CA ARCserve® Replication and High Availability

SOAP API Reference Guide

r16.5

arcserve®

Arcserve Product References

This document references the following Arcserve products:

- Arcserve® Replication
- Arcserve® High Availability (HA)
- Arcserve® Assured Recovery®
- Arcserve® Content Distribution

Contact Arcserve

The Arcserve Support team offers a rich set of resources for resolving your technical issues and provides easy access to important product information.

https://www.arcserve.com/support

With Arcserve Support:

- You can get in direct touch with the same library of information that is shared internally by our Arcserve Support experts. This site provides you with access to our knowledge-base (KB) documents. From here you easily search for and find the product-related KB articles which contain field-tested solutions for many top issues and common problems.
- You can use our Live Chat link to instantly launch a real-time conversation between you and the Arcserve Support team. With Live Chat, you can get immediate answers to your concerns and questions, while still maintaining access to the product.
- You can participate in the Arcserve Global User Community to ask and answer questions, share tips and tricks, discuss best practices and participate in conversations with your peers.
- You can open a support ticket. By opening a support ticket online, you can expect a callback from one of our experts in the product area you are inquiring about.

You can access other helpful resources appropriate for your Arcserve product.

Providing Feedback About Product Documentation

If you have comments or questions about Arcserve product documentation, please contact us.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- This is the first release of this reference guide.

Contents

Chapter 1: Introduction

This section contains the following topics:

Programming with SOAP API - the Flow

To control Arcserve Replication and High Availability using SOAP API, the client should authenticate against RHA CS and then create a session. This operation is accomplished by invoking the create_session API function. After the client creates a session, the client can invoke any API function described in this reference guide. When the task is complete, the client calls the close_session API to invalidate the session and free the resources.

Overview of SOAP API

This guide describes SOAP API exported by Arcserve Replication and High Availability. The APIs let you manage various aspects of replication and high availability. For example, you can control RHA scenarios and can perform various operations such as running or stopping scenarios and perform a scenario failover operation. To use Arcserve Replication and High Availability SOAP API, you can use any programming language that contains integrated web services support, such as Visual Basic or C#.

Note: The sample code provided in this reference guide is written in the C#.

The SOAP API URL used in the sample code is http://127.0.0.1:8088/ws_man/xosoapapi.asmx. Substitute address part of the URL with your RHA CS IP address or host name when calling API.

Conventions Used in this Reference Guide

The Arcserve Replication and High Availability SOAP API Reference Guide uses the argument types described in the following table:

Type	Description
bool	boolean value

Type	Description
out uint	output integer
out ulong	output unsigned long integer
out string	output string
uint	unsigned integer
ulong	unsigned long integer
ushort	unsigned short integer

Chapter 2: SOAP APIs

This section contains the following topics:

Scenario Management APIs

The following sections describe APIs that let you perform basic scenario management tasks.

This section contains the following topics:

Starting Work: create_session()

The create_session API function lets you authenticate against Control Service and returns the session ID. You pass the session ID as an argument when calling any other API function. The open session can be invalidated by using the close_session API (see page 24).

Arguments

The create_session API function includes the arguments described in the following table:

Name	Type	Description
user_name	string	The user name. **Example:** MyDomain\Administrator
password	string	The password for the user name. Example: Arcserve
error_code	out uint	A zero value indicates that API function was executed successfully. A nonzero value indicates failure API.

Return Values

When authentication using create_session is successful, this function returns uint value with the session ID and an error_code argument set to 0. Otherwise, the MAX uint value (0xFFFFFFFF) is returned and the error_code argument contains the detailed error code.

Note: You should specify the user_name argument in the form <DOMAIN_NAME>\<USER_NAME>. For example, test_domain\Administrator.

Examples

Example 1: Creating a web services object.

```
xosoapapi_c get_mng()
    {
       xosoapapi_c mng = new xosoapapi_c();
       return mng;
    }
```

Example 2: Creating a session.

```
public bool create_session_example()
    {
      try
      {
        uint err = 0;
        string user_name = global::api_examples.Properties.Settings.Default.user_name;
        string password = global::api_examples.Properties.Settings.Default.password;
        _session_id = get_mng().create_session(user_name, password, out err);
        if (_session_id == 0xffffffff)
        {
          return false;
        }
        return true;
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Creating Scenarios: create_scenario_ex()

The create_scenario_ex API function lets you create scenarios. After you create scenarios, you can perform operations such as running the scenario, changing the scenario properties, adding or removing hosts, and so on.

Arguments

The create_scenario_ex API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
product	uint	Specify an integer value for the following products: 0 - DR scenario 1 - HA scenario 2 - CD scenario

Name	Type	Description
application	uint	Specify an integer value for the following applications: 0 - File server scenario. 1- Exchange scenario 2 - Oracle scenario 3 - Not used 4 - SQL server scenario 5 - IIS scenario 6 - Control Service scenario 7 - Hyper-v scenario 8 - Sharepoint scenario 9 - vCenter scenario 10 - CRM scenario 11- Full system scenario 12 - Customize scenario
is_ar	bool	Specifies whether the scenario supports assured recovery (AR): ■ true: supports AR ■ false: does not support AR
is_cdp	bool	Always false, not used.
integrate_opt	uint	Specify a real integer value for the following applications: 0 - No integration 1 - Backup 2 - Arcserve D2D 3 - Arcserve Central Applications
group_id	uint	Group ID. When the group_id is 0xFFFFFFFF, the scenario belongs to the default scenario group, which is usually named scenarios.

Name	Type	Description
scenario_data	out string	The data for the scenario. After calling this API, the argument contains the updated scenario data. **Note:** For more information, see Example in this topic.
group_data	out string	The group data.

Return Values

This API returns the scenario ID that was created. The other APIs, such as run and stop require the scenario id. When the scenario id is zero, the API failed. Else, non-zero values indicate that the API completed successfully.

Note: When this API call completes successfully, the skeleton of the scenario is created. All of the properties for the scenario have the default values. Important properties such as the root directory, the host IP, and so on are empty. You call other APIs to fulfill the properties. The subsequent APIs in this reference guide describe how to fulfill the properties.

Example

```
uint product = 0;
        uint app = 0;
        bool is_ass_rec = false;
        bool is_cdp = false;
        uint si_opt = 0;
        uint group_id = 0xFFFFFFFF;
        string scenario_data = "";
        string group_data = "";

        uint  scenario_id =  get_mng().create_scenario_ex(_session_id, product, app, is_ass_rec, is_cdp, si_opt,
group_id, out scenario_data, out group_data);
```

Setting Root Directories: set_root_dir()

The set_root_dir() API function lets you edit the path of the root directory.

Arguments

The set_root_dir API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_id	uint	The ID of the scenario.
host_index	uint	The index of the host that you want to set.
root_dir_index	uint	The index of the root directory, start from the zero, if you have two root directories, the indexes are 0 and 1.
root_dir	string	The folder name. **Note:** This argument is required for only full system scenarios. However, this API function does not currently support setting root directories for full system scenarios.

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Note: The create_scenario_ex (see page 11) API creates only a skeleton of the scenario. You can call this API to set the root directory.

Example

```
uint master_host_index = 1;

get_mng().set_root_dir(_session_id, scenario_id, master_host_index, 0, "E:/test");
```

Adding Root Directories: add_root_dir()

The add_root_dir API function lets you add a new root directory for the scenario.

Arguments

The add_root_dir API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	he session ID that was returned by calling the create_session (see page 9) function API.
scenario_data	ref string	The data for the scenario. After calling this API, the argument contains the updated scenario data.
root_directories	string	The folder name.
new_root_dir_index	out uint	The index of the newly created root directory.

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Example: Adding a root directory

```
String scenario_data = get_mng().get_scenario_data(_session_id, scenario_id);

//add root directory
get_mng().add_root_dir(_session_id, ref scenario_data, "c:/test", out new_root_dir_id);
```

Adding Replicas: add_replica()

The add_replica API function lets you add a new replica host for the scenario. If you want to replicate the master data to more than one replica host, you can call this API to add some replica hosts.

Arguments

The add_replica API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_data	ref string	The data for the scenario. After calling this API, the argument contains the updated scenario data.
host_index	uint	The parent host index; the master host index is always 1; the first replica usually is 2.
new_replica_index	out uint	The index of the newly added replica host

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Example: Adding a replica

```
//add one replica under the master host
get_mng().add_replica(_session_id, ref scenario_data, master_host_index, out new_replica_index);
```

Adding Multiple Replicas: add_replica_ex()

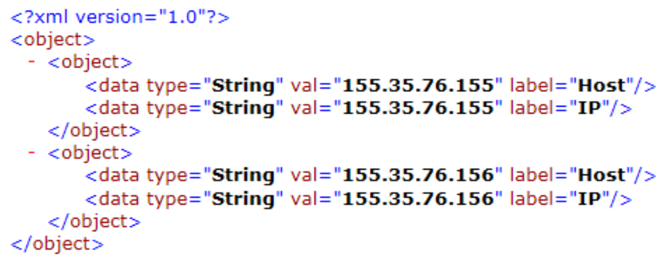
The add_replica_ex API function lets you add one or more new replica hosts for the scenario at one time.

Arguments

The add_replica_ex API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_data	ref string	The data for the scenario. After calling this API, the argument contains the updated scenario data.
host_index	uint	The parent host index; the master host index is always 1; the first replica usually is 2.
host_list	ref string	The host list provides information about hosts in an xml format. The xml contains information about the hosts that you want to add to the scenario. **Note:** See the screen following this table.

For example, the following xml code contains information about two hosts.



```
<?xml version="1.0"?>
<object>
  - <object>
        <data type="String" val="155.35.76.155" label="Host"/>
        <data type="String" val="155.35.76.155" label="IP"/>
    </object>
  - <object>
        <data type="String" val="155.35.76.156" label="Host"/>
        <data type="String" val="155.35.76.156" label="IP"/>
    </object>
</object>
```

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Example: Add multiple replicas

```
string host_list = "<?xml version=\"1.0\"?><object> <object><data label=\"Host\" val=\"155.35.76.155\" type=\"String\" /><data label=\"IP\" val=\"155.35.76.155\" type=\"String\" /> </object><object><data label=\"Host\" val=\"155.35.76.156\" type=\"String\" /><data label=\"IP\" val=\"155.35.76.156\" type=\"String\" /> </object></object>";


//add two replicas under the master host
get_mng().add_replica_ex(_session_id, ref scenario_data, master_host_index, ref host_list);
```

Setting Scenario Properties: set_scenario_data()

The set_scenario_data API function lets you edit the scenario property. Most of the properties of the scenario can be updated by this API.

Arguments

The set_scenairo_data API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_id	uint	The ID of the scenario.
property_name	string	The property name of the host. Each property has a unique name.
property_value	string	The value of the property.

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Example: Set scenario properties

```
//set scenario data properties
get_mng().set_scenario_data(_session_id, scenario_id, "Scenario.ReplicateCompressAttr", "True");
```

The following table lists common scenario property names.

Property Name	Description
Scenario.ScenarioName	The scenario name
Scenario.BuildShares	Synchronize the windows share
Scenario.SyncADS	Replicate NTFS ADS

Setting Host Properties: set_host_data()

The set_host_data API function lets you edit the host property. Most of the properties of the host can be updated by this API.

Arguments

The set_host_data API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
host_index	uint	The index of the host.
scenario_id	uint	The ID of the scenario.
property_name	string	The property name of the host. Each property has a unique name.
property_value	string	The value of the property.

Return Values

This function returns a value of true when the API call completes successfully. Otherwise, this function returns a value of false.

Example: Set host properties

```
//set the host name and IP address for the host
get_mng().set_host_data(_session_id, scenario_id, master_host_index,
"Scenario.ReplicationTree.ReplNode.CommonHostProps.Host", "master");

get_mng().set_host_data(_session_id, scenario_id, master_host_index,
"Scenario.ReplicationTree.ReplNode.CommonHostProps.IP", "155.35.78.187");
```

The following table lists common master host property names.

Property Name	Description
Scenario.ReplicationTree.ReplNode.CommonHostProps.Host	Master host name
Scenario.ReplicationTree.ReplNode.CommonHostProps.IP	Master host IP
Scenario.ReplicationTree.ReplNode.CommonHostProps.Data_IP	Master replication IP address
Scenario.ReplicationTree.ReplNode.CommonHostProps.Port	Master host connection port number
Scenario.ReplicationTree.ReplNode.CommonHostProps.SyncScriptBefore	Run script before synchronization
Scenario.ReplicationTree.ReplNode.CommonHostProps.SyncScriptBefore.Path	The script path
ReplicationTree.ReplNode.CommonHostProps.SyncScriptBefore.Args	The script arguments

Starting Scenarios: run()

The run API function lets you run scenarios. As a best practice, call the add_credentials_ex (see page 26) API to authenticate the Master host before you call the run API.

Arguments

The run API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_id	uint	The scenario ID to be started.

Name	Type	Description
sync_method	uint	The synchronization method, which can be one of the following options: **0** - File synchronization **1** - Block synchronization **2** - Volume synchronization (for only FullSystem scenario.)
ignore_same_files	bool	Ignore the same size/time files.
arc_upt	bool	This parameter applies to only Arcserve Backup integrated scenarios. When the scenario is not a Arcserve Backup integrated parameter, you should set this parameter to false. When the scenario is integrated with Arcserve Backup, call the add_bab_credentials (see page 26) API before you call the run API.
verification_and_run	uint	Reserved. Must always be defined as 1.
message	out string	Contains a detailed description of the error when this API fails.

Return Values

This function returns a value of true when the scenario starts successfully. Otherwise, this function returns a value of false and provides a detailed description of the error in the message argument.

Example

```
public bool run_example()
    {
        try
        {
            uint scenario_id = _scenario_id;
            //0:File Sync;1:Block Sync;2:Volume Sync
            uint sync_method = 0;
            bool ignore_same_files = true;
            bool arc_integrated = false;
            string message = "";
            return get_mng().run(_session_id, scenario_id, sync_method, ignore_same_files, arc_integrated, 1, out
message);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Stopping Scenarios: stop()

The stop API function lets you stop running scenarios.

Arguments

The stop API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID to be stopped.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously. When you set this argument to true, the API function will not return a value until the scenario is stopped. Otherwise, the function return a value immediately.
why_not_reason	out string	Contains a detailed description of the error when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message returned in the why_not_reason argument to determine why the API failed.

Example

```
public bool stop_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string why_not_reason = "";
        bool execute_sync = true;
        return get_mng().stop(_session_id, scenario_id, execute_sync, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Pausing Data Replication: suspend_replication()

The suspend_replication API function lets you suspend replication operations for a host.

Arguments

The suspend_replication API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID to be suspended.
replica_index	uint	The replica host index in a scenario. Typically, the value of the replica index is 2.

Name	Type	Description
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously. When you set this argument to true, the API function will not return a value until the operation completes. Otherwise, the function return a value immediately.
message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool suspend_replication_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string message = "";
        bool execute_sync = true;
        uint replica_index = 2;
        return get_mng().suspend_replication(_session_id, scenario_id, replica_index, execute_sync, out
message);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Finishing Work: close_session()

The close_session API function lets you log out of the Control Service. After you log in to the Control Service, you call the close_session argument to log out of the Control Service.

Arguments

The close_session API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the session was closed. When the return value is false, the session was not closed.

Example

```
public bool close_session_example()
    {
      try
      {
        string why_not_reason = "";
        return get_mng().close_session(_session_id, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Managing Credentials: add_credentials_ex()

The add_credentials_ex API function lets you add credentials for hosts.

Arguments

The add_credentials_ex API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) function API.
scenario_id	uint	The scenario ID to which you want to add credentials.
host_name	string	The host IP address. **Example:** 155.35.76.44.
port	uint	The engine port number. Typically, the port number is 25000.
user_name	string	The user name. **Example:** Administrator
password	string	The password for the user name. **Example:** Arcserve
domain_name	string	The domain name. **Example:** arcserve.com

Return Values

The return type is ulong. A return value of zero indicates that API completed successfully. When the return value does not equal zero, the API failed.

Example

```
public bool add_credentials_ex_example()
    {
       try
       {
          uint scenario_id = _scenario_id;
          //Add credential for Master
          string host_ip = "155.35.66.138";
          uint port = 25000;
          string user_name = "administrator";
          string password = "caworld";
          string domain_name = "155.35.66.138";
          ulong res = get_mng().add_credentials_ex(_session_id, scenario_id, host_ip, port, user_name, 
password, domain_name);
          //Add credential for Replica
          host_ip = "155.35.66.142";
          domain_name = "155.35.66.142";
          res = get_mng().add_credentials_ex(_session_id, scenario_id, host_ip, port, user_name, password, 
domain_name);
          return (res == 0);
       }
       catch (Exception ex)
       {
          System.Windows.Forms.MessageBox.Show(ex.Message);
       }
       return false;
    }
```

Integrating with Arcserve Backup: add_bab_credentials()

The add_bab_credentials API function lets you add credentials so that you can access Arcserve Backup.

Arguments

The add_bab_credentials API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID to which you want to add Arcserve credentails.

Name	Type	Description
username	string	The user name. **Example:** Administrator
password	string	The password for the user name. **Example:** Arcserve
async_id	out ulong	When this API function is called asynchronously, the async_id argument will contain a nonzero value, which allows the function to wait for the operation to complete complete and retrieve the result.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully.

Example

```
public bool add_bab_credentials_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string username = "admin";
        string password = "caworld";
        ulong async_id = 0;
        bool res = get_mng().add_bab_credentials(_session_id, scenario_id,username,password,out async_id);
        return res;
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Updating Scenario Properties: update_scenario()

The update_scenario API function lets you update scenario data at run time.

Arguments

The update_scenario API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session API.
scenario_id	uint	The scenario ID that you want to update.
scenario_data_str	ref string	The data about the scenario. When the operation completes successfully, the new scenario data is provided.
why_not_reason	ref string	Contains the reason for failure when this API fails.

Return Values

The return values provide the data for the scenario in xml format.

Example

```
public bool update_scenario_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string why_not_reason = "";
        string scenario_data_str = get_mng().get_scenario_data(scenario_id);
        //do some changes for  the scenario.
        return get_mng().update_scenario(_session_id, scenario_id, ref scenario_data_str, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Removing Scenarios: remove_scenario()

The remove_scenario API function lets you remove scenarios from the Control Service.

Arguments

The remove_scenario API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID that will run.
arc_upd	bool	This parameter applies to only Arcserve Backup integrated scenarios.
why_not-reason	out string	Contains the reason for failure when this API fails.

Return Value

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool remove_scenario_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        bool arc_integrated = false;
        string why_not_reason = "";
        return get_mng().remove_scenario(_session_id, scenario_id, arc_integrated, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Importing Scenarios: import_scenario()

The import_scenario API function lets you import scenarios to the Control Service.

Arguments

The import_scenario API function includes the arguments described in the following table:

Type	Name	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
group_id	uint	The scenario group ID that you want to import. If you are not sure of the group ID, set this to 0xFFFFFFFF.
scenario_id	out uint	Retrieves the scenario id when the API completes successfully.
scenario_data	string	The data about the scenario. Typically, you retrieve the scenario data string from a scenario file. The data is in an XML format.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return values provide the data for the scenario in xml format.

Example

```
public bool import_scenario_example()
    {
       try
       {
          uint scenario_id = 0;
          string why_not_reason = "";
          string scenario_data = "load the data from a scenario file.";
          uint group_id = 0xFFFFFFFF;
          return get_mng().import_scenario(_session_id, group_id,scenario_data,out scenario_id,out
why_not_reason);
       }
       catch (Exception ex)
       {
          System.Windows.Forms.MessageBox.Show(ex.Message);
       }
       return false;
    }
```

Synchronizing Running Scenarios: synchronize()

The synchronize API function lets you synchronize the data for scenarios.

Arguments

The synchronize API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID that you want to synchronize.
sync_method	uint	The synchronization method, which can be one of the following options: **0** - File synchronization **1** - Block synchronization **2** - Volume synchronization (for only FullSystem scenario.)
ignore_same_files	bool	Ignore the same size/time files.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.

Name	Type	Description
message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool synchronize_example()
    {
        try
        {
            uint scenario_id = _scenario_id;
            string message = "";
            bool execute_sync = true;
            uint sync_method = 1;
            bool ignore_same_files = false;
            return get_mng().synchronize(_session_id, scenario_id, sync_method, ignore_same_files,
execute_sync, out message);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Resuming Replication: resume_replication()

The resume_replication API function lets you resume replication operations for a host.

Arguments

The suspend_replication API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session API.

Name	Type	Description
scenario_id	uint	The scenario ID for which you want to resume replication.
replica_index	uint	The replica host index in a scenario. Typically, the value of the replica index is 2. This is the host for which replication will be suspended. For example, the data changes are aggregated in a spool without copying the data to a disk until the replication operation resumes.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.
message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool resume_replication_example()
    {
        try
        {
            uint scenario_id = _scenario_id;
            string message = "";
            bool execute_sync = true;
            uint replica_index = 2;
            return get_mng().resume_replication(_session_id, scenario_id, replica_index, execute_sync, out
message);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Adding Rewind Bookmarks: set_rewind_bookmark()

The set_rewind_bookmark API function lets you set bookmarks for scenarios.

Arguments

The set_rewind_bookmark API function includes the arguments described in the following table:

Name	Type	Description
scenario_id	string	The scenario ID for which you want to set bookmarks.
host_index	uint	Always = 1.
bookmark_msg	string	The bookmark name.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool set_rewind_bookmark_example()
    {
      try
      {
        string scenario_id = _scenario_id.ToString();
        uint host_index = 1;
        string why_not_reason = "";
        string bookmark_msg = "test bookmark";
        return get_mng().set_rewind_bookmark(scenario_id, host_index, bookmark_msg, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

High Availability Scenario Management APIs

The following sections describe APIs that let you manage high availability scenarios.

This section contains the following topics:

Triggering Assured Recovery: start_ar()

The start_ar API function lets you perform an assured recovery operation (AR) for a scenario. When you perform an automatic AR, you do not need to call other APIs to stop the AR. The AR will stop after the AR operation completes. When you perform a manual AR, call the API resume_application (see page 33) to stop the AR operation.

Arguments

The start_ar API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID for which you want to perform assured recovery.
replica_index	uint	The replica host index in a scenario. Typically, the value of the replica index is 2. This is the host for which replication will be suspended.
auto_ar	bool	Run AR automatically or manually. ■ True- automatically ■ False-manually
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.
message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool start_ar_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string message = "";
        bool execute_sync = true;
        uint replica_index = 2;
        bool auto_ar = true;
        return get_mng().start_ar(_session_id, scenario_id, replica_index, auto_ar, execute_sync, out message);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Disabling Heartbeat in High Availability Scenarios: stop_is_alive()

Is-alive is an electronic signal that replica servers send to master servers to identify the status of the node. While high availability scenarios run, the replica server periodically sends an electronic signal (ping) to the master server. By default, the frequency of the ping is 30 seconds. You can trigger a switchover event when the replica server cannot ping the master after a predetermined period of time elapses (the default is 300 seconds).

The stop_is_alive API function lets you suspend the is-alive check.

Arguments

The stop_is_alive API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.

Name	Type	Description
scenario_id	uint	The scenario ID for which you want to suspend the is-alive check.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.
err_message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool stop_is_alive_example()
    {
      try
      {
        uint scenario_id = _ha_scenario_id;
        string err_messages = "";
        bool execute_sync = true;
        return get_mng().stop_is_alive(session_id, scenario_id, execute_sync, out err_messages);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Switching Over for High Availability Scenarios: switchover()

The switchover API function lets you perform switchover operations.

With full system, high availability scenarios, you can perform switchover operations to any replica server. When you want to switch over to non-failover replica servers, you call the execute_action API before you call the switchover API.

Note: The execute_action API is described in the examples.

Arguments

The switchover API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID for which you want to perform switchover operations.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.
run_reverse_scenario	bool	Run or do not run the backward scenario after the switchover operation occurs.
err_message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Examples

Example 1

```
public bool switchover_example()
    {
        try
        {
            uint scenario_id = _ha_scenario_id;
            string err_messages = "";
            bool execute_sync = true;
            bool run_reverse_scenario = false;
            return get_mng().switchover(_session_id, scenario_id, execute_sync, run_reverse_scenario, out
err_messages);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Example 2

```
public bool switchover_2nd_example()
    {
        try
        {
set_xcmd_data("switchover", "switchover_index","3" );
            uint scenario_id = _ha_scenario_id;
            string err_messages = "";
            bool execute_sync = true;
            bool run_reverse_scenario = false;
            return get_mng().switchover(_session_id, scenario_id, execute_sync, run_reverse_scenario, out
err_messages);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Example 3

By default, Replication and High Availability performs switchover operations to the predefined failover, replica servers. With full system, high availability scenarios, you can switch over to non-failover replica servers. However, when you want to switch over to non-failover servers using the switchover API, you call the execute_action API before you call the switchover API as illustrated by the following example:

```
set_xcmd_data("switchover", "switchover_index","3" );

public bool set_xcmd_data(string cmd_name_str,string cmd_data_str,string cmd_value_str)
    {
      try
      {
        string result_data = "";
        string action_data;

        XmlDocument doc = new XmlDocument();
        XmlNode actions = doc.CreateNode(XmlNodeType.Element, xomngapi.WANSync_c.xo_actions, "");

        XmlNode commonNode = doc.CreateNode(XmlNodeType.Element,
xomngapi.WANSync_c.action_common_lab, "");
        XmlAttribute attrSession = doc.CreateAttribute(xomngapi.WANSync_c.action_com_session_id);
        XmlAttribute attrScenario = doc.CreateAttribute(xomngapi.WANSync_c.action_com_scenario_id);
        XmlAttribute attrHostindex = doc.CreateAttribute(xomngapi.WANSync_c.action_com_host_index);
        XmlAttribute attrUsedfor = doc.CreateAttribute(xomngapi.WANSync_c.action_used_for);

        attrSession.Value = xomngapi.WANSync_c.WANSync.session_id.ToString();
        attrScenario.Value = this.id.ToString();
        attrUsedfor.Value = xomngapi.WANSync_c.action_x_command_data;

        commonNode.Attributes.Append(attrSession);
        commonNode.Attributes.Append(attrScenario);
        commonNode.Attributes.Append(attrHostindex);
        commonNode.Attributes.Append(attrUsedfor);

        XmlNode xo_cmd = doc.CreateNode(XmlNodeType.Element, xomngapi.WANSync_c.xo_cmd, "");
        XmlAttribute cmd_name = doc.CreateAttribute(xomngapi.WANSync_c.action_cmd_name);
        XmlAttribute cmd_data = doc.CreateAttribute(xomngapi.WANSync_c.action_cmd_data);
        XmlAttribute cmd_value = doc.CreateAttribute(xomngapi.WANSync_c.action_cmd_value);

        cmd_name.Value = cmd_name_str;
        cmd_data.Value = cmd_data_str;
        cmd_value.Value = cmd_value_str;

        xo_cmd.Attributes.Append(cmd_name);
        xo_cmd.Attributes.Append(cmd_data);
        xo_cmd.Attributes.Append(cmd_value);

        actions.AppendChild(commonNode);
        commonNode.AppendChild(xo_cmd);
        doc.AppendChild(actions);
        action_data = doc.OuterXml;
        string error;
        return get_mng().execute_action(action_data, true, out result_data, out error);
      }
      catch (System.Exception)
      {
        return false;
      }
```

Enabling Heartbeats in High Availability Scenarios: start_is_alive()

Is-alive is an electronic signal that replica servers send to master servers to identify the status of the node. While high availability scenarios run, the replica server periodically sends an electronic signal (ping) to the master server. By default, the frequency of the ping is 30 seconds. You can trigger a switchover event when the replica server cannot ping the master after a predetermined period of time elapses (the default is 300 seconds).

The start_is_alive API function lets you resume the is-alive check.

Arguments

The start_is_alive API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID for which you want to start the is-alive check.
execute_sync	bool	Regulates whether to call this API synchronously or asynchronously.
err_message	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool start_is_alive_example()
    {
      try
      {
        uint scenario_id = _ha_scenario_id;
        string err_messages = "";
        bool execute_sync = true;
        return get_mng().start_is_alive(session_id, scenario_id, execute_sync, out err_messages);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

VSS Snapshot Management APIs

The following sections describe APIs that let you manage VSS snapshots.

This section contains the following topics:

Mounting VSS Snapshot on Replica Servers: mount_snapshot()

The mount_snapshot API function lets you mount VSS snapshots to a specific folder on a replica server.

Arguments

The mount_snapshot API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
host_name	string	The engine host name

Name	Type	Description
ip_string	string	The IP address of the host_name.
host_port	ushort	The engine port number. Typically, the port number is 25000.
mount_path	string	The folder where you want to mount the snapshot.
snapshot_id	string	The VSS snapshot ID.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool mount_snapshot_example()
    {
      try
      {
        string host_name = "155.35.66.142";
        string ip_string = "155.35.66.142";
        ushort host_port = 25000;
        string mount_path = "c:/mount";
        string snapshot_id = "{9CFDE664-62D5-4fd8-A304-2B664900B98F}";
        string why_not_reason = "";
        return get_mng().mount_snapshot(session_id, host_name, ip_string, host_port, snapshot_id,
mount_path, out why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Unmounting VSS Snapshots from Replica Servers: unmount_snapshot()

The unmount_snapshot API function lets you unmount VSS snapshots from a folder.

Arguments

The unmount_snapshot API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
host_name	string	The engine host name
ip_string	string	The IP address of the host_name.
host_port	ushort	The engine port number. Typically, the port number is 25000.
snapshot_id	string	The VSS snapshot ID.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool unmount_snapshot_example()
    {
      try
      {
        string host_name = "155.35.66.142";
        string ip_string = "155.35.66.142";
        ushort host_port = 25000;
        string snapshot_id = "{9CFDE664-62D5-4fd8-A304-2B664900B98F}";
        string why_not_reason = "";
        return get_mng().unmount_snapshot(session_id, host_name, ip_string, host_port, snapshot_id, out
why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Removing VSS Snapshots from Replica Servers: delete_snapshot()

The delete_snapshot API function lets you delete VSS snapshots from replica servers.

Arguments

The delete_snapshot API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
host_name	string	The engine host name
ip_string	string	The IP address of the host_name.
host_port	ushort	The engine port number. Typically, the port number is 25000.
snapshot_id	string	The VSS snapshot ID.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool delete_snapshot_example()
    {
      try
      {
        string host_name = "155.35.66.142";
        string ip_string = "155.35.66.142";
        ushort host_port = 25000;
        string snapshot_id = "{9CFDE664-62D5-4fd8-A304-2B664900B98F}";
        string why_not_reason = "";
        return get_mng().delete_snapshot(session_id, host_name, ip_string, host_port, snapshot_id, out
why_not_reason);
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Getting Lists of VSS Snapshots from Replica Servers: get_snapshot_list()

The get_snapshot_list API function lets you get a list of the VSS snapshots from a host.

Arguments

The get_snapshot_list API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
host_name	string	The engine host name
ip_string	string	The IP address of the host_name.
host_port	ushort	The engine port number. Typically, the port number is 25000.

Name	Type	Description
snapshot_list	out string	The volume snapshot list.
why_not_reason	out string	Contains the reason for failure when this API fails.

Return Values

The return type is Boolean. When the return value is true, the API completed successfully. When the return value is false, the API did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Example

```
public bool get_snapshot_list_example()
    {
        try
        {
            string host_name = "155.35.66.142";
            string ip_string = "155.35.66.142";
            ushort host_port = 25000;
            string snapshot_list = "";
            string why_not_reason = "";
            return get_mng().get_snapshot_list(session_id, host_name, ip_string, host_port, out snapshot_list, out
why_not_reason);
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
        return false;
    }
```

Gathering Statistics APIs

The following sections describe APIs that let you manage the process of gathering statistics about scenarios.

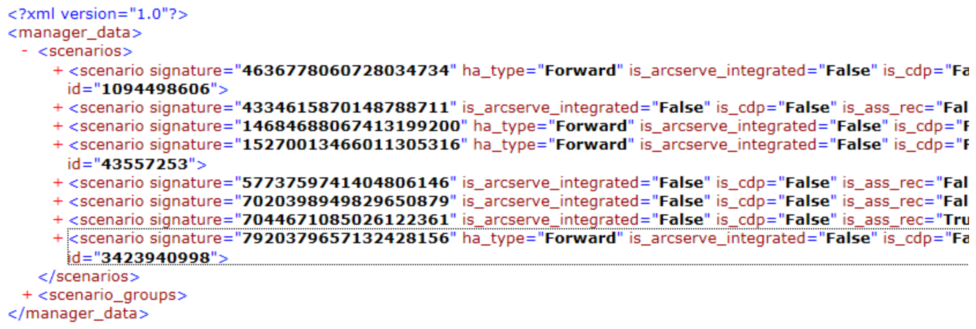
This section contains the following topics:

Getting Extended Scenario Statistics: get_data_ex()

The get_data_ex API function lets you get all of the scenario information, such as the following:

- Scenario events
- Scenario states
- Synchronization and replication statistics

The parameter manager_data is an xml format string. The data contains all of the scenario information. For example, the scenario status (running, stopped, and so on), events, scenario statistics, and so on. The xml format string resembles the following:



```
<?xml version="1.0"?>
<manager_data>
 - <scenarios>
    + <scenario signature="4636778060728034734" ha_type="Forward" is_arcserve_integrated="False" is_cdp="Fa
      id="1094498606">
    + <scenario signature="4334615870148788711" is_arcserve_integrated="False" is_cdp="False" is_ass_rec="Fal
    + <scenario signature="14684688067413199200" ha_type="Forward" is_arcserve_integrated="False" is_cdp="F
    + <scenario signature="15270013466011305316" ha_type="Forward" is_arcserve_integrated="False" is_cdp="F
      id="43557253">
    + <scenario signature="5773759741404806146" is_arcserve_integrated="False" is_cdp="False" is_ass_rec="Fal
    + <scenario signature="7020398949829650879" is_arcserve_integrated="False" is_cdp="False" is_ass_rec="Fal
    + <scenario signature="7044671085026122361" is_arcserve_integrated="False" is_cdp="False" is_ass_rec="Tru
    + <scenario signature="7920379657132428156" ha_type="Forward" is_arcserve_integrated="False" is_cdp="Fa
      id="3423940998">
   </scenarios>
 + <scenario_groups>
</manager_data>
```

The following sections demonstrate how to use the xml string.

Arguments

The get_data_ex API includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenarios_with_statistics	uint	The array of scenario IDs. Retreives the statistics for the scenarios.
last_update_time	ulong	Last updated timestamp.
request_flag	uint	Requests the data type. The values can be as follows: **1** - scenario data **2** - cdp data [not used] **4** - host management data **8** - snapshot data **15** - all above data

Name	Type	Description
manager_data	out string	Return the data for the scenario in xml format.

Return Values

The return type is Boolean. When the return value is true, the command completed successfully. When the return value is false, the command did not complete successfully. If the return value is false, review the message to determine the reason the API failed.

Examples

Example 1:

```
public bool get_data_ex_example()

    {

        try

        {

            uint[] scenarios_with_statistics = new uint[] { _scenario_id };

            uint request_flag = 1;

            ulong last_update_time = 0;

            string manager_data = "";

            bool res = get_mng().get_data_ex(_session_id, scenarios_with_statistics, request_flag, ref
last_update_time, out manager_data);

            return res;

        }

        catch (Exception ex)

        {

            System.Windows.Forms.MessageBox.Show(ex.Message);

        }

        return false;

    }
```

Example 2:

This function parses the XML buffer data (manager_data) that was returned by the get_data_ex() function. The following example describes how to get the scenario state when it is running or stopped:

```
string get_scenario_state (string manager_data, string scenario_name_or_id)
    {
      string scenario_state = "unknown";
      XmlDocument manager_data_doc = new XmlDocument();
      manager_data_doc.LoadXml(manager_data);
      XmlNode root_node = manager_data_doc.SelectSingleNode(".//manager_data");
      if (root_node == null)
      {
        return scenario_state;
      }
      //get all the scenario data information
      XmlNode scenario_nodes = root_node.SelectSingleNode(".//scenarios");
      if (scenario_nodes == null)
      {
        return scenario_state;
      }
      foreach (XmlNode scenario_node in scenario_nodes.ChildNodes)
      {
        uint scenario_id = 0;
        string scenario_name = "";
        XmlAttribute id_attr = scenario_node.Attributes["id"];
        if (id_attr != null)
          scenario_id = Convert.ToUInt32(id_attr.Value);
        XmlAttribute name_attr = scenario_node.Attributes["name"];
        if (name_attr != null)
          scenario_name = name_attr.Value;

        if (scenario_name_or_id != scenario_id.ToString() && scenario_name_or_id.ToLower() !=
scenario_name.ToLower())
          continue;

        //get the scenario status, running or stopped
        foreach (XmlNode node in scenario_node.ChildNodes)
        {
          //get the scenario state, running or stop
          if (0 == string.Compare("state", node.Name, true))
          {
            if (node.Attributes["val"] != null)
            {
              scenario_state = node.Attributes["val"].Value.ToLower();
            }
          }
        }
      }
    return scenario_state;
}
```

Example 3:

This function parses the XML buffer data (manager_data) that was returned by the get_data_ex() function. The following example describes how to get all of the events for the scenario:

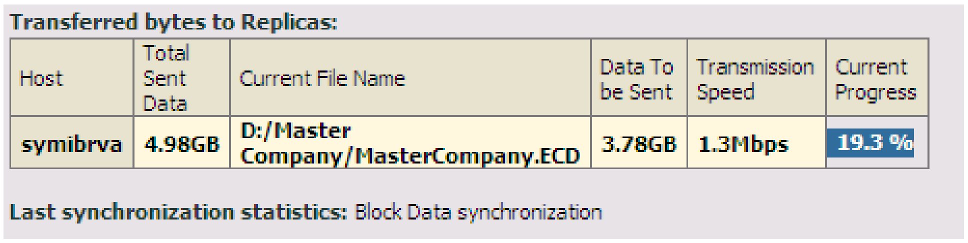
```
void get_scenario_events(string manager_data, string scenario_name_or_id, ref ArrayList events)
    {
        XmlDocument manager_data_doc = new XmlDocument();
        manager_data_doc.LoadXml(manager_data);
        XmlNode root_node = manager_data_doc.SelectSingleNode(".//manager_data");
        if (root_node == null)
        {
            return ;
        }
        //get all the scenario data information
        XmlNode scenario_nodes = root_node.SelectSingleNode(".//scenarios");
        if (scenario_nodes == null)
        {
            return ;
        }
        foreach (XmlNode scenario_node in scenario_nodes.ChildNodes)
        {
            uint scenario_id = 0;
            string scenario_name = "";
            XmlAttribute id_attr = scenario_node.Attributes["id"];
            if (id_attr != null)
                scenario_id = Convert.ToUInt32(id_attr.Value);
            XmlAttribute name_attr = scenario_node.Attributes["name"];
            if (name_attr != null)
                scenario_name = name_attr.Value;

            if (scenario_name_or_id != scenario_id.ToString() && scenario_name_or_id.ToLower() !=
scenario_name.ToLower())
                continue;

            //get the scenario status, running or stopped
            foreach (XmlNode node in scenario_node.ChildNodes)
            {
                //get the scenario state, running or stop
                if (0 == string.Compare("gen", node.Name, true))
                {
                    events.Add(new event_data_c(node));
                }
            }
        }
    }
```

Example 4:

You define the scenarios_with_statistics parameter to get the synchronization and replication statistics for a scenario. The parameter is an array. To get the statistics for more than one scenario, you add the IDs of the scenarios to the array.

The get_data_ex gets the following statistics:

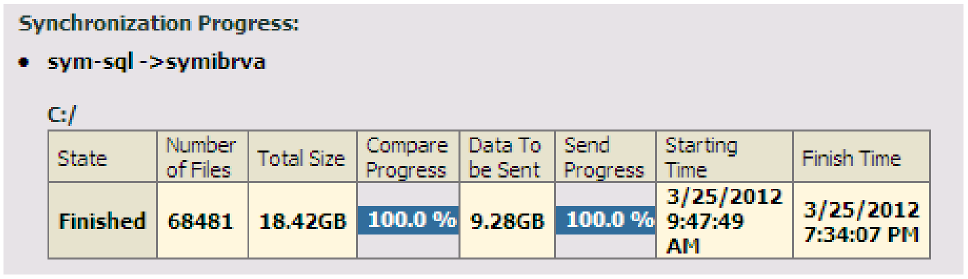
Note: The following screens illustrate the process of transferring and synchronization statistics information.



Transferred bytes to Replicas:

Host	Total Sent Data	Current File Name	Data To be Sent	Transmission Speed	Current Progress
symibrva	4.98GB	D:/Master Company/MasterCompany.ECD	3.78GB	1.3Mbps	19.3 %

Last synchronization statistics: Block Data synchronization



Synchronization Progress:

- sym-sql ->symibrva

C:/

State	Number of Files	Total Size	Compare Progress	Data To be Sent	Send Progress	Starting Time	Finish Time
Finished	68481	18.42GB	100.0 %	9.28GB	100.0 %	3/25/2012 9:47:49 AM	3/25/2012 7:34:07 PM

Code

This function parses the XML buffer data (manager_data) that was returned by the get_data_ex() function. The following code demonstrates how to get the statistics of transferring and synchronization. (See the previous screens.)

```
void get_scenario_sync_statistics(string manager_data, string scenario_name_or_id, ref ArrayList sync_statistics)

    {
        XmlDocument manager_data_doc = new XmlDocument();
        manager_data_doc.LoadXml(manager_data);
        XmlNode root_node = manager_data_doc.SelectSingleNode(".//manager_data");
        if (root_node == null)
        {
            return;
        }
        //get all the scenario data information
        XmlNode scenario_nodes = root_node.SelectSingleNode(".//scenarios");
        if (scenario_nodes == null)
        {
            return;
        }
        foreach (XmlNode scenario_node in scenario_nodes.ChildNodes)
        {
            uint scenario_id = 0;
            string scenario_name = "";
            XmlAttribute id_attr = scenario_node.Attributes["id"];
            if (id_attr != null)
                scenario_id = Convert.ToUInt32(id_attr.Value);
            XmlAttribute name_attr = scenario_node.Attributes["name"];
            if (name_attr != null)
                scenario_name = name_attr.Value;

            if (scenario_name_or_id != scenario_id.ToString() && scenario_name_or_id.ToLower() !=
scenario_name.ToLower())
                continue;

            //get the scenario status, running or stopped
            foreach (XmlNode node in scenario_node.ChildNodes)
            {
                //get the scenario state, running or stop
                if (0 == string.Compare("statistics", node.Name, true))
                {
                    sync_statistics.Add(new host_statistics_c(node));
                }
            }
        }
    }
```

Example 5:

The following code demonstrates how to retrieve sync statistics for scenarios.

```
ArrayList sync_statistics = new ArrayList();
        get_scenario_sync_statistics(manager_data, "FileServer 1", ref sync_statistics);
        //show the statistics
        foreach (host_statistics_c stat in sync_statistics)
        {
          //host name
          string host_name = stat.host_name;
          //transmission statistics
          foreach (transfer_to_replica_c trans in stat.trans_to_reps)
          {
            //handle the transfer data such as speed.
            ulong speed = trans.transmission_speed;
          }
          //sync statistics
          foreach (sync_statistics_host_c sync_host in stat.children_hosts)
          {
            //root directory
            foreach(sync_statistics_root_dir_c root_dir in sync_host.sync_root_dirs)
            {
              //root_dir.total_size
            }
          }
        }
```

Getting Scenario Statistics: get_scenario_data()

The get_scenario_data API function lets you get the scenario IDs.

Arguments

The get_scenario_data API function includes the arguments described in the following table:

Name	Type	Description
session_id	uint	The session ID that was returned by calling the create_session (see page 9) API.
scenario_id	uint	The scenario ID.

Return Values

The return values provide the data for the scenario.

Example

```
public bool get_scenario_data_example()
    {
      try
      {
        uint scenario_id = _scenario_id;
        string scenario_data_str = get_mng().get_scenario_data(session_id.scenario_id);
        return true;
      }
      catch (Exception ex)
      {
        System.Windows.Forms.MessageBox.Show(ex.Message);
      }
      return false;
    }
```

Index