struct Malta_Resource_ConnectionNotify
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
     CORBA_boolean                established ;
     }
     Malta_Resource_ConnectionNotify ;

typedef struct Malta_Resource_InternetProtocol
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
     CORBA_unsigned_long source_ip_address ;
     CORBA_unsigned_short         source_ip_port ;
     CORBA_unsigned_long destination_ip_address ;
     CORBA_unsigned_short         destination_ip_port ;
     CORBA_unsigned_short         protocol ;
     }
     Malta_Resource_InternetProtocol ;

typedef struct Malta_Resource_ClientTdmaAssignment
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
     Malta_TdmaAssignmentSeq      assignments ;
     CORBA_boolean                assignments_are_range ;
     }
     Malta_Resource_ClientTdmaAssignment ;

typedef struct Malta_Resource_PSTNSetup
     {
     tbt_ancillary_data_T         _tbt_ancillary ;
     Malta_ResourceAttribute      attribute ;
```

```
     CORBA_OctetSeq calling_id ;
     CORBA_OctetSeq called_id ;
     }
     Malta_Resource_PSTNSetup ;

typedef struct Malta_Resource_NISDNSetup
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_OctetSeq                setup_parameters ;
     }
     Malta_Resource_NISDNSetup ;

typedef struct Malta_Resource_NISDNConnection
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_unsigned_short          b_channel ;
     }
     Malta_Resource_NISDNConnection ;

typedef struct Malta_Resource_Q922Connections
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_Q922ConnectionSeq       connections ;
     }
     Malta_Resource_Q922Connections ;

typedef struct Malta_Resource_SharedResource
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_ResourceNumber          resource_number ;
     }
     Malta_Resource_SharedResource ;
```

```
typedef struct Malta_Resource_SharedRequestId
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_ResourceNumber          resource_id ;
     }
     Malta_Resource_SharedRequestId ;

typedef struct Malta_Resource_HeadEndList
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     Malta_HeadEndList             head_ends ;
     }
     Malta_Resource_HeadEndList ;

typedef struct Malta_Resource_AtmVcConnection
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
```

```
     Malta_ResourceAttribute       attribute ;
     Malta_AtmVpi                  vpi ;
     Malta_AtmVci                  vci ;
     }
     Malta_Resource_AtmVcConnection ;

typedef struct Malta_Resource_SdbContinuousFeed
     {
     tbt_ancillary_data_T              _tbt_ancillary ;
     Malta_ResourceAttribute           attribute ;
     Malta_SdbService                  service_id ;
     Malta_SdbProgramAssociationSeq    programs ;
     }
     Malta_Resource_SdbContinuousFeed ;

typedef struct Malta_Resource_SdbAssociations
     {
     tbt_ancillary_data_T              _tbt_ancillary ;
     Malta_ResourceAttribute           attribute ;
     Malta_AssociationTag              control_assoc_tag ;
     Malta_AssociationTag              program_assoc_tag ;
     }
     Malta_Resource_SdbAssociations ;

typedef struct Malta_Resource_SdbEntitlement
     {
     tbt_ancillary_data_T              _tbt_ancillary ;
     Malta_ResourceAttribute           attribute ;
     Malta_SdbService                  service_id ;
     CORBA_UShortSeq                   excluded_programs ;
     CORBA_UShortSeq                   included_programs ;
     }
     Malta_Resource_SdbEntitlement ;
```

Arguments

Returns

Errors

Description

Notes

::Malta::PegasusDescriptorList

Synopsis

```
typedef CORBA_sequence_any         Malta_PegasusDescriptorList ;

typedef struct Malta_Resource_Pegasus_AtscModulationMode
     {
     tbt_ancillary_data_T     _tbt_ancillary ;
     Malta_ResourceAttribute  attribute ;
     CORBA_octet              transmission_system ;
     CORBA_octet              inner_coding_mode ;
     CORBA_octet              split_bit_stream_mode ;
     CORBA_octet              modulation_format ;
     CORBA_unsigned_long symbol_rate ;
     CORBA_octet              interleave_depth ;
     CORBA_octet              modulation_mode ;
     CORBA_octet              forward_error_correction ;
     }
     Malta_Resource_Pegasus_AtscModulationMode ;

typedef struct Malta_Resource_Pegasus_HeadEndId
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_unsigned_short          flag ;
     DSM_ServiceDomain             head_end_id ;
     CORBA_unsigned_long           transport_stream_id ;
     }
     Malta_Resource_Pegasus_HeadEndId ;

typedef struct Malta_Resource_Pegasus_ServerConditionalAccess
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_sequence_unsigned_short ca_system_id_list ;
     CORBA_octet                   copy_protection ;
     Malta_ClientList              user_list ;
     }
     Malta_Resource_Pegasus_ServerConditionalAccess ;

typedef struct Malta_Resource_Pegasus_ClientConditionalAccess
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     Malta_ResourceAttribute       attribute ;
     CORBA_unsigned_short          ca_system_id ;
     CORBA_sequence_octet          ca_info ;
     }
     Malta_Resource_Pegasus_ClientConditionalAccess ;

typedef struct Malta_PegasusDescriptor_AssetId
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
```

```
     CORBA_sequence_octet          asset_id ;
     }
     Malta_PegasusDescriptor_AssetId ;

typedef struct Malta_PegasusDescriptor_NodeGroupId
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_sequence_octet          node_group_id ;
     }
     Malta_PegasusDescriptor_NodeGroupId ;

typedef struct Malta_PegasusDescriptor_InternetProtocol
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_unsigned_short          port ;
     CORBA_unsigned_long           address ;
     }
     Malta_PegasusDescriptor_InternetProtocol ;

typedef struct Malta_PegasusDescriptor_StreamHandle
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_unsigned_long           stream_handle ;
     }
     Malta_PegasusDescriptor_StreamHandle ;

typedef struct Malta_PegasusDescriptor_ApplicationRequestData
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_sequence_octet          data ;
     }
     Malta_PegasusDescriptor_ApplicationRequestData ;

typedef struct Malta_PegasusDescriptor_ApplicationResponseData
     {
     tbt_ancillary_data_T          _tbt_ancillary ;
     CORBA_sequence_octet          data ;
     }
     Malta_PegasusDescriptor_ApplicationResponseData ;
```

Arguments

Returns

Errors

Description

Notes

::Malta::PegasusServiceInfo

Synopsis

```
typedef struct
     {
     CORBA_OctetSeq                service_id ;
     Malta_PegasusDescriptorList   descriptors ;
     }
     Malta_PegasusServiceInfo ;

typedef struct Malta_PegasusServiceInfoSeq
     {
     unsigned long                 _length ;
     unsigned long                 _maximum ;
     Malta_PegasusServiceInfo      * _buffer ;
     }
     Malta_PegasusServiceInfoSeq ;
```

Description

`ServiceInfo` names, and provides the descriptors for, one Pegasus service. `ServiceInfoSeq` is a list of `ServiceInfo` for multiple services.

Notes

1. The `service_id` has a fixed length of 16 bytes.

mlt_pegasus_descriptors_marshal

Synopsis

```
void mlt_pegasus_descriptors_marshal
          ( unsigned                          protocol_id,
          unsigned                            protocol_version,
          CORBA_OctetSeq                     * service_gateway,
          Malta_PegasusServiceInfoSeq        * service_info,
          CORBA_OctetSeq                    ** private_data,
          CORBA_Environment                  * ev ) ;
```

Arguments

`protocol_id`	(in) Pegasus SSP protocol level
`protocol_version`	(in) Pegasus SSP protocol version
`service_gateway`	(in) name of service gateway (length is fixed 16 bytes)
`service_info`	(in) Pegasus descriptors
`private_data`	(out) descriptors formatted for private data section

Returns

`void`

Errors

Description

Converts Pegasus descriptor lists to a form usable in the user-network message private_data sections.

Notes

mlt_pegasus_descriptors_unmarshal

Synopsis

```
void mlt_pegasus_descriptors_unmarshal
          ( CORBA_OctetSeq                 * private_data,
          unsigned                         * protocol_id,
          unsigned                         * protocol_version,
          CORBA_OctetSeq                  ** service_gateway,
          Malta_PegasusServiceInfoSeq     ** service_info,
          CORBA_Environment                * ev ) ;
```

Arguments

`private_data`	(in) data from private part of User-Network Session messages
`protocol_id`	(out) Pegasus SSP protocol level
`protocol_version`	(out) Pegasus SSP protocol version
`service_gateway`	(out) name of service gateway (length is fixed 16 bytes)
`service_info`	(out) Pegasus descriptors

Returns

Errors

Description

Decodes the private data from an incoming User-Network message.

Notes

Alphabetical Index

Alphabetical Index - page 2