Blocking API

Published: May 27, 2013

Introduction

This chapter describes the features and operations of the Blocking API and provides code examples. It also introduces reply timeout feature that is unique to the Blocking API.

Note If you only need to develop an *automatic integration*, skip this chapter and go directly to Chapter 4, "Nonblocking API."

Multithreading Support

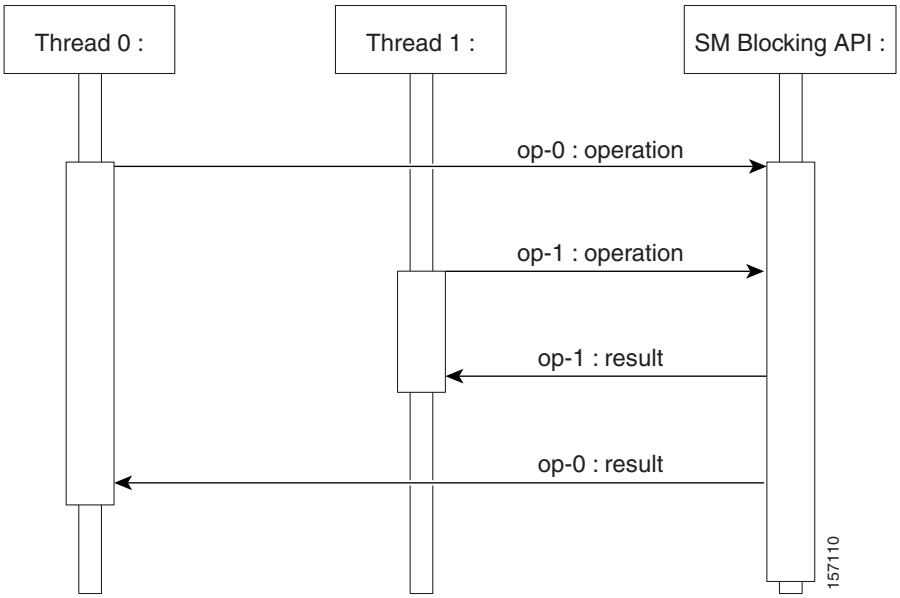
The Blocking API supports a configurable number of threads calling its methods simultaneously. For more information about configuring the number of threads, see the "C++ init Method" section on page 3-36.

Note In a multithreaded scenario for the Blocking API, the order of invocation is *not* guaranteed.

Multithreading Support Example

Thread 0 calls operation 0 at time 0, and thread 1 calls operation 1 at time 1, where time 1 is later than time 0. In this example, it is possible that operation 1 may be performed *before* operation 0, as shown in Figure 3-1 (the vertical scale is time).

Figure 3-1 Multithreading Support



The Cisco Service Control Subscriber Manager allocates five threads to process each API instance. You should develop a multithreaded application that uses the API with a number of threads (in the order of the five threads). Implementing with more threads might result in longer delays for calling the threads.

Operation Timeout Error Code

A Blocking operation returns only when the operation result has been retrieved from the Cisco Service Control Subscriber Manager. If a networking malfunction or other error prevents the operation result from being retrieved, the caller waits indefinitely. The Cisco Service Control Subscriber Manager API provides ways to work around this situation.

The setReplyTimeout method of the reply timeout feature enables the caller to set a timeout. It returns a ReturnCode with the ERROR_CODE_CLIENT_OPERATION_TIMEOUT error when a reply does not return within the timeout period.

Calling the setReplyTimeout function with an int value sets a reply timeout. The reply timeout is interpreted in milliseconds. A zero value indicates that the operation should wait (freeze or stop responding) until a result arrives—or indefinitely, if no result arrives.

Blocking API Methods

This section lists the methods of the Blocking API.

The Blocking API is a superset of the Nonblocking API. Except for differences in return values and result handling, identical operations in both APIs have the same functions and syntax structure.

The C API and C++ API share the same function signature as the first parameter in all functions, except for SMB_ prefix, for all function names for the Blocking C APIs and for the API handle of type SMB_HANDLE. The function description defines any other differences between the APIs.

The Blocking API subscriber management methods can be classified into the following categories:

- **Dynamic IP and property allocation**—For using the Cisco Service Control Subscriber Manager API for integration with an AAA system, the following methods are relevant:
  - login, page 3-4
  - logoutByName, page 3-8
  - logoutByNameFromDomain, page 3-10
  - logoutByMapping, page 3-12
  - loginCable, page 3-13
  - logoutCable, page 3-15

Note These methods are not designed to add or remove subscribers from the database, but to modify dynamic parameters (such as IP addresses) of existing subscribers.

- **Static/manual subscriber configuration**—For GUI usage, for example, the following methods are relevant:
  - addSubscriber, page 3-16
  - removeSubscriber, page 3-18
  - removeAllSubscribers, page 3-19
  - setPropertiesToDefault, page 3-33
  - removeCustomProperties, page 3-34
- **Simple read-only operations performed independently in subscriber awareness mode**—The following methods are relevant:
  - getNumberOfSubscribers, page 3-20
  - getNumberOfSubscribersInDomain, page 3-21
  - getSubscriber, page 3-22
  - subscriberExists, page 3-23
  - subscriberLoggedIn, page 3-24
  - getSubscriberNameByMapping, page 3-25
  - getSubscriberNames (all), page 3-26
  - getSubscriberNames (filter by property), page 3-28
  - getSubscriberNamesInDomain, page 3-29
  - getSubscriberNamesWithPrefix, page 3-31
  - getSubscriberNamesWithSuffix, page 3-32
  - getDomains, page 3-33

It is possible to combine methods from different categories in a single application. The classification is presented only for clarification purposes.

- **API maintenance—Initialization, connection, disconnection**—The following methods are relevant:
  - C++ setLogger Method, page 3-35
  - C++ init Method, page 3-36
  - C SMB_init Function, page 3-37

Note The examples that appear at the end of the described methods are in C++. Every example described at the end of the methods should be preceded by the following sample code:

```
SmApiBlocking bapi;
// Init with default parameters
bapi.init();
// Connect to the SM
bapi.connect((char*)"1.1.1.1");
```

login

The following sections describe login operation information:

Syntax

The login syntax is as follows:

```
ReturnCode* login(char* argName,
                  char** argMappings,
                  MappingType* argMappingTypes,
                  int argMappingsSize,
                  char** argPropertyKeys,
                  char** argPropertyValues,
                  int argPropertySize,
                  char* argDomain,
                  bool argIsAdditive,
                  int argAutoLogoutTime)
```

Description

The login method adds or modifies a domain, mappings, and properties of a subscriber that already exists in the Cisco Service Control Subscriber Manager database. A login can be called with partial data, for example, with only mappings or only properties provided and NULL put in the unchanged fields.

If another subscriber with the same (or colliding) mappings already exists in the same domain, the colliding mappings are removed from the other subscriber and assigned to the new subscriber.

If the subscriber does not exist in the Cisco Service Control Subscriber Manager database, it is created with the data provided.

Parameters

Login parameters are as follows:

- argName—For a description, see the "Subscriber Name Format" section on page 2-7.
- argMappings—For a description, see the "Network ID Mappings" section on page 2-7. If no mappings are specified, and the argIsAdditive parameter is TRUE, the previous mappings are retained. If no such mappings exist, the operation fails.
- argMappingTypes—For a description, see the "Network ID Mappings" section on page 2-7.
- argMappingsSize—The size of the argMappings and argMappingTypes arrays.
- argPropertyKeys—For a description, see the "Subscriber Properties" section on page 2-10.
- argPropertyValues—For description, see the "Subscriber Properties" section on page 2-10.
- argPropertySize—The size of the argPropertyKeys and argPropertyValues arrays.
- argDomain—For description, see the "Subscriber Domains" section on page 2-9.

  If domain is NULL, but the subscriber already has a domain, the existing domain is retained.

  If the domain differs from the domain that was previously assigned to the subscriber, the subscriber is removed automatically from the SCEs of the previous domain and moved to the SCEs of the new domain.
- argIsAdditive—Mapping parameters.
  - TRUE—Adds the mappings provided by this call to the subscriber record.
  - FALSE—Overrides the mappings that already exist in the subscriber record with the mappings provided by this call.
- argAutoLogoutTime—Applies only to the mappings provided as arguments to the login method.
  - Positive value (*N*)—Automatically logs out the mappings (similar to a logout method being called) after *N* seconds.
  - 0 value—Maintains the current expiration time for the given mappings.
  - Negative value—Disables any expiration time that might have been set for the mappings given.

Return Value

The return value is a pointer to a ReturnCode structure with a void type, unless an error has occurred.

Error Codes

The following list presents the error codes that the login method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_DATABASE_EXCEPTION

- ERROR_CODE_UNKNOWN

  Instances of the following value specifications can cause this error:

  - NULL value for the domain parameter for the subscriber that does not exist or does not have a domain
  - Invalid values for the propertyValues parameter

For a description of error codes, see Appendix A, “List of Error Codes”.

Example

To add the IP address 192.168.12.5 to an existing subscriber named alpha without affecting existing mappings:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

bapi.login(
   "alpha",                          // subscriber name
   &ip_address,
   &map_type,
   1,                               // one mapping
   NULL, NULL, 0,                   // no properties
   "subscribers",                   // domain
   true,                            // isMappingAdditive is true
   -1);                             // autoLogoutTime set to infinite
```

To add the IP address 192.168.12.5 overriding previous mappings:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

bapi.login(
   "alpha",                          // subscriber name
   &ip_address,
   &map_type,
   1,
   NULL, NULL,0,
   "subscribers",                   // domain
   false,                           // isMappingAdditive is false
   -1);                             // autoLogoutTime set to infinite
```

To extend the auto logout time of 192.168.12.5 that was previously assigned to alpha:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

bapi.login(
   "alpha",                          // subscriber name
   &ip_address,
   &map_type,   1,
   NULL, NULL, 0,
   "subscribers",                   // domain
   false,                           // isMappingAdditive is false
   300);                            // autoLogoutTime set to 300 seconds
```

To modify a dynamic property of alpha,for example, package ID:

```
char* prop_name = "packageID";
char* prop_value = "10";

bapi.login(
```

```
   "alpha",
   NULL, NULL, 0,
   &prop_name,                    // property key
   &prop_value,                   // property value
   1,                             // one property
   "subscribers",                 // domain
   false,
   -1);
```

To add the IP address 192.168.12.5 to an existing subscriber named alpha without affecting existing mappings and to modify a dynamic property of alpha,for example, package ID:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

char* prop_name = "packageID";
char* prop_value = "10";

bapi.login(
   "alpha",
   &ip_address,
   &map_type,
   1,
   &prop_name,                    // property key
   &prop_value,                   // property value
   1,
   "subscribers",                 // domain
   true,                          // isMappingAdditive is set to true
   -1);
```

To add the IPv6 address abcc:abcd:bcde:fce1:10::/64 to an existing subscriber named alpha without affecting the existing mappings:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcc:abcd:bcde:fce1:10::/64";
bapi.login(
         "alpha", // subscriber name
         &ip_address,
         &map_type,
         1, // one mapping
         NULL, NULL, 0, // no properties
         "subscribers", // domain
         true, // isMappingAdditive is true
         -1); // autoLogoutTime set to infinite
```

To add the IPv6 address abcd:bcde:fce1:10::/64 overriding the existing mappings:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcd:bcde:fce1:10::/64";
bapi.login(
         "alpha", // subscriber name
         &ip_address,
         &map_type,
         1,
         NULL, NULL,0,
         "subscribers", // domain
         false, // isMappingAdditive is false
         -1); // autoLogoutTime set to infinite
```

To extend the auto logout time of the IPv6 address abcd:bcde:fce1:10::/64 that was previously assigned to alpha:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcd:bcde:fce1:10::/64";
```

```
bapi.login(
         "alpha", // subscriber name
         &ip_address,
         &map_type, 1,
         NULL, NULL, 0,
         "subscribers", // domain
         false, // isMappingAdditive is false
         300); // autoLogoutTime set to 300 seconds
```

To modify a dynamic property of alpha,for example, package ID:

```
char* prop_name = "packageID";
char* prop_value = "10";
bapi.login(
         "alpha",
         NULL, NULL, 0,
         &prop_name, // property key
         &prop_value, // property value
         1, // one property
         "subscribers", // domain
         false,
         -1);
```

To add the IPv6 address abcd:bcde:fce1:10::/64 to an existing subscriber named alpha without affecting existing mappings and to modify a dynamic property of alpha, for example, package ID:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcd:bcde:fce1:10::/64";
char* prop_name = "packageID";
char* prop_value = "10";
bapi.login(
         "alpha",
         &ip_address,
         &map_type,
         1,
         &prop_name, // property key
         &prop_value, // property value
         1,
         "subscribers", // domain
         true, // isMappingAdditive is set to true
         -1);
```

logoutByName

The following sections provide information about the logoutByName operation:

Syntax

The logoutByName syntax is as follows:

```
ReturnCode* logoutByName(char* argName,
```

```
char** argMappings,
MappingType* argMappingTypes,
int argMappingsSize)
```

Description

The logoutByName method locates the subscriber in the database and removes mappings from the subscriber.

Parameters

The logoutByName parameters are as follows:

argName—For description, see the "Subscriber Name Format" section on page 2-7.

argMappings—For description, see the "Network ID Mappings" section on page 2-7. If no mappings are specified, all the subscriber mappings are removed.

argMappingTypes—For description, see the "Network ID Mappings" section on page 2-7.

argMappingsSize—The size of the argMappings and argMappingTypes arrays.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber was found and the subscriber mappings were removed from the subscriber database.
- FALSE—If the subscriber was not found in the subscriber database.

Error Codes

The following list provides the error codes that the logoutByName method might return:

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

Example

To remove IP address 192.168.12.5 from subscriber alpha:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

bapi.logoutByName(
   "alpha",
   &ip_address,
   &map_type,
   1);
```

To remove the IPv6 address abcd:bcde:fce1:10::/64 from the subscriber alpha:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcd:bcde:fce1:10::/64";
```

```
bapi.logoutByName(
        "alpha",
        &ip_address,
        &map_type,
        1);
```

To remove all the IPv6 addresses from the subscriber alpha:

```
bapi.logoutByName("alpha", NULL, NULL,0);
```

To remove all the IP addresses from the subscriber alpha:

```
bapi.logoutByName("alpha", NULL, NULL,0);
```

logoutByNameFromDomain

The following sections provide information about the logoutByNameFromDomain operation:

Syntax

The logoutByNameFromDomain syntax is as follows:

```
ReturnCode* logoutByNameFromDomain (char* argName,
                                    char** argMappings,
                                    MappingType* argMappingTypes,
                                    int argMappingsSize,
                                    char* argDomain)
```

Description

The logoutByNameFromDomain method locates the subscriber in the database according to the specified domain and removes mappings from the subscriber.

Parameters

argName—For description, see the "Subscriber Name Format" section on page 2-7.

argMappings—For description, see the "Network ID Mappings" section on page 2-7. If no mappings are specified, all the subscriber mappings are removed.

argMappingTypes—For description, see the "Network ID Mappings" section on page 2-7.

argMappingsSize—The size of the argMappings and argMappingTypes arrays.

argDomain—See the description of subscriber domains in the "Subscriber Domains" section on page 2-9.

The operation fails if either of the following conditions exists:

- The domain is null, but the subscriber exists in the database and belongs to a domain.
- The domain specified is incorrect.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber was found and the subscriber mappings were removed from the subscriber database.
- FALSE—If the subscriber was not found in the subscriber database.

Error Codes

The following list provides error codes that logoutByNameFromDomain method might return:

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, “List of Error Codes”.

Example

To remove IP address 192.168.12.5 of subscriber alpha from domain subscribers:

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";

bapi.logoutByNameFromDomain(
   "alpha",
   &ip_address,
   &map_type,
    1,
   "subscribers");
```

To remove all IP addresses of subscriber alpha from domain subscribers:

```
bapi.logoutByNameFromDomain(
   "alpha",
   NULL,
   NULL,
   0,
   "subscribers");
```

To remove the IPv6 address abcd:bcde:fce1:10::/64 of the subscriber alpha from the domain subscribers:

```
MappingType map_type = TYPE_IPV6;
char* ip_address = "abcd:bcde:fce1:10::/64";
bapi.logoutByNameFromDomain(
            "alpha",
            &ip_address,
            &map_type,
```

```
            1,
            "subscribers");
```

To remove all the IPv6 addresses of the subscriber alpha from domain subscribers:

```
bapi.logoutByNameFromDomain(
            "alpha",
            NULL,
            NULL,
            0,
            "subscribers");
```

logoutByMapping

The following sections provide information about the logoutByMapping operation:

Syntax

The logoutByMapping syntax is as follows:

```
ReturnCode* logoutByMapping(char* argMapping,
                            MappingType argMappingType,
                            char* argDomain)
```

Description

The logoutByMapping method locates a subscriber based on domain and mapping, and removes the subscriber mappings. The subscriber remains in the database.

Parameters

argMapping—For description, see the "Network ID Mappings" section on page 2-7.

argMappingType—For description, see the "Network ID Mappings" section on page 2-7.

argDomain—For description, see the "Parameters" section on page 3-10 of the logoutByNameFromDomain method.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber was found and removed from the subscriber database.
- FALSE—If the subscriber was not found in the subscriber database.

Error Codes

The following list provides error codes that logoutByMapping method might return:

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, “List of Error Codes”.

Example

To remove the IP address 192.168.12.5 from domain subscribers:

```
bapi.logoutByMapping(
   "192.168.12.5",
   IP_RANGE,
   "subscribers");
```

To remove the IPv6 address abcd:bcde:fce1:10::/64 from the domain subscribers:

```
bapi.logoutByMapping(
      "abcd:bcde:fce1:10::/64",
      TYPE_IPV6,
      "subscribers");
```

loginCable

The following sections provide information about the loginCable operation:

Syntax

The loginCable syntax is as follows:

```
ReturnCode* loginCable(char* argCpe,
                       char* argCm,
                       char* argIp,
                       int argLease,
                       char* argDomain,
                       char** argPropertyKeys,
                       char** argPropertyValues,
                       int argPropertySize)
```

Description

The loginCable is a login method adapted for the cable environment, which calls the cable support module in the Cisco Service Control Subscriber Manager. This method logs in customer premise equipments (CPE) to the Cisco Service Control Subscriber Manager. To log in a cable modem, specify the cable modem MAC address in both CPE and cable modem arguments. For additional information, see the "CPE as Subscriber in Cable Environment" appendix of *Cisco Service Control Management Suite Subscriber Manager User Guide*.

Note The name of the CPE in the Cisco Service Control Subscriber Manager database is the concatenation of the CPE and cable modem values with two underscore ("_") characters between them. The caller must ensure that the lengths of CPE and cable modem add up to no more than 38 characters.

Parameters

argCpe—Unique identifier of the CPE (usually a MAC address).

argCm—Unique identifier of the cable modem (usually a MAC address).

argIp—CPE IP address.

argLease—CPE lease time.

argDomain—For description, see the "Subscriber Domains" section on page 2-9.

The domain is usually CMTS IP.

Note Domain aliases must be set on the Cisco Service Control Subscriber Manager to correctly interpret the CMTS IP as a domain name. For information about alias configuration, see the "Default Domains Configuration" section of the *Cisco SCMS Subscriber Manager User Guide*.

argPropertyKeys—For description, see the "Subscriber Properties" section on page 2-10. If the CPE is provided with partial or no application properties, the values for the missing application properties are copied from the application properties of the cable modem to which this CPE belongs. Each cable modem application property thus serves as a default for the CPE under it.

argPropertyValues—For description, see the "Subscriber Properties" section on page 2-10.

argPropertySize—Size of the argPropertyKeys and argPropertyValues arrays.

Return Value

The return value is a pointer to a ReturnCode structure with a void type.

Error Codes

None.

Example

To add the IP address 192.168.12.5 to a cable modem called CM1 with two hours lease time:

```
bapi.loginCable(
```

```
   "CM1",
   "CM1",
   "192.168.12.5",
   7200,       // lease time in seconds
   "subscribers",
    NULL, NULL, 0);           // no properties
```

To add the IP address 192.168.12.50 to a CPE called CPE1 behind CM1 with a lease time of one hour:

```
bapi.loginCable(
   "CPE1",
   "CM1",
   "192.168.12.50",
   3600,   // lease time in seconds
   "subscribers",
   NULL, NULL, 0);
```

logoutCable

The following sections provide information about the logoutCable operation:

Syntax

The logoutCable syntax is as follows:

```
ReturnCode* logoutCable(char* argCpe,
                        char* argCm,
                        char* argIp,
                        char* argDomain)
```

Description

The logoutCable method indicates a logout (CPE becoming offline) event to the Cisco Service Control Subscriber Manager cable support module.

Parameters

argCpe—See the description in the "Parameters" section on page 3-14 of the loginCable method.

argCm—See the description in the "Parameters" section on page 3-14 of the loginCable method.

argIp—See the description in the "Parameters" section on page 3-14 of the loginCable method.

argDomain—See the description in the "Parameters" section on page 3-14 of the loginCable method.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the CPE was found and removed from the subscriber database.
- FALSE—If the CPE was not found in the subscriber database.

Error Codes

None.

Example

To remove the IP address 192.168.12.5 from CPE1 that is behind CM1:

```
bool isExist = bapi.logoutCable(
   "CPE1",
   "CM1",
   "192.168.12.5",
   "subscribers");
```

addSubscriber

The following sections provide information about the addSubscriber operation:

Syntax

The addSubscriber syntax is as follows:

```
ReturnCode* addSubscriber(char* argName,
                          char** argMappings,
                          MappingType* argMappingTypes,
                          int argMappingsSize,
                          char** argPropertyKeys,
                          char** argPropertyValues,
                          int argPropertySize,
                          char** argCustomPropertyKeys,
                          char** argCustomPropertyValues,
                          int argCustomPropertySize,
                          char* argDomain)
```

Description

The addSubscriber creates a new subscriber record according to the given data and adds it to the Cisco Service Control Subscriber Manager database. If a subscriber by this name already exists, it is removed before the new one is added. In contrast to the login method**,** which modifies fields passed to it and leaves unspecified fields unchanged, addSubscriber sets the subscriber exactly as specified by the parameters passed to it.

Note Use the login method for logging in existing subscribers, instead of using the addSubscriber method. Use login to set dynamic mappings and properties. Use addSubscriber method to set the static mappings and properties the first time the subscriber is created.

Note With addSubscriber method**,** the auto-logout feature is always disabled. To enable auto-logout, use the login method.

Example:

Subscriber AB*,* already set up in the subscriber database, has a single IP mapping of IP1.

If an addSubscriber operation for AB is called with no mappings specified (NULL in both the mappings and mappingTypes fields), AB is left with no mappings.

However, calling a login operation with these NULL-value parameters does not change mappings of AB; AB still has its previous IP mapping of IP1.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

argMappings—See the description in the "Network ID Mappings" section on page 2-7.

argMappingTypes—See the description in the "Network ID Mappings" section on page 2-7.

argMappingsSize—Size of the argMappings and argMappingTypes arrays.

argPropertyKeys—See the description in the "Subscriber Properties" section on page 2-10.

argPropertyValues—See the description in the "Subscriber Properties" section on page 2-10.

argPropertySize—Size of the argPropertyKeys and argPropertyValues arrays.

argCustomPropertyKeys—See the description in the "Custom Properties" section on page 2-10.

argCustomPropertyValues—See the description in the "Custom Properties" section on page 2-10.

argPropertySize—Size of the argCustomPropertyKeys and argCustomPropertyValues arrays.

argDomain—See the description in the "Subscriber Domains" section on page 2-9.

A NULL value indicates that the subscriber does not have a domain.

Return Value

The return value is a pointer to a ReturnCode structure with a void type.

Error Codes

The following list provides the error codes that the addSubscriber method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_DATABASE_EXCEPTION
- ERROR_CODE_UNKNOWN

  This error code indicates that invalid values were supplied for the propertyValues parameter.

For a description of error codes, see Appendix A, "List of Error Codes".

Example

To add a new subscriber, alpha, with custom properties:

```
char* propKeys[] = { "work_phone", "home_phone" };
char *propValues[] = { "123456", "898765" };

bapi.addSubscriber(
   "alpha",
   NULL, NULL, 0,// dynamic mappings will be set by login
   NULL, NULL, 0,// dynamic properties will be set by login
   propKeys, propValues, 2,// 2 custom properties
   "subscribers");        // default domain
```

To add a new IPv6 subscriber, alpha, with custom properties:

```
char* propKeys[] = { "work_phone", "home_phone" };
char *propValues[] = { "123456", "898765" };

bapi.addSubscriber(
    "alpha",
    NULL, NULL, 0,// dynamic mappings will be set by login
    NULL, NULL, 0,// dynamic properties will be set by login
    propKeys, propValues, 2,// 2 custom properties
    "subscribers"); // default domain
```

removeSubscriber

The following sections provide information about the removeSubscriber operation:

Syntax

The removeSubscriber syntax is as follows:

```
ReturnCode* removeSubscriber(char* argName)
```

Description

The removeSubscriber method removes a subscriber completely from the Cisco Service Control Subscriber Manager database.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber was found in the database and successfully removed.
- FALSE—If the conditions for TRUE were not met; that is, the subscriber was not found in the database, or the subscriber was found but was not successfully removed.

Error Codes

The following list provides error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

Example

To remove subscriber alpha entirely from the database:

```
bapi.removeSubscriber("alpha");
```

removeAllSubscribers

The following sections provide information about the removeAllSubscribers operation:

Syntax

The removeAllSubscribers syntax is as follows:

```
ReturnCode* removeAllSubscribers()
```

Description

The removeAllSubscribers method removes all subscribers from the Cisco Service Control Subscriber Manager, leaving the database with no subscribers.

Note This method might take time to execute. To avoid operation timeout exceptions, set a high operation timeout (up to five minutes) before calling this method.

Return Value

The return value is a pointer to a ReturnCode structure with a void type.

Error Codes

None.

getNumberOfSubscribers

The following sections provide information about the getNumberOfSubscribders operation:

Syntax

The getNumberOfSubscribders syntax is as follows:

```
ReturnCode* getNumberOfSubscribers()
```

Description

The getNumberOfSubscribders method retrieves the total number of subscribers in the Subscriber Manager database.

Return Value

The return value is a pointer to a ReturnCode structure holding an integer that indicates the number of subscribers in the Cisco Service Control Subscriber Manager.

Error Codes

None.

getNumberOfSubscribersInDomain

The following sections provide information about the getNumberOfSubscribersInDomain operation:

Syntax

The getNumberOfSubscribersInDomain syntax is as follows:

```
ReturnCode* getNumberOfSubscribersInDomain(char* argDomain)
```

Description

The getNumberOfSubscribersInDomain method retrieves the number of subscribers in a subscriber domain.

Parameters

argDomain—Name of a subscriber domain that exists in the domain repository of the Subscriber Manager.

Return Value

The return value is a pointer to a ReturnCode structure holding an integer that indicates the number of subscribers in the domain provided.

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE _DOMAIN_NOT_FOUND

For a description of error codes, see Appendix A, “List of Error Codes”.

getSubscriber

The following sections provide information about the getSubscriber operation:

Syntax

The getSubscriber syntax is as follows:

```
ReturnCode* getSubscriber(char* argName)
```

Description

The getSubscriber method retrieves a subscriber record. Each field is formatted as an integer, string, or string array, as described in the Return Value section. If the subscriber does not exist in the Subscriber Manager database, an exception is thrown.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

Return Value

The return value is a pointer to a ReturnCode structure holding an array of ReturnCode structures with nine elements. There is no array value that is NULL.

Table 3-1 lists the element values and their meanings.

Table 3-1 Element Values and Meanings

Element Value	Meaning
Index 0	Subscriber name (char*)
Index 1	Array of mappings (char**)
Index 2	Array of mapping types (short*)
Index 3	Domain name (char*)
Index 4	Array of property names (char**)
Index 5	Array of property values (char**)
Index 6	Array of custom property names (char**)

Table 3-1 Element Values and Meanings (continued)

Element Value	Meaning
Index 7	Array of custom property values (char**)
Index 8	Array of auto-logout time, as seconds from now, or value of -1 if not set (long 1*) one per mapping (index1)

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, “List of Error Codes”.

Example

To retrieve the subscriber record of alpha:

```
ReturnCode* sub = bapi.getSubscriber("alpha");
// sub name
char* name = sub->u.objectArray[0]->u.stringVal;

// sub mapping
char** mappings = sub->u.objectArray[1]->u.stringArrayVal;

// mappings types
short* types = sub->u.objectArray[2]->u.shortArrayVal;

char* domainName = (char*)sub->u.objectArray[3]->u.stringVal;

char** propertyNames = (char**)sub->u.objectArray[4]->u.stringArrayVal;

char** propertyValues = (char**)sub->u.objectArray[5]->u.stringArrayVal;

char** customPropertyName = (char**)sub->u.objectArray[6]->u.stringArrayVal;

char** customPropertyValues = (char**)sub->u.objectArray[7]->u.stringArrayVal;

long* autoLogoutTime = sub->u.objectArray[8]->u.longArrayVal;
```

subscriberExists

The following sections provide information about the subscriberExists operation:

Syntax

The subscriberExists syntax is as follows:

```
ReturnCode* subscriberExists(char* argName)
```

Description

The subscriberExists method verifies that a subscriber exists in the Cisco Service Control Subscriber Manager database.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber was found in the Cisco Service Control Subscriber Manager database.
- FALSE—If the subscriber could not be found.

Error Codes

None.

subscriberLoggedIn

The following sections provide information about the subscriberLoggedIn operation:

Syntax

The subscriberLoggedIn syntax is as follows:

```
ReturnCode* subscriberLoggedIn(char* argName)
```

Description

The subscriberLoggedIn method checks whether a subscriber that already exists in the Cisco Service Control Subscriber Manager database is logged in; that is, if the subscriber also exists in an SCE database.

When the Cisco Service Control Subscriber Manager is configured to work in Pull mode, a TRUE value returned by this method does not guarantee that the subscriber actually exists in an SCE database, but rather that the subscriber is available to be pulled by an SCE if needed.

If the subscriber does not exist in the Cisco Service Control Subscriber Manager database, an exception is thrown.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

Return Value

The return value is a pointer to a ReturnCode structure with a Boolean type:

- TRUE—If the subscriber is logged in.
- FALSE—If the subscriber is not logged in.

Error Codes

The following list provides error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

getSubscriberNameByMapping

The following sections provide information about the getSubscriberNameByMapping operation:

Syntax

The getSubscriberNameByMapping syntax is as follows:

```
ReturnCode* getSubscriberNameByMapping(char* argMapping,
                                       MappingType argMappingType,
                                       char* argDomain)
```

Description

The getSubscriberNameByMapping method finds a subscriber name according to a mapping and a domain.

Parameters

argMapping—See the description in the "Network ID Mappings" section on page 2-7.

argMappingType—See the description in the "Network ID Mappings" section on page 2-7.

argDomain—Name of the domain to which the subscriber belongs.

The operation fails if either of the following conditions exists:

- The domain is null, but the subscriber exists in the database and belongs to a domain.
- The specified domain is incorrect.

Return Value

The return value is a pointer to a ReturnCode structure with a String (char*) type:

- Subscriber name—If a subscriber record was found.
- NULL—If no subscriber record with the supplied mapping could be found in the Cisco Service Control Subscriber Manager database.

Error Codes

The following list presents error codes that this method might return:

- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

getSubscriberNames (all)

The following sections provide information about the getSubscriberNames (all) operation:

Syntax

The getSubscriberNames (all) syntax:

```
ReturnCode* getSubscriberNames(char* argFirstName,
                               int argAmount)
```

Description

The getSubscriberNames (all) method retrieves a bulk of subscriber names from the Cisco Service Control Subscriber Manager database, starting with argFirstName**,** followed by the next argAmount subscribers (in alphabetical order).

If argFirstName is NULL, the first subscriber name (alphabetically) that exists in the Cisco Service Control Subscriber Manager database is used.

Note There is *no* guarantee that the total number of subscribers (in all the bulks) equals the value returned from getNumOfSubscribers at any time. The values differ if some subscribers are added or removed while the bulks are being retrieved.

Parameters

argFirstName—The last subscriber name from the last bulk (the first name to look for). Use NULL to start with the first (alphabetic) subscriber.

argAmount—The limit on the number of subscribers that is returned. If this value is higher than the Cisco Service Control Subscriber Manager limit (1000), the Cisco Service Control Subscriber Manager limit is used.

Note Do *not* retrieve a value of more than 500 subscribers.

Return Value

The return value is a pointer to a ReturnCode structure with a string array (char**) holding a list of subscriber names alphabetically ordered.

The getSubscriberNames (all) method returns all the subscribers found in the Cisco Service Control Subscriber Manager database, starting at the requested subscriber. The array size is limited by the smaller of the values between argAmount and the Cisco Service Control Subscriber Manager limit (1000).

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, “List of Error Codes”.

Example

To receive an alphabetical list of subscriber names:

```
bool hasMoreSubscribers;
char* lastBulkEnd = NULL;
char tmpName[50];
int bulkSize = 100;
```

```
do
{
    ReturnCode* subscribers = smApi.getSubscriberNames(lastBulkEnd,bulkSize);

    hasMoreSubscribers = false;

    if ((isReturnCodeError(subscribers) == false) &&
        (subscribers->type == STRING_ARRAY_T) && (subscribers->u.stringArrayVal != NULL))
    {

        for (int i = 0; i < subscribers->size; i++)
        {
            // do something with subscribers->u.stringArrayVal[i]
        }

        if (subscribers->size == bulkSize)
        {
            hasMoreSubscribers = true;
            strcpy (tmpName, subscribers->u.stringArrayVal[bulkSize - 1]);
            lastBulkEnd = tmpName;
        }
    }
    freeReturnCode(subscribers);

} while (hasMoreSubscribers);
```

getSubscriberNames (filter by property)

The following sections provide information about getSubscriberNames (filter by property):

Syntax

The getSubscriberNames (filter by property) syntax:

```
ReturnCode* getSubscriberNames(char* lastBulkEnd, char* propName, char* propValue,
                               int numOfSubscribers)
```

Description

The getSubscriberNames (filter by property) method retrieves a bulk of subscriber names from the Cisco Service Control Subscriber Manager database, starting with lastBulkEnd, followed by numOfSubscribers (in alphabetical order), and based on the property name and property value.

If lastBulkEnd is NULL, the first subscriber name (alphabetically) that exists in the Cisco Service Control Subscriber Manager database is used.

Note There is *no* guarantee that the total number of subscribers (in all the bulks) equals the value returned from getNumOfSubscribers at any time. The values might differ if some subscribers are added or removed when the bulks are being retrieved.

Parameters

lastBulkEnd—The last subscriber name from the last bulk (the first name to look for). Use NULL to start with the first (alphabetic) subscriber.

propName—The property name of the character string array used to filter subscriber names.

propValue—The property value of the character string array used to filter subscriber names.

numOfSubscribers—The limit on the number of subscribers that is returned. If this value is higher than the Cisco Service Control Subscriber Manager limit (1000), the Cisco Service Control Subscriber Manager limit is used.

Note Do *not* retrieve a value of more than 500 subscribers.

Return Value

The return value is a pointer to a ReturnCode structure with a string array (char**) holding a list of alphabetically ordered subscriber names that match the property name and its provided value.

The getSubscriberNames (filter by property) method returns all the subscribers found in the Cisco Service Control Subscriber Manager database, starting at the requested subscriber. The array size is limited by the smaller of the values between numOfSubscribers and the Cisco Service Control Subscriber Manager limit (1000).

Error Codes

The following list presents the error codes that this method might return:

- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, “List of Error Codes”.

getSubscriberNamesInDomain

The following sections provide information about the getSubscriberNamesInDomain operation:

Syntax

The getSubscriberNamesInDomain syntax:

```
ReturnCode* getSubscriberNamesInDomain(char* argFirstName,
                                       int argAmount,
                                       char* argDomain)
```

Description

The getSubscriberNamesInDomain method retrieves subscribers that are associated with the specified domain from the Cisco Service Control Subscriber Manager database.

The getSubscriberNamesInDomain operation functions in the same way as the getSubscriberNames (all) operation described in the "getSubscriberNames (all)" section on page 3-26.

Parameters

argFirstName—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argAmount—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argDomain—The name of a subscriber domain that exists in the Cisco Service Control Subscriber Manager domain repository.

Return Value

The return value is an alphabetically ordered array of subscriber names that belong to the specified domain.

For more information, see the "Return Value" section on page 3-27 of the getSusbcriberNames (all) operation.

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _DOMAIN_NOT_FOUND
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

getSubscriberNamesWithPrefix

The following sections provide information about the getSubscriberNamesWithPrefix operation:

Syntax

The getSubscriberNamesWithPrefix syntax is as follows:

```
ReturnCode* getSubscriberNamesWithPrefix(char* argFirstName,
                                         int argAmount,
                                         char* argPrefix)
```

Description

The getSubscriberNamesWithPrefix method retrieves subscribers from the Cisco Service Control Subscriber Manager database whose names begin with a specified prefix.

The function of this operation is the same as the getSubscriberNames (all) operation described in the "getSubscriberNames (all)" section on page 3-26.

Parameters

argFirstName—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argAmount—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argPrefix—Case-sensitive string that marks the prefix of the required subscriber names.

Return Value

The return value is an alphabetically ordered array of subscriber names that start with the required prefix. See the "Return Value" section on page 3-27 of the getSusbcriberNames (all) operation.

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

getSubscriberNamesWithSuffix

The following sections provide information about the getSubscriberNamesWithSuffix operation:

Syntax

The getSubscriberNamesWithSuffix syntax is as follows:

```
ReturnCode* getSubscriberNamesWithSuffix(char* argFirstName,
                                         int argAmount,
                                         char* argSuffix)
```

Description

The getSubscriberNamesWithSuffix method retrieves subscribers from the Cisco Service Control Subscriber Manager database whose names end with the specified suffix.

The function of this operation is the same as the getSubscriberNames (all) operation as described in the "getSubscriberNames (all)" section on page 3-26.

Parameters

argFirstName—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argAmount—See the description in the "Parameters" section on page 3-27 of the getSusbcriberNames (all) operation.

argSuffix—Case-sensitive string that marks the suffix of the required subscriber names.

Return Value

The return value is an alphabetically ordered array of subscriber names that end with the required suffix. See the "Return Value" section on page 3-27 of the getSusbcriberNames operation.

Error Codes

The following list provides the error codes that this method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

getDomains

The following sections provide information about the getDomains operation:

Syntax

The getDomains syntax is as follows:

```
ReturnCode* getDomains()
```

Description

The getDomains method provides a list of current subscriber domains in the Cisco Service Control Subscriber Manager domain repository.

Return Value

A pointer to a ReturnCode structure with a string array (char**) holding a complete list of subscriber domain names in the Cisco Service Control Subscriber Manager.

Error Codes

None.

setPropertiesToDefault

The following sections provide information about the setPropertiesToDefault operation:

Syntax

The setPropertiesToDefault syntax is as follows:

```
ReturnCode* setPropertiesToDefault(char* argName,
                                   char** argPropertyKeys,
                                   int argPropertySize)
```

Description

The setPropertiesToDefault method resets the specified application properties of a subscriber. If an application is installed, the relevant application properties are set to the default value of the properties according to the currently loaded application information. If an application is not installed, an ERROR_CODE_ILLEGAL_STATE error code is returned.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

argPropertyKeys—See the description in the "Subscriber Properties" section on page 2-10.

argPropertySize—Size of the argPropertyKeys array.

Return Value

The return value is a pointer to a ReturnCode structure with a void type.

Error Codes

The following list provides the error codes that the setPropertiesToDefault method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

removeCustomProperties

The following sections provide information about the removeCustomProperties operation:

Syntax

The removeCustomProperties syntax is a s follows:

```
ReturnCode* removeCustomProperties(char* argName,
                                   char** argCustomPropertyKeys,
                                   int argCustomPropertySize)
```

Description

The removeCustomProperties method resets the specified custom properties of a subscriber.

Parameters

argName—See the description in the "Subscriber Name Format" section on page 2-7.

argCustomPropertyKeys—See the description in the "Custom Properties" section on page 2-10.

argCustomPropertySize—Size of the argCustomPropertyKeys array.

Return Value

The return value is a pointer to a ReturnCode structure with a void type.

Error Codes

The following list provides the error codes that removeCustomProperties method might return:

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

For a description of error codes, see Appendix A, "List of Error Codes".

C++ setLogger Method

The following sections provide information about the C++ setLogger method operation:

Syntax

```
The C++ setLogger syntax is a s follows:

void setLogger(Logger *argLogger)
```

Description

The C++ setLogger method sets an implementation of the abstract Logger class. Use this method to integrate the Cisco Service Control Subscriber Manager API log messages with the host application log.

Parameters

argLogger—Implementation of the abstract Logger class.

Return Value

None.

C++ init Method

The following sections provide information about the C++ init method operation:

Syntax

The C++ init syntax is as follows:

```
Bool init(int argSupportedThreads,
          int argThreadPriority,
          Uint32 argBufferSize,
          Uint32 argKeepAliveDuration,
          Uint32 argConnectionTimeout,
          Uint32 argReconnectTimeout)
```

Description

The C++ init Method configures and initializes the API.

Note This method must be called before performing any operation of the C++ API.

Parameters

argSupportedThreads—Number of threads the API should support.

argThreadPriority—Priority for the PRPC protocol network thread.

argBufferSize—Internal buffer size (set 2,000,000 bytes as default value).

argKeepAliveDuration—Hint about the desired delay between PRPC protocol keepalive messages (set 10 seconds as default value).

argConnectionTimeout—Hint about the desired timeout on a non-responding PRPC protocol connection (set 20 seconds as default value).

argReconnectTimeout—Timeout after which the API attempts to re-establish the connection to Cisco Service Control Subscriber Manager.

Return Value

The return value is a Boolean value:

- TRUE—Success
- FALSE—Fail

Example

```
SmBlockingApi bapi;
bool success = bapi.init(10,
                         0,
                         2000000, //default
                         10,      //default
                         20,      //default
                         0);      //default (no reconnect)
```

C SMB_init Function

The following sections provide information about the C SMB_init function:

Syntax

The C SMB_init function syntax is as follows:

```
SMB_HANDLE SMB_init (int argSupportedThreads,
                     int argThreadPriority,
                     Uint32 argBufferSize,
                     Uint32 argKeepAliveDuration,
                     Uint32 argConnectionTimeout)
```

Description

The C SMB_init function allocates, configures, and initializes the API.

Note This method must be called before performing any operation of the C API.

Parameters

argSupportedThreads—Number of threads the API should support.

argThreadPriority—Priority for the PRPC protocol network thread.

argBufferSize—Internal buffer size (set 2,000,000 bytes as default value).

argKeepAliveDuration—Hint about the desired delay between PRPC protocol keepalive messages (set 10 seconds as default value).

argConnectionTimeout—Hint about the desired timeout on a non-responding PRPC protocol connection (set 20 seconds as default value).

Return Value

The return values is an SMB_HANDLE handle to the API. A NULL handle means the initialization has failed.

Example

```
SMB_HANDLE api;
// initialize an API
api = SMB_init(10, // 10 threads
               0,
               300000, // 3,000,000 bytes
               10,     // default
               30);    // 30 sec connection timeout
```

C SMB_release Function

The following sections provide information about the C SMB_release Function operation:

Syntax

The C SMB_release function syntax is as follows:

```
void SMB_release(SMB_HANDLE argApiHandle)
```

Description

The C SMB_release function releases the resources used by the API. This function must be called at the end of the use of the API.

Parameters

argApiHandle—API handle received by using the SMB_init function.

Return Value

None.

setReconnectTimeout

The following sections provide information about the setReconnectTimeout operation:

Syntax

The setReconnectTimeout syntax is as follows:

```
void setReconnectTimeout(Uint32 reconnectTimeout)
```

Description

The setReconnectTimeout method sets the reconnection timeout.

When the connection to the Cisco Service Control Subscriber Manager is down, the API attempts to re-establish the connection after “reconnection timeout” seconds.

Parameters

reconnectTimeout—The reconnection timeout in seconds.

Return Value

None.

setName

The following sections provide information about the setName operation:

Syntax

The setName syntax is as follows:

```
void setName(char *argName)
```

Description

The setName method sets the name of the API. This name serves as a unique identifier for the API-Cisco Service Control Subscriber Manager connection. The setName function should be called before calling the connect method.

Parameters

argName—The name of the API,

Return Value

None.

connect

The following sections provide information about the connect operation:

Syntax

The connect syntax is as follows:

```
bool connect(char* argHostName, Uint16 argPort = 14374)
```

Description

The connect method attempts to establish a PRPC protocol connection to the Cisco Service Control Subscriber Manager.

Parameters

argHostName—IP address of the Cisco Service Control Subscriber Manager or the name of the host.

argPort—TCP port to be used to connect the Cisco Service Control Subscriber Manager (default is 14374).

Return Value

The return value is a Boolean value:

- TRUE—Success
- FALSE—Fail

disconnect

The following sections provide information about the disconnect operation:

Syntax

The disconnect syntax is as follows:

```
bool disconnect()
```

Description

The disconnect method attempts to terminate the PRPC protocol connection to the Cisco Service Control Subscriber Manager.

Return Value

The return value is a Boolean value:

- TRUE—Success
- FALSE—Fail

isConnected

The following sections provide information about the isConnected operation:

Syntax

The isConnected syntax is as follows:

```
bool isConnected();
```

Description

The isConnected method checks whether the PRPC protocol connection to the Cisco Service Control Subscriber Manager is operating.

Return Value

The return value is a Boolean value:

- TRUE—The connection is up.
- FALSE—The connection is down.

Blocking API C++ Code Examples

This section provides two code examples:

Getting Number of Subscribers

The following example prints to standard output (stdout) the total number of subscribers in the Cisco Service Control Subscriber Manager database and the number of subscribers in each subscriber domain:

```
#include "SmApiBlocking.h"
#include <stdio.h>

int main(int argc, char* argv[])
{
    SmApiBlocking bapi;
    //initiation
    bapi.init();
    bapi.setReplyTimeout(300000); //set timeout for 5 minutes
    bapi.connect(argv[1]); // connect to the SM
    //operations
    ReturnCode* domains = bapi.getDomains();
    ReturnCode* totalSubscribers=bapi.getNumberOfSubscribers();
    if ((isReturnCodeError(domains) == false) &&
                        (isReturnCodeError(totalSubscribers) == false))
    {
        printf("number of susbcribers in the database:\t\t %d\n",
                                    totalSubscribers->u.intVal);
        for (int i=0; i<domains->size; i++)
        {
            ReturnCode* numberOfSusbcribersInDomain=
                            bapi.getNumberOfSubscribersInDomain(
                                domains->u.stringArrayVal[i]);
            if (isReturnCodeError(numberOfSusbcribersInDomain) == false)
            {
                printf("number of susbcribers domain %s:\t\t%d\n",
                       domains->u.stringArrayVal[i],
                       numberOfSusbcribersInDomain->u.intVal);
            }
            freeReturnCode (numberOfSusbcribersInDomain);
        }
    }
    freeReturnCode (domains);
    freeReturnCode (totalSubscribers);
    //finalization
    bapi.disconnect();
    return 0;
}
```

Adding a Subscriber, Printing Information, and Removing a Subscriber

The following program adds an IPv4 subscriber to the subscriber database, retrieves the subscriber information, prints it to standard output (stdout), and removes the subscriber from the subscriber database:

```
#include "SmApiBlocking.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
    checkArguments(argc,argv);
    SmApiBlocking bapi;
    //initiation
    bapi.init();
    bapi.setReplyTimeout(10000);   //set timeout for 10 seconds
    bapi.connect(argv[1]);        // connect to the SM
    //add subscriber
    printf("adding subscriber to SM\n");
    MappingType type = IP_RANGE;
    char* customKey = "custom-key";
    char* customVal = "10";
    ReturnCode* ret = bapi.addSubscriber(
                                        argv[2],    // name
                                        &(argv[3]), // mapping
                                        &type,      // mapping type
                                        1,          // one mapping
                                        &(argv[4]), // property key
                                        &(argv[5]), // property value
                                        1,          // number of properties
                                        &customKey, // custom property key
                                        &customVal, // custom property value
                                        1,          // number of custom properties
                                        argv[6]);   // domain
    freeReturnCode (ret);

    //Print subscriber
    printf("Printing subscriber:\n");
    ReturnCode* subfields = bapi.getSubscriber(argv[1]);
    if (isReturnCodeError(subfields) == false)
    {
        printf("\tname:\t\t%s\n", subfields->u.objectArray[0]->u.stringVal);
        printf("\tmapping:\t%s\n", subfields->u.objectArray[1]->u.stringArrayVal[0]);
        printf("\tdomain:\t\t%s\n", subfields->u.objectArray[3]->u.stringVal);
        printf("\tautologout:\t%d\n", subfields->u.objectArray[8]->u.intVal);
        // Remove subscriber
        printf("removing subscriber from SM\n");
        bapi.removeSubscriber(argv[1]);
    }
    else
    {
        printf("error in subscriber retrieval\n");
    }
    freeReturnCode(subfields);
    //finalization
    bapi.disconnect();
    return 0;
}

void checkArguments(int argc, char* argv[])
{
if (argc != 7)
    {
        printf("usage: AddPrintRemove <SM-address> <subscriber-name> "
```

```
                "<IP mapping> <property-key> <property-value> <domain>");
        exit(1);
    }
}
```

The following program adds an IPv6 subscriber to the subscriber database, retrieves the subscriber information, prints it to standard output (stdout), and removes the subscriber from the subscriber database:

```
#include "SmApiBlocking.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
    checkArguments
    (argc,argv);
    SmApiBlocking bapi;
    //initiation
    bapi.init();
    bapi.setReplyTimeout(10000); //set timeout for 10 seconds
    bapi.connect(argv[1]); // connect to the SM
    //add subscriber
    printf("adding subscriber to SM\n");
    MappingType type = TYPE_IPV6;
    char* customKey = "custom-key";
    char* customVal = "10";
    ReturnCode* ret = bapi.addSubscriber(
    argv[2], // name
    &(argv[3]), // mapping
    &type, // mapping type
    1, // one mapping
    &(argv[4]), // property key
    &(argv[5]), // property value
    1, // number of properties
    &customKey, // custom property key
    &customVal, // custom property value
    1, // number of custom properties
    argv[6]); // domain
    freeReturnCode (ret);
    //Print subscriber
    printf("Printing subscriber:\n");
    ReturnCode* subfields = bapi.getSubscriber(argv[2]);
    if (isReturnCodeError(subfields) == false)
    {
        printf("\tname:\t\t%s\n", subfields->u.objectArray[0]->u.stringVal);
        printf("\tmapping:\t%s\n", subfields->u.objectArray[1]->u.stringArrayVal[0]);
        printf("\tdomain:\t\t%s\n", subfields->u.objectArray[3]->u.stringVal);
        printf("\tautologout:\t%d\n", subfields->u.objectArray[8]->u.intVal);
        // Remove subscriber
        printf("removing subscriber from SM\n");
        bapi.removeSubscriber(argv[2]);
    }
    else
    {
        printf("error in subscriber retrieval\n");
    }
    freeReturnCode(subfields);
    //finalization
    bapi.disconnect();
    return 0;
}

void checkArguments(int argc, char* argv[])
{
    if (argc != 7)
```

```
    {
        printf("usage: AddPrintRemove <SM-address> <subscriber-name> "
        "<IPV6 mapping> <property-key> <property-value> <domain>");
        exit(1);
    }
}
```

Blocking API C Code Examples

This section provides two code examples:

- Getting Number of Subscribers, page 3-45
- Adding a Subscriber, Printing Information, and Removing a Subscriber, page 3-46

Getting Number of Subscribers

The following example prints to standard output (stdout) the total number of subscribers in the Cisco Service Control Subscriber Manager database and the number of subscribers in each subscriber domain:

```
#include "SmApiBlocking_c.h"
#include <stdio.h>

int main(int argc, char* argv[])
{
    //initiation
    SMB_HANDLE bapi = SMB_init(10,0,2000000,10,20);
    if (bapi == NULL)
    {
        // init failure
        return -1;
    }
    SMB_setReplyTimeout(bapi,300000); //set timeout for 5 minutes
    SMB_connect(bapi,argv[1],14374); // connect to the SM
    //operations
    ReturnCode* domains = SMB_getDomains(bapi);
    ReturnCode* totalSubscribers= SMB_getNumberOfSubscribers(bapi);
    if ((isReturnCodeError(domains) == false) &&
                (isReturnCodeError(totalSubscribers) == false))
    {
        printf("number of susbcribers in the database:\t\t %d\n",
                        totalSubscribers->u.intVal);
        for (int i=0; i<domains->size; i++)
        {
            ReturnCode* numberOfSusbcribersInDomain=
            SMB_getNumberOfSubscribersInDomain(bapi, domains->u.stringArrayVal[i]);
            if(isReturnCodeError(numberOfSusbcribersInDomain) == false
            {
                printf("number of susbcribers domain %s:\t\t%d\n",
                       domains->u.stringArrayVal[i],
                       numberOfSusbcribersInDomain->u.intVal);
            }
            freeReturnCode (numberOfSusbcribersInDomain);
        }
    }
    freeReturnCode (domains);
    freeReturnCode (totalSubscribers);
    //finalization
    SMB_disconnect(bapi);
```

```
    SMB_release(bapi);
    return 0;
}
```

Adding a Subscriber, Printing Information, and Removing a Subscriber

The following program adds a subscriber to the subscriber database, retrieves the subscriber information, prints it to standard output (stdout), and removes the subscriber from the subscriber database:

```
#include "SmApiBlocking_c.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
    checkArguments(argc,argv);

    //initiation
    SMB_HANDLE bapi = SMB_init(10,0,2000000,10,20);
    if (bapi == NULL)
    {
        // init failure
        return -1;
    }
    SMB_setReplyTimeout(bapi,10000);  //set timeout for 10 seconds
    SMB_connect(bapi,argv[1], 14374);// connect to the SM
    //add subscriber
    printf("adding subscriber to SM\n");
    MappingType type = IP_RANGE;
    char* customKey = "custom-key";
    char* customVal = "10";
    ReturnCode* ret = SMB_addSubscriber(
                                        bapi,       // handle
                                        argv[2],    // name
                                        &(argv[3]), // mapping`
                                        &type,      // mapping type
                                        1,          // one mapping
                                        &(argv[4]), // property key
                                        &(argv[5]), // property value
                                        1,          // number of properties
                                        &customKey, // custom property key
                                        &customVal, // custom property value
                                        1,          // number of custom properties
                                        argv[6]);   // domain
    freeReturnCode (ret);

    //Print subscriber
    printf("Printing subscriber:\n");
    ReturnCode* subfields = SMB_getSubscriber(bapi,argv[2]);
    if (isReturnCodeError(subfields) == false)
    {
        printf("\tname:\t\t%s\n", subfields->u.objectArray[0]->u.stringVal);
        printf("\tmapping:\t%s\n", subfields->u.objectArray[1]->u.stringArrayVal[0]);
        printf("\tdomain:\t\t%s\n", subfields->u.objectArray[3]->u.stringVal);
        printf("\tautologout:\t%d\n", subfields->u.objectArray[8]->u.intVal);
        // Remove subscriber
        printf("removing subscriber from SM\n");
        SMB_removeSubscriber(bapi,argv[2]);
    }
    else
    {
        printf("error in subscriber retrieval\n");
    }
    freeReturnCode(subfields);
```

```
    //finalization
    SMB_disconnect(bapi);
    SMB_release(bapi);
    return 0;
}

void checkArguments(int argc, char* argv[])
{
    if (argc != 7)
    {
        printf("usage: AddPrintRemove <SM-address> <subscriber-name> "
               "<IP mapping> <property-key> <property-value> <domain>");
        exit(1);
    }
}
```

The following program adds a subscriber to the subscriber database, retrieves the subscriber information, prints it to standard output (stdout), and removes the subscriber from the subscriber database:

```
#include "SmApiBlocking_c.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
    checkArguments(argc,argv);
    //initiation
    SMB_HANDLE bapi = SMB_init(10,0,2000000,10,20);
    if (bapi == NULL)
    {
        // init failure
        return -1;
    }
    SMB_setReplyTimeout(bapi,10000); //set timeout for 10 seconds
    SMB_connect(bapi,argv[1], 14374);// connect to the SM
    //add subscriber
    printf("adding subscriber to SM\n");
    MappingType type = TYPE_IPV6;
    char* customKey = "custom-key";
    char* customVal = "10";
    ReturnCode* ret = SMB_addSubscriber(
    bapi, // handle
    argv[2], // name
    &(argv[3]), // mapping`
    &type, // mapping type
    1, // one mapping
    &(argv[4]), // property key
    &(argv[5]), // property value
    1, // number of properties
    &customKey, // custom property key
    &customVal, // custom property value
    1, // number of custom properties
    argv[6]); // domain
    freeReturnCode (ret);
    //Print subscriber
    printf("Printing subscriber:\n");
    ReturnCode* subfields = SMB_getSubscriber(bapi,argv[2]);
    if (isReturnCodeError(subfields) == false)
    {
        printf("\tname:\t\t%s\n", subfields->u.objectArray[0]->u.stringVal);
        printf("\tmapping:\t%s\n", subfields->u.objectArray[1]->u.stringArrayVal[0]);
        printf("\tdomain:\t\t%s\n", subfields->u.objectArray[3]->u.stringVal);
        printf("\tautologout:\t%d\n", subfields->u.objectArray[8]->u.intVal);
        // Remove subscriber
        printf("removing subscriber from SM\n");
        SMB_removeSubscriber(bapi,argv[2]);
```

```
    }
    else
    {
        printf("error in subscriber retrieval\n");
    }
    freeReturnCode(subfields);
    //finalization
    SMB_disconnect(bapi);
    SMB_release(bapi);
    return 0;
}

void checkArguments(int argc, char* argv[])
{
    if (argc != 7)
    {
        printf("usage: AddPrintRemove <SM-address> <subscriber-name> "
        "<IPV6 mapping> <property-key> <property-value> <domain>");
        exit(1);
    }
}
```