guatemala
DSM-CC Download Server
API Reference Manual

Bionic Buffalo Corporation
2003.09.23

Contents

Document Information and Copyright

Document Information and Copyright

Document Context

This document is a portion of the documentation for Bionic Buffalo's guatemala product. Information about the product and other, related documentation can be found at *http://www.tatanka.com/prod/guatemala.html*

Bionic Buffalo offers a family of products implementing DSM-CC and related protocols. For more information, see
http://www.tatanka.com/prod/dsmcc.html

Copyright, License, and Trademarks

Author and Publisher

Bionic Buffalo Corporation
502 North Division Street
Carson City, Nevada 89703
USA

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

PGP/GnuPG key fingerprint:

```
a836 e7b0 24ad 3259 7c38 b384 8804 5520 2c74 1e5a
```

Document History

Project name: *guatemala*
File name: *gtm_api_reference*
Formal revision date: Tuesday 2003.09.23
Last modified: 2003-09-24-14:35

For the latest version of this document, or for other forms of this document, see
http://www.tatanka.com/doc/man/guatemala/index.html

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

Introduction

Introduction

Bionic Buffalo's gibraltar product implements DSM-CC download servers as specified in *Extensions for DSM-CC* (ISO/IEC 13818-6), one of the MPEG-2 family of specifications.

The gibraltar product includes a library of software which can be used to implement download server applications. It also includes a server application (dnlserver) which can be used without additional programming.

This document describes the API available with the gibraltar library. For an overview of the entire product, refer to *DSM-CC Download Server Introduction*. For information on dnlserver, refer to *DSM-CC Download* dnlerver *User's Guide*.

Before reading the gibraltar manuals, it is suggested that you first read the *DSM-CC Download Common Introduction,* which is part of the angola product documentation. That document provides an overview of the download protocol and how it is used in applications.

Using the Library

The library is provided in shrouded C source code (“portable object code”) format, with a Makefile and documentation. Instructions for installation are included with the distribution.

Library routines may be called as functions from a C program, or from a program in other languages which support the same calling sequences.

An application using the gibraltar library calls gtm_install_download_server_mgr(), which returns a reference to an object of class Guatemala::ServerManager. This ServerManager object is then used to create one or more Guatemala::Server objects.

All of the Server objects created from a single ServerManager object share the same i/o addresses. Otherwise, they operate independently. If necessary, an application may create multiple ServerManager objects to allow operation of servers on multiple addresses.

The ServerManager and Server objects in a running application can be called in two ways:

- the application can call them directly, as subroutines
- another application (in the same machine, or over the network) can call them using the CORBA IIOP (the internet inter-ORB protocol)

The second method allows complete control of the download server from a remote location.

When the application requires asynchronous notification of certain events, it must create an object of class `Guatemala::Application` which is registered with the `ServerManager`. By using an object for this purpose (instead of a callback address), the `ServerManager` can notify remote applications.

Interface and Operation Descriptions

The rest of this document consists of descriptions of the interfaces and operations available from the `gibraltar` library. The format is traditional. Because of the two ways of accessing `ServerManager` and `Server` objects, the API descriptions in this document specify the IDL interface as well as the C interface to each class.

`gtm_install_download_server_mgr`

Name

`gtm_install_download_server_mgr` -- create an object of class `Guatemala::ServerManager`

Synopsis - C

```
#include <guatemala.h>

void    gtm_install_download_server_mgr
                ( gtm_server_parameters_T     * parameters,
                Guatemala_ServerManager       * server_manager,
                CORBA_Environment             * ev ) ;
```

Arguments

`parameters` -- *(in/out)* configuration parameters
`server_manager` -- *(out)* created `Guatemala::ServerManager` object
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This function creates an object of class `Guatemala::ServerManager` using the specified **`parameters`**.

Notes

1. This is the first `guatemala` function called by an application.

Example

Availability

guatemala (Bionic Buffalo)

See also

gtm_server_parameters_T

Reference

`gtm_server_parameters_T`

Name

`gtm_server_parameters_T` -- structure used to specify parameters for the installation of a download server manager

Synopsis - C

```
#include <guatemala.h>

typedef struct gtm_server_parameters_T
        {
        CORBA_ORB                       orb ;
        PortableServer_POA              poa ;
        CosNaming_NamingContext         naming_context ;
        unsigned char                 * server_manager_name ;
        mar_netlink_T                   channels [ 4 ] ;
        int                             sockets [ 4 ] ;
        size_t                          maximum_message_sizes [ 4 ] ;
        struct timespec                 timer_granularity ;
        unsigned long                   open_file_limit ;
        }
        gtm_server_parameters_T ;
```

Description

The `gtm_server_parameters_T` structure us used to pass arguments to, and return results from, an invocation of the `gtm_install_download_server_mgr ()` operation.

Notes

1. The **`orb`** and **`poa`** are used to create and register the `Guatemala::ServerManager` object.

2. If a **`naming_context`** and **`server_manager`**_name are defined, then the new `ServerManager` will be named accordingly in the designated context.

3. The subscript for **`channels`**, **`sockets`**, and **`maximum_message_sizes`** is one of the values of `Angola::Channel`. There may be one physical channel, but each of the four logical channel must be described separately. When it is necessary for the library to determine if two logical channels are mapped to the same physical channel, the software compares the values of the corresponding **`channels`** elements with each other.

4. If non-zero values are specified for **sockets**, then the sockets will be presumed open and configured. Otherwise, sockets will be created and configured according to **channels**, and returned to the caller for (optional) additional configuration. Regardless of whether or not the **sockets** are pre-opened, the **channels** must be specified.

5. The **maximum_message_sizes** represents the largest messages which might be sent or received on each of the four channels. It may be left zero for a default value.

6. **timer_granularity** may be left zero for a default value.

7. **open_file_limit** is the greatest number of files which may be open simultaneously. It may be left zero for a default value.

Example

Availability

guatemala (Bionic Buffalo)

See also

gtm_install_download_server_mgr

Reference

`Guatemala::Application`

Name

`Guatemala::Application` -- interface used by object belonging to an application

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface               Application ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

typedef CORBA_Object            Guatemala_Application ;
```

Description

An application creates a `Guatemala::Application` object when it wants to be notified asynchronously of events in the download server. The application registers the object by modifying the `Guatemala::ServerManager::application` attribute.

Notes

1. The `Application` interface has operations `complete`, `request`, and `server_created`.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Application::complete`

Guatemala::Application::request
Guatemala::Application::server_created
Guatemala::ServerManager::application

Reference

`Guatemala::Application::complete`

Name

`Guatemala::Application::complete` -- notifies an application that an operation is complete or was cancelled

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Application
        {
        void    complete
                    ( in Server                     server,
                    in Angola::CancelReason         reason,
                    in Angola::Statistics           statistics,
                    in Angola::ModuleSynopsisList   modules ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Application_complete
                ( Guatemala_Application         o,
                Guatemala_Server                server,
                Angola_CancelReason             reason,
                Angola_Statistics             * statistics,
                Angola_ModuleSynopsisList     * modules,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the application to be notified
`server` -- *(in)* the `Server` conducting the operation
`reason` -- *(in)* if zero, indicates that the operation completed normally; if nonzero, the reason for the operation cancellation
`statistics` -- *(in)* summary of operation activity
`modules` -- *(in)* the modules regarding which the `Application` is being notified
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation notifies an application that a download operation has completed, or was cancelled.

Notes

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Application`

Reference

`Guatemala::Application::request`

Name

`Guatemala::Application::request` -- notifies an application that a `DownloadInfoRequest` message has been received

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Application
        {
        void    request
                    ( in Server                     server,
                    in DSM::CompatibilityDescriptor compat_descr,
                    in CORBA::OctetSeq              private_data,
                    out Angola::CancelReason        cancel_reason ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Application_request
                ( Guatemala_Application         o,
                Guatemala_Server                server,
                DSM_CompatibilityDescriptor   * compat_descr,
                CORBA_sequence_octet          * private_data,
                Angola_CancelReason           * cancel_reason,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the application to be notified
`server` -- *(in)* the Server which received the request
`compat_descr` -- *(in)* the request's compatibility descriptor
`private_data` -- *(in)* the request's private data
`cancel_reason` -- *(out)* if nonzero, the reason to cancel the download operation
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation notifies an application that a `DownloadInfoRequest` message was received. It gives the application an opportunity to load the carousel, cancel the operation, or take other appropriate action.

Notes

1. If the application returns a nonzero value for **`cancel_reason`**, then the `Server` will cancel the operation for the reason given.

2. If the application wants to continue with the operation, it may invoke `Server::load` (to load the `Server`'s contents), or it may set the `Server::compatibility_descriptor` or `Server::private_data` attributes before returning.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Application`
`Guatemala::Server::compatibility_descriptor`
`Guatemala::Server::load`
`Guatemala::Server::private_data`

Reference

`Guatemala::Application::server_created`

Name

`Guatemala::Application::server_created` -- notifies an application that a Server has been created

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Application
        {
        void    server_created
                    ( in Server                     server ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Application_server_created
                ( Guatemala_Application         o,
                Guatemala_Server                server,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the application to be notified
`server` -- *(in)* the `Server` which was created
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation notifies an application that a new server was created.

Notes

1. `Server`s are created in response to `Client` connections on reliable networks. Each connection corresponds to a single `Server`.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Application`
`Guatemala::Server`

Reference

`Guatemala::Server`

Name

`Guatemala::Server` -- class of objects implementing a Download Server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface               Server ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

typedef CORBA_Object            Guatemala_Server ;
```

Description

A `Guatemala::Server` object implements the Download Server defined by the DSM-CC specification. Each `Server` object has a separate state, and implements a separate state machine.

Notes

1. `Server` objects are created by `ServerManager` objects, using the `create_server` operation.

2. The `Server` interface has the `session_parameters`, `compatibility_descriptor`, and `private_data` attributes. Operations are `activate`, `deactivate`, `load`, `subset_add`, `subset_delete`, and `destroy`.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

Guatemala::ServerManager::create_server
Guatemala::Server::activate
Guatemala::Server::compatibility_descriptor
Guatemala::Server::deactivate
Guatemala::Server::destroy
Guatemala::Server::load
Guatemala::Server::private_data
Guatemala::Server::session_parameters
Guatemala::Server::subset_add
Guatemala::Server::subset_delete

Reference

`Guatemala::Server::activate`

Name

`Guatemala::Server::activate` -- start a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface Server
                {
                void    activate ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_activate
                ( Guatemala_Server              o,
                CORBA_Environment             * ev ) ;
```

Arguments

o -- *(in)* the server to be activated
ev -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation activates a `Guatemala::Servant`.

Notes

3. Before invoking this operation, an application should set the `Server::session_parameters` attribute, and set the server contents using the `Server::load` operation.

4. Once a `Server` is activated, its initial action will depend on the `scenario` specified in the `session_parameters`. In the flow-controlled scenario, the `Server` will wait for a `DownloadInfoRequest` message from a client before commencement of operation. In the non-flow-controlled and carousel scenarios, the `Server` will commence operations without waiting.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Server::load`
`Guatemala::Server::session_parameters`

Reference

`Guatemala::Server::compatibility_descriptor`

Name

`Guatemala::Server::compatibility_descriptor` -- the `DSM::CompatibilityDescriptor` used in `DownloadInfoIndication` and `DownloadInfoResponse` messages

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        attribute DSM::CompatibilityDescriptor  compatibility_descriptor ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

DSM_CompatibilityDescriptor
      * Guatemala_Server__get_compatibility_descriptor
                ( Guatemala_Server              o,
                CORBA_Environment             * ev ) ;
void    Guatemala_Server__set_compatibility_descriptor
                ( Guatemala_Server              o,
                DSM_CompatibilityDescriptor   * compatibility_descriptor,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server to which the attribute applies
`compatibility_descriptor` -- *(`set` only) (in)* new or initial value for the compatibility descriptor
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current value of the compatibility descriptor

Exceptions

Description

This attribute specifies the compatibility descriptor sent in `DownloadInfoIndication` and `DownloadInfoResponse` messages.

Notes

1. This is *not* the compatibility descriptor received from clients in the `DownloadInfoRequest` messages. That value is passed to the `Application` by means of the `Application::request` operation.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

Reference

`Guatemala::Server::deactivate`

Name

`Guatemala::Server::deactivate` -- terminate a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface Server
                {
                void    deactivate ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_deactivate
                ( Guatemala_Server              o,
                CORBA_Environment             * ev ) ;
```

Arguments

o -- *(in)* the server to be deactivated
ev -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation terminates the operation of a `Guatemala::Server`. The `Server` may be reactivated again. The `Server` remains in existence until it is destroyed.

Notes

Example

Availability

guatemala (Bionic Buffalo)

See also

Guatemala::Server::activate
Guatemala::Server::destroy

Reference

`Guatemala::Server::destroy`

Name

`Guatemala::Server::destroy` -- destroy a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface Server
                {
                void    destroy ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_destroy
                ( Guatemala_Server              o,
                CORBA_Environment             * ev ) ;
```

Arguments

o -- *(in)* the server to be destroyed
ev -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation destroys a `Guatemala::Server` object. References to the `Server` are no longer valid.

Notes

Example

Availability

guatemala (Bionic Buffalo)

See also

Reference

`Guatemala::Server::load`

Name

`Guatemala::Server::load` -- load contents into a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        void load
            ( in string                     directory,
            in Angola::ModuleSynopsisList   contents,
            in boolean                      suppress_master_directory ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_load
                ( Guatemala_Server              o,
                CORBA_string                    directory,
                Angola_ModuleSynopsisList     * contents,
                CORBA_boolean                   suppress_master_directory,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server into which content is to be loaded
`directory` -- *(in)* pathname to directory containing files to be loaded
`contents` -- *(in)* descriptions of modules to be loaded
`suppress_master_directory` -- *(in)* for carousels, suppress `DownloadInfoIndication` messages containing a master directory
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation loads content into a `Guatemala::Server`. The **`contents`** are a sequence of `ModuleSynopsis`, each of which describes one module. All of the modules in the list become available for download by clients. Modules already loaded into the `Server` are removed and completely replaced.

Notes

1. The **`directory`** is the base directory used to locate modules whose source is specified by path name.

2. In carousel scenarios, if **`suppress_master_directory`** is `TRUE`, then there will not be a `DownloadInfoIndication` message sent to describe the complete contents of the carousel. The only `DownloadInfoIndication` messages sent will be those resulting from `subset_add` operations.

3. If the `Server` is sending content already loaded, the scenario will be cancelled, abruptly terminating any transmissions in progress. This is true even when the new content is the same as the old content. For a more graceful transition, use `Guatemala::Server::update`.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Server::subset_add`
`Guatemala::Server::update`

Reference

`Guatemala::Server::private_data`

Name

`Guatemala::Server::private_data` -- the private data used in `DownloadInfoIndication` and `DownloadInfoResponse` messages

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        attribute CORBA::OctetSeq           private_data ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

CORBA::OctetSeq
      * Guatemala_Server__get_private_data
                ( Guatemala_Server              o,
                CORBA_Environment             * ev ) ;
void    Guatemala_Server__set_private_data
                ( Guatemala_Server              o,
                CORBA_OctetSeq                * private_data,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server to which the attribute applies
`private_data` -- *(`set` only) (in)* new or initial value for the private data
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current value of the private data

Exceptions

Description

This attribute specifies the private data sent in `DownloadInfoIndication` and `DownloadInfoResponse` messages.

Notes

1. This is *not* the private data received from clients in the `DownloadInfoRequest` messages. That value is passed to the `Application` by means of the `Application::request` operation.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

Reference

`Guatemala::Server::session_parameters`

Name

`Guatemala::Server::session_parameters` -- parameters for a download session

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        attribute Angola::SessionParameters session_parameters ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

Angola_SessionParameters
        Guatemala_Server__get_session_parameters
                ( Guatemala_Server          o,
                CORBA_Environment         * ev ) ;
void    Guatemala_Server__set_session_parameters
                ( Guatemala_Server          o,
                Angola_SessionParameters  * session_parameters,
                CORBA_Environment         * ev ) ;
```

Arguments

`o` -- *(in)* the server to which the attribute applies
`session_parameters` -- *(`set` only) (in)* new or initial parameters for the `Session`
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current `Session` parameters

Exceptions

Description

This attribute specifies the scenario (flow-controlled download, non-flow-controlled download, or carousel), window size, block size, and timing parameters for the Session.

Notes

1. Not all size and timing parameters are used for all scenarios. Applicability is indicated in the following table:

Parameter	*Flow controlled download*	*Non flow controlled download*	*Carousel*
`blockSize`	yes	yes	yes
`windowSize`	yes	no	no
`ackPeriod`	yes	no	no
`tCDownloadWindow`	yes	no	no

2. In the flow-controlled scenario, the parameters specified for the server may not be the parameters used in the session. The server negotiates a set of session parameters based on its own limits, and on the limits specified by the client in the client's `DownloadInfoRequest` message.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

Reference

`Guatemala::Server::subset_add`

Name

`Guatemala::Server::subset_add` -- define a subset of a complex carousel

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        void subset_add
            ( in Angola::ModuleSynopsisList     subset,
            in DSM::CompatibilityDescriptor     compat_descr,
            in CORBA::OctetSeq                  private_data,
            inout unsigned long                 subset_handle ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_subset_add
                ( Guatemala_Server              o,
                Angola_ModuleSynopsisList     * subset,
                DSM_CompatibilityDescriptor   * compat_descr,
                CORBA_sequence_octet          * private_data,
                CORBA_unsigned_long           * subset_handle,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server into which content is to be loaded
`subset` -- *(in)* the modules contained within the subset
`compat_descr` -- *(in)* the subset's compatibility descriptor
`private_data` -- *(in)* the subset's private data
`subset_handle` -- *(in/out)* a handle to identify the subset
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation defines a subset of a carousel's contents. The subset causes the transmission of a distinct `DownloadInfoIndication` record. The use of subsets allows various clients to select from among the modules in a single carousel.

Notes

1. This operations does not load contents into the carousel. It only defines a subset of contents already loaded.

2. The `reference` members of the **`subset`** elements are ignored, since the module's source was already defined by the `load` operation.

3. The `moduleId` and `moduleVersion` of each **`subset`** element must correspond to a `moduleId` and `moduleVersion` of a module defined by the contents of a `load` operation.

4. If a new subset is being defined, then the `subset_handle` should be set to zero, and a nonzero value will be returned. The returned `subset_handle` may be used in a subsequent `subset_delete` operation, or in a different `subset_add` operation to redefine the same subset.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Server::subset_delete`

Reference

`Guatemala::Server::subset_delete`

Name

`Guatemala::Server::subset_delete` -- remove a subset definition from a carousel

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        void subset_delete
            ( in unsigned long                  subset_handle ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_load
                ( Guatemala_Server              o,
                CORBA_unsigned_long             subset_handle,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server into which content is to be loaded
`subset_handle` -- *(in)* a handle identifying the subset to be deleted
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation deletes the definition of a carousel subset.

Notes

1. The **subset_handle** must have come from a previous subset_add operation.

Example

Availability

guatemala (Bionic Buffalo)

See also

Guatemala::Server::subset_add

Reference

`Guatemala::Server::update`

Name

`Guatemala::Server::update` -- modify the contents of a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface Server
        {
        void update
            ( in string                     directory,
            in Angola::ModuleSynopsisList   update,
            in boolean                      wait_complete,
            in boolean                      suppress_master_directory ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

void    Guatemala_Server_load
                ( Guatemala_Server              o,
                CORBA_string                    directory,
                Angola_ModuleSynopsisList     * update,
                CORBA_boolean                   wait_complete,
                CORBA_boolean                   suppress_master_directory,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the server into which content is to be loaded
`directory` -- *(in)* pathname to directory containing files to be loaded
`update` -- *(in)* descriptions of modules to be added, modified, or removed
`wait_complete` -- *(in)* allow current download operations to complete
`suppress_master_directory` -- *(in)* for carousels, suppress `DownloadInfoIndication` messages containing a master directory
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation modifies the contents of a `Guatemala::Server`. The **`update`** is a sequence of `ModuleSynopsis`, each of which describes one module. The existing contents of the server are compared against the update, by matching `moduleId` values.

Any modules already in the contents, but not found in the **`update`**, are retained in the contents without change. Any modules found in the **`update`**, but not already in the contents, are added to the contents. Any modules found in both, with the module's `Angola_ModuleSynopsis_DELETE` flag set in the **`update`**, are removed from the contents. Any other modules found in both (without the `DELETE` flag set) cause the contents to be modified, without removing the module.

Notes

1. *(Flow-controlled and non-flow-controlled scenarios only.)* The `Server::update` operation takes place atomically: all additions, modifications, and deletions take place at once. If **`wait_complete`** is `TRUE`, then any scenarios in progress are allowed to complete, but any new scenarios will use the updated image. In such cases, it may be possible for downloading of old and new images to different clients to overlap. The `Angola_ModuleSynopsis_WAIT_COMPLETE` flags for individual modules are ignored in these scenarios.

2. *(Carousel scenarios only.)* If a module is being modified or deleted, and if the module's `Angola_ModuleSynopsis_WAIT_COMPLETE` flag is set in **`update`**, then the modification or deletion will not take place until any current download transmissions have completed. If no such transmissions are currently taking place, then the modification or deletion will take place immediately. Setting the **`wait_complete`** argument `TRUE` has the same effect as setting the individual `Angola_ModuleSynopsis_WAIT_COMPLETE` flags for all modules.

3. *(Carousel scenarios only.)* There is nothing in the protocol to prevent a client from beginning the download of a module being sent, but due to be replaced or deleted, and there is no way to

notify the client that such a deletion or substitution is pending. Such an attempt will fail, regardless of the state of the module's `Angola_ModuleSynopsis_WAIT_COMPLETE` flag.

4. The **`directory`** is the base directory used to locate modules whose source is specified by path name.

5. In carousel scenarios, if **`suppress_master_directory`** is `TRUE`, then there will not be a `DownloadInfoIndication` message sent to describe the complete contents of the carousel. The only `DownloadInfoIndication` messages sent will be those resulting from `subset_add` operations.

6. To completely replace the contents of the server, abruptly terminating any download operations in progress, use `Guatemala::Server::load`.

7. The `Server` will automatically update the `moduleVersion` of each module to be modified, when the content modification time (`st_mtime`) has changed, or when the length of the file (`st_size`) has changed. The `moduleVersion` will not be automatically updated solely on account of a changed `moduleInfo` value. The application may force an update of `moduleVersion` by setting the module's `Angola_ModuleSynopsis_UPDATE_VERSION` flag.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Server::subset_add`
`Guatemala::Server::load`

Reference

`Guatemala::ServerManager`

Name

`Guatemala::ServerManager` -- class of objects which can create `Server` objects

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
        {
        interface               ServerManager ;
        } ;
```

Synopsis - C

```
#include <guatemala.h>

typedef CORBA_Object            Guatemala_ServerManager ;
```

Description

A `Guatemala::ServerManager` creates `Guatemala::Server` objects. The `Server` objects created by a `ServerManager` share the same i/o addresses. The `ServerManager` interface includes the `application` attribute, and the `create_server` operation.

Notes

1. A `ServerManager` object is returned by the function `gtm_install_download_server_mgr`.

2. Each `ServerManager` object may be associated with zero or one `Guatemala::Application` objects. The association is specified by the `ServerManager::application` attribute.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

gtm_install_download_server_mgr
Guatemala::ServerManager::application
Guatemala::ServerManager::create_server

Reference

`Guatemala::ServerManager::application`

Name

`Guatemala::ServerManager::application` -- the `Guatemala::Application` object associated with a `ServerManager`

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface ServerManager
        {
        attribute Application    application ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

Guatemala_Application
        Guatemala_ServerManager__get_application
                ( Guatemala_ServerManager           o,
                CORBA_Environment                 * ev ) ;
void    Guatemala_ServerManager__set_application
                ( Guatemala_ServerManager           o,
                Guatemala_Application               application,
                CORBA_Environment                 * ev ) ;
```

Arguments

`o` -- *(in)* the `ServerManager` with which the `Application` is associated
`application` -- *(`set` only) (in)* the `Application` to associate with the `ServerManager`
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) the `Application` associated with the `ServerManager`

Exceptions

Description

This attribute specifies the `Guatemala::Application` associated with the `ServerManager`.

Notes

1. The `Application` object is created by the application which uses the `guatemala` library, `ServerManager`, or `Server` objects.

Example

Availability

`guatemala` (Bionic Buffalo)

See also

`Guatemala::Application`

Reference

`Guatemala::ServerManager::create_server`

Name

`Guatemala::ServerManager::create_server` -- create a download server

Synopsis - IDL

```
#include <guatemala.idl>

module Guatemala
    {
    interface ServerManager
        {
        Server  create_server ( ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <guatemala.h>

Guatemala_Server
....    Guatemala_ServerManager_create_server
                ( Guatemala_ServerManager       o,
                CORBA_Environment             * ev ) ;
```

Arguments

o -- *(in)* the `ServerManager` to create the `Server`
ev -- *(in/out)* CORBA environment

Returns

the new `Server` object

Exceptions

Description

This operation creates a new `Guatemala::Server` object.

Notes

Example

Availability

guatemala (Bionic Buffalo)

See also

Guatemala::Server

Reference