WANSync® Control Library API™

WANSync Control Library API Reference Guide

Change History

- 6/10/05
  - Added information regarding Suspend/Resume Replica to the appropriate sections of the guide.
  - Revised *Source files*.
  - Added two new methods to *Class methods for class ws_statistics_c*.
- 8/31/05
  - Added new note format.
  - Added date column to Revision History table.
- 7/15/06
  - Editorial changes.
- 10/15/06
  New software release.
- 11/18/06
  - New software release.

Contents

Chapter 1: Getting Started

Source Files 1
Classes and Data Structures 2

Chapter 2: Using the Library

Connect to a Master Scenario 3
Traverse the Master Scenario List 3
Monitor and Manage a Master Scenario 4
Disconnect from a Master Scenario 4
Collect Statistical Information. 4
Other Uses for ws_control_c Object 4

Chapter 3: Using the Library

XO Error Codes. 5
XO Events 7
    Event Types. 7
    Event IDs 8
    enum ws_sync_method_e. 9
enum ws_report_type_e 9
struct ws_directory_statistics_s 10
class ws_statistics_c 11
    Class Methods for class ws_statistics_c. 11
class ws_scenario_c 14
    Class Methods for class ws_scenario_c 14
class ws_control_c 22
    Class methods for class ws_control_c 22

Index

Chapter 1: Getting Started

The WANSync Control Library provides an easy to use API that allows you to monitor and manage the replication process. Through the WANSync Control Library, an application can connect to the XOsoft Engine running on a given host:

- To collect statistical information of all existing scenarios (master and replica)
- To retrieve the list of existing master scenarios -- the application can then traverse this list and control each of these scenarios

Controlling individual master scenarios is enabled by the following procedures:

- Start and stop the replication scenario
- Initiate the synchronization process
- Enable/disable logging, and retrieve log files
- Collect scenario and root directory statistics
- Create and retrieve a difference report
- Collect synchronization/replication report files
- Receive scenario events
- Suspend/Resume replica

Source Files

The source files included with the installation are:

- **xo_base.h** - Declaration of base types and error codes
- **ws_control.h** - Declaration of WANSync Control Library classes and structures
- **sample.h** - Sample of using WANSync Control Library
- **sample.cpp**- Sample of using WANSync Control Library

Classes and Data Structures

The classes and data structures are:

- **class ws_control_c** - Used for:
  - Collecting statistical information of all existing scenarios
  - Retrieving the list of existing master scenarios on XOsoftEngines
- **class ws_scenario_c** - Manages a specific master scenario
- **class ws_statistics_c** - Holds scenario statistics
- **struct ws_directory_statistics_s** - Holds root directory statistics
- **enum ws_sync_method_e** - Defines the synchronization method
- **enum ws_report_type_e** - Defines the various types of WANSync system reports

Chapter 2: Using the Library

The following sections provide an overview of the library's main objects, their uses, and their related methods.

Connect to a Master Scenario

To establish a connection to an XOsoftEngine service, create an instance of ws_control_c and invoke ws_control_c::attach_to_scenarios(), with the required host and port pair.

If the connection succeeds, a list of ws_scenario_c objects is created, one for each existing master scenario. This list is internally stored and shared by all ws_control_c objects. Invoking additional calls to ws_control_c::attach_to_scenarios() with different host and port pair will append new ws_scenario_c objects to the original list.

Traverse the Master Scenario List

In order to traverse the master scenario objects, use the ws_control_c::first_scenario() and ws_control_c::next_scenario() methods. Each of these methods returns a pointer to a ws_scenario_c object. Use this pointer to manage the corresponding master scenario.

Monitor and Manage a Master Scenario

To monitor and manage a master scenario, use the ws_scenario_c object methods which allow the following:

- Start and stop the replication scenario
- Initiate the synchronization process
- Enable/disable logging, and retrieve log files
- Collect scenario and root directory statistics
- Create and retrieve a difference report
- Collect synchronization/replication report files
- Receive scenario events

Disconnect from a Master Scenario

Disposal of the ws_scenario_c objects is done by calling the ws_control_c::detach_from_scenarios() method, and not by deleting the ws_scenario_c object directly.

Note, that since the list of ws_scenario_c objects is shared by all ws_control_c objects, the disposal of the ws_scenario_c objects is done only after calling ws_control_c:: detach_from_scenarios() from all created ws_control_c objects.

Collect Statistical Information

To collect statistical information of all existing scenarios (master and replica) on a running XOsoft Engine service, create an instance of ws_control_c and invoke ws_control_c::get_host_statistics(), with the required host and port pair. If this function succeeds, it initializes an array of ws_statistics_c objects, each for an existing scenario (master or replica) on the XOsoft Engine running on a given host.

Other Uses for ws_control_c Object

The ws_control_c object is also used for registering a new license key and setting the WANSync Super User group. In order to invoke each of the latter methods, it is mandatory to first dispose of the ws_scenario_c objects by calling ws_control_c::detach_from_scenarios().

Chapter 3: Using the Library

This chapter describes the library codes and class methods in detail.

XO Error Codes

The following table provides a list of possible XOSTATUS codes returned by many of the library's methods.

Code	Description	Name
0	The operation completed successfully.	XO_ERROR_OK
1	Incorrect library object is used.	XO_ERROR_INVALID_OBJECT
2	The parameter is incorrect.	XO_ERROR_INVALID_PARAMETER
3	The data area passed to a library call is too small.	XO_ERROR_INSUFFICIENT_BUFFER
4	No more data is available.	XO_ERROR_NO_MORE_DATA
5	The library cannot uncompress the file.	XO_ERROR_UNCOMPRESS_FILE_FAILED
6	There is a mismatch between the library version and the version of the XOsoftEngine to which it tries to connect.	XO_ERROR_VERSION_MISMATCHING
7	Unable to connect to master host.	XO_ERROR_CONNECTION_FAILED
8	The connection was seized by another host.	XO_ERROR_CONNECTION_SEIZED
9	Connection with master host is lost.	XO_ERROR_CONNECTION_CLOSED
10	User is not authorized to manage the scenario.	XO_ERROR_NOT_AUTHORIZED
11	License is inappropriate. Use ws_control_c::register_license() to reset the license.	XO_ERROR_INAPPROPRIATE_LICENSE

Code	Description	Name
12	No license. Use ws_control_c::register_license() to set the license.	XO_ERROR_NO_LICENSE
13	License expired. Use ws_control_c::register_license() to reset the license.	XO_ERROR_LICENSE_EXPIRED
14	Unable to read scenario's XML configuration file.	XO_ERROR_READ_CONFIGURATION_FAILED
15	The XML configuration file of the scenario was not changed by the WANSync editor.	XO_ERROR_CONFIGURATION_CHANGED_ MANUALLY
16	Unable to check if the scenario configuration has changed.	XO_ERROR_VERIFY_CONFIGURATION _FAILED
17	Detected change in scenario configuration and trying to download new configuration.	XO_ERROR_DOWNLOAD_SCENARIO_IN_ PROGRESS
18	If this is a return value from ws_control_c::attach_to_scenarios(), the scenarios from the specific XosoftEngine could not be downloaded; otherwise, this means that a change in the scenario configuration has been detected, and an attempt to download it failed. When the application sends a new command, the download is retried.	XO_ERROR_DOWNLOAD_SCENARIO_FAILED
19	The command timeout period has expired.	XO_ERROR_TIMEOUT
20	Unable to send the command to the specific scenario.	XO_ERROR_SEND_CMD_FAILED
21	Unable to send the file to the specific scenario. (Enterprise Rewinder only).	XO_ERROR_SEND_FILE_FAILED
22	Unable to perform the requested operation. Check operation's pre-conditions.	XO_ERROR_OPERATION_FAILED
23	Illegal operation. Check pre-conditions.	XO_ERROR_OPERATION_ABORTED
24	Unable to complete the requested operation because the scenario has stopped.	XO_ERROR_SCENARIO_STOPPED
25	Unable to get the scenario state.	XO_ERROR_GET_SCENARIO_STATE_FAILED
26	Previous synchronization/verification process is still in progress.	XO_ERROR_SYNC_OR_DIFF_IN_PROGRESS
27	WANSync Super User Group not detected. Specify the Super User Group by invoking ws_control_c::set_su_group().	XO_ERROR_SU_GROUP_NOT_SPECIFIED

Code	Description	Name
28	WANSync Super User Group has been manually modified. Contact support team.	XO_ERROR_SU_GROUP_VIOLATED
29	No mapping between WANSync Super User Group Name and security ID was done. Make sure that you are logged into the domain in which the Super User Group is defined.	XO_ERROR_SU_GROUP_NOT_MAPPED
30	Unable to read WANSync Super User Group. Contact support team.	XO_ERROR_READ_SU_GROUP_FAILED
31	Invalid WANSync Super User Group was specified.	XO_ERROR_INAPPROPRIATE_SU_GROUP
32	Unable to manually resume replica which is in scheduled suspend.	XO_ERROR_SCHEDULED_SUSPEND
33	Unable to manually resume replica which is in scheduled offline processing.	XO_ERROR_SCHEDULED_OFFLINE_ PROCESSING
34	Unable to manually resume replica which is in scheduled DB verification.	XO_ERROR_SCHEDULED_DB_VERIFY
35	Unable to suspend replica because there is a suspended replica in this scenario.	XO_ERROR_SUSPEND_IN_PROGRESS
36	Unable to suspend replica because there is a replica in offline processing in this scenario. Unable to resume replica in offline processing.	XO_ERROR_OFFLINE_PROCESSING_IN_ PROGRESS
37	Unable to suspend replica because there is a replica in DB verification in this scenario. Unable to resume replica in DB verification.	XO_ERROR_DB_VERIFY_IN_PROGRESS

XO Events

The following subsections describe XO events.

Event Types

The following table presents the various types of XO system events.

Note: The XO event type is provided as a parameter to ws_scenario_c::on_new_event().

Code	Description	Name
0	Informational	XO_INFO
1	Error conditions	XO_ERR
2	Action must be taken immediately	XO_ALERT
3	Critical conditions	XO_CRIT
4	Warning conditions	XO_WARNING
5	Normal but significant condition	XO_SIGNF_INFO
6	Debug-level events	XO_DEBUG

Event IDs

The following table presents the various event IDs of the XO system events.

Note: The XO event ID is provided as a parameter to ws_scenario_c::on_new_event().

Code	Description	Name
0	Informational	XO_EVENTID_INFO
1	Error conditions	XO_EVENTID_ERROR
2	Connection lost	XO_EVENTID_LOST_CONNECTION
3	Reestablish connection	XO_EVENTID_REESTABLISH_CONNECTION
4	XOFS queue overflow due to excessive number of changes	XO_EVENTID_XOFS_QUEUE_OVERFLOW
5	XOFS queue normalized	XO_EVENTID_XOFS_QUEUE_OK
6	Unable to apply file change	XO_EVENTID_APPLY_FILE_CHANGE_FAILED
7	Synchronization/verification finished	XO_EVENTID_SYNC_FINISHED
8	All modifications during synchronization period are replicated	XO_EVENTID_SYNC_JOURNALS_FINISH
9	Previous synchronization/verification in progress	XO_EVENTID_SYNC_IN_PROG

enum ws_sync_method_e

Defines the synchronization method:

```
enum ws_sync_method_e
{
        initial_synchronization_t,
        file_synchronization_t,
        block_data_synchronization_t
};
```

Note: The enum type is provided as a parameter to the following methods:
- ws_scenario_c::sync()
- ws_scenario_c::generate_difference_report()

Value	Description
initial_synchronization_t	Initial Synchronization — This type of synchronization transfers all master directories without checking data on replicas.
file_synchronization_t	File Synchronization — This type of synchronization is used to synchronize entire files. This option is recommended when most of the changes are file replacements or creations of new files and not file editing.
block_data_synchronization_t	Block/Data Synchronization (Continual) — This type of synchronization transfers only differences. This method reduces synchronization time in case of a slow network, because it avoids sending redundant information (the same algorithm used in the rsync package).

enum ws_report_type_e

Defines the various types of WANSync system reports.

```
enum ws_report_type_e
{
report_sync_t,          /* Synchronization report  */
report_sync_t,          /* Replication report      */
report_difference_t     /* Difference report       */
report_t                /* Generic report          */
};
```

Note: The enum type is provided as a parameter to ws_scenario_c::on_received_report_file().

Value	Description
report_sync_t	Synchronization report
report_replication_t	Replication report
report_difference_t	Difference report
report_t	Generic report

struct ws_directory_statistics_s

Describes statistical information per root directory.

Note: Use the ws_statistics_c::dir_statistics() method in order to retrieve root directory statistics.

```
struct ws_directory_statistics_s
{
const char* dir_full_path;
/* Online file changes */
__uint64            num_replicated_bytes;
unsigned int        num_updated_files;
unsigned int        num_removed_files;
unsigned int        num_renamed_files;
time_t              last_change;                /* Time in seconds since 1/1/70 */
/* File changes due to Sync */
__uint64            num_sync_bytes;
unsigned int        num_sync_changed_files;
unsigned int        num_sync_removed_files;
/* Others */
unsigned intnum_errors;
};
```

You can collect the following data:

- Online file changes (Kbytes transferred, files changed/removed/renamed, last modification time)
- File changes due to synchronization (Kbytes transferred, files changed/removed)
- Number of possible errors

class ws_statistics_c

Describes statistical information of a scenario. You can collect the following data:

- Start time of the replication process
- State information (spool size, registry changes)
- Statistics per root directory
- Scenario name

Note: Use the ws_scenario_c::get_statistics() method in order to retrieve statistics.

Class Methods for class ws_statistics_c

Class methods for class ws_statistics_c are:

Method	ws_statistics_c::ws_statistics_c()
Description	Use this member function to construct a ws_statistics_c object.
Arguments	

Method	bool ws_statistics_c::is_running() const
Description	Returns the scenario status.
Arguments	

Method	bool ws_statistics_c::is_scenario_master() const
Description	Returns an indicator if the statistics belong to a scenario's master node.
Arguments	

Method	time_t ws_statistics_c::replication_start_time() const
Description	Returns the start time of the replication process. Time is in number of seconds, since January 1, 1970.
Arguments	

Method	const char* ws_statistics_c::master_host_name() const

Description	Returns a pointer to a null terminated string holding the host name of the master host of the scenario, which is the DNS name or the IP address.
Arguments	

Method	unsigned short ws_statistics_c::master_port() const
Description	Returns the incoming port used for TCP/IP communications of the master host of the scenario.
Arguments	

Method	uint64 ws_statistics_c::spool_size() const
Description	Returns the spool size in bytes.
Arguments	

Method	uint64 ws_statistics_c::bytes_received() const
Description	Returns the number of bytes coming in to the specific scenario from its parent, or from the application by which it is managed, if the current host is a master host. It includes all replication and synchronization data.
Arguments	

Method	uint64 ws_statistics_c::bytes_sent() const
Description	Returns the number of bytes sent from the specific scenario to its immediate children. It includes all replication and synchronization data.
Arguments	

Method	unsigned int ws_statistics_c::num_of_online_registry_changes() const
Description	Returns the number of online changes in the registry.
Arguments	

Method	unsigned int num_of_sync_registry_changes() const
Description	Returns the number of changes in the registry due to the synchronization process.

Arguments	

Method	const char* ws_statistics_c::scenario_name() const
Description	Returns the scenario name.
Arguments	

Method	unsigned int ws_statistics_c::num_of_directories() const
Description	Returns the number of root directories. Use this number as a maximum index value passed to ws_statistics_c::dir_statistics().
Arguments	

Method	XOSTATUS ws_statistics_c::dir_statistics(unsigned int dir_index, ws_directory_statistics_s* dir_statistics) const
Description	Fills the ws_directory_statistics_s structure with the statistics of the desired root-directory identified by dir_index.
Arguments	
dir_index [in]	Specifies the index of the root directory. An application can call the ws_statistics_c::num_of_directories() function to discover the range of acceptable root directory indexes.
dir_statistics [out]	Pointer to a ws_directory_statistics_s buffer that the function fills with the requested information.

class ws_scenario_c

Represents a replication scenario.

Note: Do not try to create a scenario object manually. Only use the method: ws_control_c::attach_to_scenarios(host_name,ip)). Also, do not try to clear (delete) a scenario object. Only use the method: ws_control_c::detach_from_scenarios() to clear all allocated memory.

The ws_scenario_c class methods allow you to send the following commands to a scenario:

- Start and stop the replication scenario
- Initiate the synchronization process
- Enable/disable logging, and retrieve log files
- Collect scenario and root directory statistics
- Create and retrieve a difference report
- Collect synchronization/replication report files
- Receive scenario events
- Suspend/Resume replica

Class Methods for class ws_scenario_c

The class methods for class ws_scenario_c are:

Method	virtual void ws_scenario_c::on_new_event(unsigned int event_type, unsigned int event_id, const char* host_name, time_t time, const char *msg);
Description	This virtual function is called asynchronously upon receipt of a WANSync event from the hosts participating in the scenario. The application can use this function to handle the events.
Arguments	
event_type [in]	XO event type (see *Event Types*).
event_id [in]	XO event ID (see *Event IDs*).

host_name [in]	A null-terminated string that contains the DNS-name of the host that generated the event.
time [in]	The time the event occurred.
msg [in]	A null-terminated string that contains a brief explanation of the event.

Method	XOSTATUS ws_scenario_c::is_running(bool *result) const
Description	Checks if scenario is running. Returns the scenario status in the result argument.
Arguments	
result [out]	Boolean value. Specifies the scenario status.

Method	XOSTATUS ws_scenario_c::scenario_name(char* buf, unsigned int buf_len) const
Description	Retrieves the scenario name.
Arguments	
buf [out]	Pointer to a buffer that receives a null-terminated string containing the scenario name.
len [in]	Specifies the maximum size, in characters, of the buffer.

Method	unsigned int ws_scenario_c::scenario_id()
Description	Retrieves the scenario unique ID
Arguments	

Method	XOSTATUS ws_scenario_c::run().
Description	Sends a Run command to the scenario.
Arguments	

Method	XOSTATUS ws_scenario_c::number_of_hosts_in_scenario (unsigned int* num_of_hosts) const
Description	Retrieves the number of hosts in the scenarios. Use this number to decide the length of the buffer passed to ws_scenario_c::generate_difference_report().

Arguments	
num_of_hosts [out]	Pointer to a variable that receives the number of hosts in the scenarios.

Method	XOSTATUS ws_scenario_c::stop().
Description	Sends a Stop command to the scenario, Including HA forward scenario.
Arguments	

Method	XOSTATUS ws_scenario_c::sync(ws_sync_method_e sync_type, bool exclude_same_files = true, unsigned int timeout=0).
Description	Sends a Synchronize command to the scenario. This command can be sent only if the scenario is running.
Arguments	
sync_type [in]	Enumerated value that specifies the synchronize method.
exclude_same_files [in]	Boolean value. If false, checks files with the same size and modification time during synchronization; If true, skips them.
timeout [in]	Specifies the time to wait, in seconds, for the synchronization of all the hosts to finish. If timeout is zero, sync() sends the Synchronize command and returns immediately. If timeout is not zero, sync() waits until the synchronization of all the hosts has finished, or the timeout interval has elapsed. Use XO_INFINITE to specify an infinite timeout.

Method	XOSTATUS ws_scenario_c::generate_difference_report(ws_sync_method_e comparison_type, bool exclude_same_files = true, unsigned int timeout=0, char* reports_names_buf=NULL, unsigned int buf_len=0)
Description	Sends a Difference report command, and will asynchronously receive a report of type report_difference_t by the ws_scenario_c:: on_received_report_file(). WANSync will compare each replica with the master and generate a difference report per replica. Report files are stored in the [library-working-directory]/[scenario-name] directory. No synchronization data is transferred. This command can be sent only if the scenario is running.
Arguments	
comparison_type [in]	Enumerated value that specifies the comparison method.
exclude_same_files [in]	Boolean value. If false, checks files with the same size and modification time during comparison; If true, skips them.
timeout [in]	Specifies the timeout in seconds for receiving the difference report of all hosts. If timeout is zero, generate_difference_report() sends the Difference report command, and returns immediately. If timeout is not zero, generate_difference_report() waits until the difference report of all hosts is received, or the timeout interval has elapsed. Use XO_INFINITE to specify an infinite timeout.

reports_names_buf [out]	Pointer to a buffer that receives a series of null-terminated strings (one for each report's full-path name). This buffer ends with a second null character.
buf_len [in]	Specifies the maximum size, in characters, of the buffer pointed to by reports_names_buf. This size includes the terminating null character. If this parameter is zero, reports_names_buf is not used. This value should be set to at least ((MAX_PATH+1) * [number-of-hosts-in-the-scenario]) + 1 to allow sufficient space for the data and the null terminator. Use ws_scenario_c::number_of_hosts_in_scenario() to determine the number of hosts in the scenario.

Method	Virtual void ws_scenario_c::on_received_report_file(ws_report_type_e type, const char *report_file_name)
Description	This virtual function is called asynchronously on receipt of a WANSync report file from the hosts participating in the scenario. The application can use this function to handle the incoming reports. The application is responsible for deleting the report file. Report files are stored in the [library-working-directory]/[scenario-name] directory. The function receives an enum type that indicates the type of the received report file, and the full path name of the report.
Arguments	
type [in]	Enumerated value that specifies the report type
report_file_name [in]	The full path name of the report.

Method	XOSTATUS ws_scenario_c::get_replication_report (char* filename, unsigned int filename_len, unsigned int timeout).
Description	Sends a Replication Report command, to receive the replication report, without waiting for the scheduled replication report. The application is responsible for deleting this file. Report files are stored in the [library-working-directory]/[scenario-name] directory. This command is sent only if the scenario is running.
Arguments	
filename [out]	Pointer to a buffer that receives a null-terminated string containing the path.
filename_len [in]	Specifies the maximum size of the buffer, in characters. This value should be set to at least MAX_PATH+1 to allow sufficient space for the path and the null terminator.
timeout [in]	Specifies the time-out interval, in seconds, to wait for receiving the replication report file. The function returns if the interval elapses, even if the operation was not finished. If timeout is XO_INFINITE, the function's timeout interval never elapses.

Method	XOSTATUS ws_scenario_c::start_stop_logging(bool start)

Description	Sends a Start/Stop command to start or stop logging WANSync messages. The logging facility is an important part of the internal WANSync debugging mechanism. It allows you to trace the replication process of the scenario. When logging is activated, the XOsoftEngine records all of its activities in detail, every file received, and every replication event that has executed. WANSync message logging may downgrade replication performance. **Note:** This command can be sent only if the scenario is running.
Arguments	
start [in]	Indicates whether to start or stop logging.

Method	XOSTATUS ws_scenario_c::get_log_file(char* filename, unsigned int filename_len, unsigned int timeout)
Description	Once the application has instructed a scenario to start logging using the start_stop_logging(true) , it can start retrieving the log files. The application is responsible for deleting this file. Log files are stored in the [library-working-directory]/[scenario-name] directory. **Note:** This command can be sent only if the scenario is running.
Arguments	
filename [out]	Pointer to a buffer that receives a null-terminated string containing the path.
filename_len [in]	Specifies the maximum size of the buffer, in characters. This value should be set to at least MAX_PATH+1 to allow sufficient space for the path and the null terminator.

timeout [in]	Specifies the time-out interval, in seconds, to wait for receiving the log file. The function returns if the interval elapses, even if the operation was not finished. If timeout is XO_INFINITE, the function's timeout interval never elapses.

Method	XOSTATUS ws_scenario_c::get_statistics (ws_statistics_c *statistics) const
Description	Retrieves scenario statistical information. The following data is available: - Time at which replication has started. - State information and various process statistics: Kbytes transferred, files changed/removed/renamed per root directory.
Arguments	
statistics [out]	Pointer to a ws_statistics_c class buffer that the function fills with the requested information. If a NULL pointer is passed, the function fails.

Method	XOSTATUS ws_scenario_c::suspend_replica(const char* host_name, unsigned int port=DEFAULT_PORT).
Description	Suspend the replica running on: (host_name, port).
Arguments	
host_name [in]	Pointer to a null terminated string holding the host name, which is the DNS name or the IP address.
port [in]	The incoming port used for TCP/IP communications. DEFAULT_PORT specifies the XOsoftEngine default port.

Method	XOSTATUS ws_scenario_c::resume_replica().
Description	Resume the manually suspended replica.
Arguments	
statistics [out]	

class ws_control_c

This class manages master scenarios and/or to collect statistical information from the XOsoft Engine running on a given host.

Class methods for class ws_control_c

Class methods are:

Method	ws_control_c::ws_control_c(const char* working_directory=NULL);
Description	Use this member function to construct a ws_ control_c object.
Arguments	
working_directory [in]	A pointer to a null-terminated string specifies a directory to be used for storing the .xmc scenario files, reports and logs. The default directory is [application-working-directory]/ws_control.
	Note: The library working-directory is set only once by the first created ws_control_c object.

Method	virtual ws_scenario_c* ws_control_c::on_create_new_scenario()
Description	This function returns the newly created ws_scenario_c object. Since ws_control_c creates the required scenario objects internally, an application that implements the derived class for ws_scenario_c, should override this function and return the derived object allocation.
Arguments	

Method	Virtual bool on_authentication(const char* hostname, char* domainname, char* username, char* paswd) **Important!** Currently, this method is not in use.

Description	Every connection between the WANSync Control Library and XOsoftEngine is authorized individually with a read/write privilege. In determining the ws_control_c object's rights to a scenario, the ws_control_c first uses the current process user rights. If the ws_control_c does not have the correct authorization level, then this virtual function will be called asynchronously and will ask for a username and password for the XOsoftEngine on the given hostname. This is applicable according to the appropriate license. Regular Windows domain users and the NTLM security provider are used for authentication and authorization. In Windows XP/2003, WANSync uses the Kerberos Authentication Protocol. This function is called repeatedly, whenever a given host's username and password is needed. The function continues to be called until the connection is authorized, or the on_authentication() function returns a value of false.
Arguments	
hostname [in]	Pointer to a const null-terminated ASCII string specifying the name of the XOsoftEngine server that needs to be authenticated.
domainname [out]	Pointer to a null-terminated ASCII string buffer to be filled with the domain-name of the user. The length cannot exceed MAX_COMPUTERNAME_LENGTH bytes.
username [out]	Pointer to a null-terminated ASCII string buffer to be filled with the user-name. The length cannot exceed UNLEN bytes.
paswd [out]	Pointer to a null-terminated ASCII string buffer to be filled with the password of the user. The length cannot exceed PWLEN bytes.

Method	XOSTATUS ws_control_c::attach_to_scenarios(const char *host_name, unsigned short port=DEFAULT_PORT).
Description	Connects to all existing master scenarios on the XOsoftEngine running on: (host_name, port).

Arguments	
host_name [in]	Pointer to a null terminated string holding the host name, which is the DNS name or the IP address.
port [in]	The incoming port used for TCP/IP communications. DEFAULT_PORT specifies the XOsoftEngine default port.

Method	void ws_control_c::detach_from_scenarios().
Description	Disconnects all scenarios and clears all the memory allocated by calls to attach_to_scenarios() . Since the list of ws_scenario_c objects is shared by all ws_control_c objects, the disposal of the ws_scenario_c objects is done only after calling ws_control_c::detach_from_scenarios() from all created ws_control_c objects.
Arguments	

Method	ws_scenario_c* ws_control_c::first_scenario() const
Description	Returns the first scenario object. The application should first call the attach_to_scenarios() method, in order to create scenarios. Do not try to directly delete this object, use detach_from_scenarios() instead. The list of scenarios is shared by all ws_control_c objects. Therefore the first ws_control_c object that attaches scenarios makes these scenarios available to all other ws_control_c objects.
Arguments	

Method	ws_scenario_c* ws_control_c::next_scenario (const ws_scenario_c* current_scenario) const
Description	Enables retrieving the *next* scenario object in the scenario list. (Use the first_scenario() method in order to retrieve the *first* scenario object in the scenario list.) Do not try to directly delete this object, use detach_from_scenarios() instead.
Arguments	

current_ scenario [in]	Pointer to the current scenario.

Method	XOSTATUS ws_control_c::get_host_statistics(const char* host_name, unsigned short port, ws_statistics_c** statistics_list, unsigned int* num_of_statistics) const
Description	Collect statistical information of all existing scenarios (master and replica) on the XOsoftEngine running on, (host_name, port).
Arguments	
host_name [in]	Pointer to a null terminated string holding the host name, which is the DNS name or the IP address.
port [in]	The incoming port used for TCP/IP communications.
statistics_list [out]	Pointer to a variable that receives a pointer to an array of ws_statistics_c objects. If the function succeeds, you must call the delete function to free the returned buffer.
num_of_ statistics [out]	Pointer to a variable that receives the number of ws_statistics_c objects returned in the statistics_list array.

Method	XOSTATUS ws_control_c::register_license(const char* key_str)
Description	Register a new license key. Note that before registering a new key call detach_from_scenarios(), otherwise operation aborts.
Arguments	
key_str [in]	A pointer to a null-terminated string that specifies a key.

Method	XOSTATUS ws_control_c::set_su_group(const char* group_name)

Description	Set WANSync Super User group. Before setting the WANSync Super User group, call detach_from_scenarios(), otherwise the operation aborts. This is applicable only to the ACL-based license.
Arguments	
group_name [in]	A pointer to a null-terminated string that specifies the Domain\Super User group name.

Method	const char*ws_control_c::get_working_directory() const
Description	Returns the library working directory, which is used for storing the .xmc scenario files, reports and logs. The default directory is [application working-directory]/ws_control. This directory can be set in the ws_control_c constructor.
Arguments	

Index

Symbols

(unsigned int* num_of_hosts) const • 15

A

API • 1

Arguments • 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26

B

Block/Data Synchronization (Continual) • 9

block_data_synchronization_t • 9

bool ws_statistics_c

    is_running() const • 11
    is_scenario_master() const • 11

Boolean value • 15, 16

buffer size
    max • 19

C

change in scenario configuration • 6

char *msg)
    const • 14

class
    ws_control_c • 2, 22
    ws_scenario_c • 2, 14
    ws_statistics_c • 2, 11

Class methods
    ws_scenario_c • 14
    ws_statistics_c • 11

class methods • 5

Class methods for class
    ws_control_c • 22

Classes
    and data structures • 2
    structures • 2

codes
    library • 5

Collect
    replication report files • 14
    root directory statistics • 4, 14
    scenario and root directory statistics • 4, 14
    scenario statistics • 1, 14
    synchronization files • 14
    synchronization/replication report files • 4, 14

collect • 4
    statistical information • 1

Collecting statistical information • 4

const
    char *msg) • 14

const char* ws_control_c

    get_working_directory() const • 26

const char* ws_statistics_c

    master_host_name() const • 11
    scenario_name() const • 13

Contact us • 2

Control Library
    WANSync • 1

Create difference report • 1

Critical conditions • 8

D

data structures • 2

Debug-level events • 8

DEFAULT_PORT • 21, 24

Defines the synchronization method • 2

Description • 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

detaching • 4

Difference report • 10, 17

difference report • 4, 14, 17
    retrieve • 1

directory • 19

disable logging • 1, 4

DomainSuper User group name • 26

E

Enable logging • 1, 4
enum
    ws_report_type_e • 2, 9
    ws_sync_method_e • 2, 8, 9
Enumerated value • 16, 18
error codes
    XO • 5
Error conditions • 8
Event
    IDs • 8
    types • 7
event id (see Event IDs).
    XO • 14
event IDs • 8
events
    XO • 7
expired
    License • 6

F

File changes • 10
File Synchronization • 9
file_synchronization_t • 9

G

Generic report • 10
Getting Started • 1

H

Holds
    root directory statistics • 2
    scenario statistics • 2

I

IDs
    Event • 8
Illegal operation • 6
buf_len • 18
current_scenario • 25
dir_index • 13
event_id • 14
event_type • 14
exclude_same_files • 16
filename_len • 19, 20
group_name • 26
host_name • 15, 21, 24, 25
hostname • 23
key_str • 25
len • 15
msg • 15
port • 21, 24, 25
report_file_name • 18
start • 20
sync_type • 16
time • 15
timeout • 16, 17, 19, 21
type • 18
working_directory • 22
Incorrect library object • 5
index
    root directory • 13
information
    State • 21
    state • 21
Informational • 8
Initial Synchronization • 9
initial_synchronization_t • 9
Initiate the synchronization process • 4
int event_id
    unsigned • 14
int event_type
    unsigned • 14
int num_of_sync_registry_changes() const
    unsigned • 12
Invalid WANSync Super User Group • 7

L

library • 3
    codes • 5
    working-directory • 22
library call • 5
library working directory • 26
License
    expired • 6
    is inappropriate • 5
Log files • 20
logging • 14, 20

M

managing a master scenario • 4
mapping
    No • 7
master
    scenario • 4
    scenarios • 1
master scenario • 2
Master scenarios • 1
max
    buffer size • 19
    size buffer • 15
maximum buffer • 20
Method • 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26
Monitoring a master scenario • 4
monitoring a master scenario • 4

N

No
    mapping • 7
No license • 6
Normal condition • 8

O

object
    ws_control_c • 4
    ws_scenario_c • 4
    ws_statistics_c • 11
objects
    ws_scenario_c • 4
Online file changes • 10
buf • 15
dir_statistics • 13
domainname • 23
filename • 19, 20
num_of_hosts • 16
num_of_statistics • 25
paswd • 23
reports_names_buf • 18
result • 15
statistics • 21
statistics_list • 25
username • 23

P

Pointer current scenario • 25

R

Receive scenario events • 1, 4, 14
Register a new license key • 25
Replica
    resume • 1, 14
    suspend • 1, 14
Replication
    Report • 19
    report • 10
replication • 1, 21
replication process
    Start • 11
replication report files
    collect • 1, 4, 14
replication scenario • 14
    start • 4
Report
    Replication • 19
report
    Replication • 10
Report files • 17, 18, 19
report_difference_t • 10

report_replication_t • 10
report_sync_t • 10
report_t • 10
Resume • 21
Resume replica • 1, 14
retrieve log files • 1, 4
Retrieves the scenario name. • 15
root directories • 13
root directory • 10
    index • 13
root directory statistics • 4
    collect • 14
    Holds • 2
Run • 15

S

sample.cpp • 1
sample.h • 1
scenario
    master • 4
scenario configuration • 6
Scenario name • 11
scenario name • 13
scenario state • 6
scenario statistics
    Collect • 1
    collect • 4, 14
    Holds • 2
scenario status • 11
scenarios
    master • 1
Set WANSync Super User group • 26
size buffer
    max • 15
spool size • 12
Start
    replication scenario • 4
    time
        replication process • 11
start replication scenario • 4
start time • 11
Start/Stop • 20
State information • 11, 21
statistical information • 4
    collect • 1
Statistics per root directory • 11
Stop • 16
stop replication scenario • 4
struct ws_directory_statistics_s • 2, 10
structures
    Classes • 2
Super User Group.
    WANSync • 7
Suspend replica • 1, 14
Suspend the replica • 21
synchronization • 14
synchronization files
    Collect • 14
synchronization process • 1
Synchronization report • 10
synchronization report files
    Collect • 4
    collect • 1
synchronization/verification • 6
Synchronize • 16
Synchronize command • 16

T

TCP/IP communications • 24, 25
The operation completed successfully. • 5
time to wait • 16
time_t time • 14
time_t ws_statistics_c
        replication_start_time() const • 11
timeout • 17
timeout period • 6
traverse master scenario objects • 3
Traversing master scenario list • 3
types
    Event • 7

U

uint64 ws_statistics_c

bytes_received() const • 12
bytes_sent() const • 12
spool_size() const • 12

uncompress file • 5

unsigned
int event_id • 14
int event_type • 14
int num_of_sync_registry_changes() const • 12

unsigned int ws_statistics_c

num_of_directories() const • 13
num_of_online_registry_changes() const • 12

unsigned short ws_statistics_c

master_port() const • 12

User not authorized • 5

V

Virtual bool • 22

Virtual void • 18

virtual void • 14

virtual ws_scenario_c* ws_control_c

on_create_new_scenario() • 22

void ws_control_c

detach_from_scenarios () • 24

W

WANSync
Control Library • 1
Super User Group • 7

WANSync event • 14

WANSync Super User Group • 6, 7

WANSync system reports • 2

Warning conditions • 8

working-directory
library • 22

ws_ control_c object • 22

ws_control.h • 1

ws_control_c • 4, 22, 23

detach_from_scenarios() • 4
ws_control_c(const char* working_directory=NULL) • 22
class • 2, 22
Class methods for class • 22
object • 4

ws_control_c constructor. • 26

ws_directory_statistics_s • 13

ws_report_type_e
enum • 2, 9

ws_scenario_c • 14, 24
class • 2, 14
Class methods • 14
object • 4
objects • 4

ws_scenario_c* ws_control_c

first_scenario() const • 24
next_scenario (const ws_scenario_c* current_scenario) const • 24

ws_statistics_c • 11

ws_statistics_c() • 11
class • 2, 11
Class methods • 11
object • 11

ws_sync_method_e
enum • 2, 8, 9

X

XML configuration file • 6

XO
error codes • 5
event id (see Event IDs). • 14
events • 7

XO event type (see Event types). • 14

XO_ALERT • 8

xo_base.h • 1

XO_CRIT • 8

XO_DEBUG • 8

XO_ERR • 8

XO_ERROR_CONFIGURATION_CHANGED_MANUALLY • 6
XO_ERROR_CONNECTION_CLOSED • 5
XO_ERROR_CONNECTION_FAILED • 5
XO_ERROR_CONNECTION_SEIZED • 5
XO_ERROR_DB_VERIFY_IN_PROGRESS • 7
XO_ERROR_DOWNLOAD_SCENARIO_FAILED • 6
XO_ERROR_DOWNLOAD_SCENARIO_IN_PROGRESS • 6
XO_ERROR_GET_SCENARIO_STATE_FAILED • 6
XO_ERROR_INAPPROPRIATE_LICENSE • 5
XO_ERROR_INAPPROPRIATE_SU_GROUP • 7
XO_ERROR_INSUFFICIENT_BUFFER • 5
XO_ERROR_INVALID_OBJECT • 5
XO_ERROR_INVALID_PARAMETER • 5
XO_ERROR_LICENSE_EXPIRED • 6
XO_ERROR_NO_LICENSE • 6
XO_ERROR_NO_MORE_DATA • 5
XO_ERROR_NOT_AUTHORIZED • 5
XO_ERROR_OFFLINE_PROCESSING_IN_PROGRESS • 7
XO_ERROR_OK • 5
XO_ERROR_OPERATION_ABORTED • 6
XO_ERROR_OPERATION_FAILED • 6
XO_ERROR_READ_CONFIGURATION_FAILED • 6
XO_ERROR_READ_SU_GROUP_FAILED • 7
XO_ERROR_SCENARIO_STOPPED • 6
XO_ERROR_SCHEDULED_DB_VERIFY • 7
XO_ERROR_SCHEDULED_OFFLINE_PROCESSING • 7
XO_ERROR_SCHEDULED_SUSPEND • 7
XO_ERROR_SEND_CMD_FAILED • 6
XO_ERROR_SEND_FILE_FAILED • 6
XO_ERROR_SU_GROUP_NOT_MAPPED • 7
XO_ERROR_SU_GROUP_NOT_SPECIFIED • 6
XO_ERROR_SU_GROUP_VIOLATED • 7
XO_ERROR_SUSPEND_IN_PROGRESS • 7
XO_ERROR_SYNC_OR_DIFF_IN_PROGRESS • 6
XO_ERROR_TIMEOUT • 6
XO_ERROR_UNCOMPRESS_FILE_FAILED • 5
XO_ERROR_VERIFY_CONFIGURATION _FAILED • 6
XO_ERROR_VERSION_MISMATCHING • 5
XO_INFINITE • 16, 17
XO_INFO • 8
XO_SIGNF_INFO • 8
XO_WARNING • 8
XOsoft Engine • 1
XOSTATUS • 13, 19, 23
XOSTATUS ws_control_c
    get_host_statistics(• 25
    register_license(const char* key_str) • 25
    set_su_group(const char* group_name) • 25
XOSTATUS ws_scenario_c
    generate_difference_report(• 17
    get_log_file(char* filename, unsigned int filename_len, unsigned int timeout) • 20
    get_statistics (ws_statistics_c *statistics) const • 21
    number_of_hosts_in_scenario • 15
    resume_replica() • 21
    run(). • 15
    scenario_name(char* buf, unsigned int buf_len) const • 15
    start_stop_logging(bool start) • 19
    stop(). • 16
    suspend_replica(const char* host_name, unsigned int port=DEFAULT_PORT). • 21