SafeNet Authentication Service (SAS)

WSDL BSIDCA API Developer's Guide

gemalto
security to be free

Document Part Number: 007-012470-001, Rev. F

Release Date: June 2016

Contents

Custom Attributes

DataLayer.Entity.RadiusAttribute

`Class: RadiusAttribute`

* *Summary:* This class defines RADIUS attribute related data.

Data Member	Data Type	Description
AttributeName	String	Name of the attribute
AttributeID	Integer	A unique identifier for the attribute.
ValueName	String	Value name.
Value	Byte Array	Actual value in Byte format.
VendorName	String	Name of the vendor.
VendorID	Inetger	A unique identifier for the vendor.
LsAuthNodes	Array of Strings	Array of Auth Nodes.
format	ENUM	The data format for the particular attribute. The values can be: text, // UTF-8 String, // Binary ipaddr, // 32 Bit IP Address (IPv4) BE integer, // Int 32 BE date, // UInt 32 BE Epoch ipv6addr, // 16 octets in network byte order

Operations

BSIDCA

BSIDCA

* *Summary:* SafeNet Management web API. Used for connecting and managing accounts using operator/account manager credentials as configured in the console. This web service requires session state to function.

BSIDCA

Connect(System.String,System.String,System.String@,System.String)

* *Summary:* Connects to the BSIDCA. Successfully connecting will establish a management session which will allow you to use the rest of the API. Multiple calls may be required in the case of outer window authentications, challenges or PIN changes.
* *Param:* **OperatorEmail:** Operator/Account manager e-mail to connect with.
* *Param:* **OTP:** OTP or Password to connect with.
* *Param:* **challenge:** Challenge message passed out if the return code requires one.
* *Param:* **validationCode:** Optional e-mail validation code if the operator in use has not had their e-mail address validated.
* *Returns:*

AUTH_FAILURE	Authentication Failed
AUTH_SUCCESS	Authentication Success
CHALLENGE	A challenge has been returned in the challenge argument. Call this function again with the response in the OTP argument.
SERVER_PIN_PROVIDED	The server has initiated a PIN change. Your new PIN is in the challenge argument. Authenticate again using this new PIN.
USER_PIN_CHANGE	You are required call this function again providing a new PIN.
OUTER_WINDOW_AUTH	Your token is out of sync. You must call this function again with the next OTP.
CHANGE_STATIC_PASSWORD	You must call this function again and provide a new static password.
STATIC_CHANGE_FAILED	Password change failed.
PIN_CHANGE_FAILED	PIN change failed

BSIDCA

PingConnection()

* *Summary:* Checks the server to see if you have a valid session.
* *Returns:*

True if you have a valid connection

BSIDCA

ActivateToken(System.String,System.Boolean,System.String,System.String)

* *Summary:* Activates a suspended token and optionally forces a PIN change/resets the current PIN.
* *Param:* **serial:** Token serial number to activate
* *Param:* **changePinOnFirstUse:** True if the user is required to change their PIN during the next authentication.
* *Param:* **newPIN:** Optional argument to set a new PIN for the token.
* *Param:* **organization:** Account
* *Returns:*

pinSuccess	Activating the token and setting new PIN for the token has successfully completed.
deletestaticFail	Cannot delete assigned static password, so activating token has failed.
newpinFail	Activating the token was successful, but setting new PIN for the token has failed.
fail	Activating the token has failed for an unknown reason.
success	The token has been activated.
tokenNotFound	
tokenNotAssigned	

BSIDCA

AssignToken(System.String,System.String,System.String)

* *Summary:* Assigns a token to a user.
* *Param:* **userName:** User to assign the token to.
* *Param:* **serial:** Serial of the token to assign.
* *Param:* **organization:** Account
* *Returns:*

Success	
Fail	Unknown failure
AlreadyHasCustomCR	The user already has a challenge response token that is incompatible with the token you are trying to assign.
AlreadyHasSMSCR	The user already has a challenge response SMS token that is incompatible with the token you are trying to assign.
ConflictingToken	The user already has a token that is not compatible with the token you are trying to assign.

BSIDCA

GetSimpleAuthMetrics(System.String,System.String,System.String)

* *Summary:* Gets the total successes and fails for a user/token.
* *Param:* **userName:** User name
* *Param:* **serial:** Token serial
* *Param:* **organization:** Account
* *Returns:*

```
Integer array that contains { totalPasses, totalFailures, total }
```

BSIDCA

GetTokensByOwner(System.String,System.String)

* *Summary:* Gets a list of serial numbers of all tokens assigned to a user.
* *Param:* **userName:** User name
* *Param:* **organization:** Account
* *Returns:*

```
Token serial numbers.
```

BSIDCA

GetTokens(System.Nullable{DataLayer.Entity.Token.TokenState},System.Nullable{DataLayer.Entity.Token.TokenType},System.String,System.String,System.String,System.Int32,System.Int32)

* *Summary:* Gets a table of tokens based on the provided search criterial.
* *Param:* **state:** Required token state. (Optional)
* *Param:* **type:** Required token type. (Optional)
* *Param:* **serial:** Serial number. (Wildcards supported, optional)
* *Param:* **container:** Container the tokens are in. (Optional)
* *Param:* **organization:** Account.
* *Param:* **startRecord:** First record to return.
* *Param:* **pageSize:** Number of records to return after first.
* *Returns:*

```
Table of token information. Columns are:

        Serial

        state           (as Text)

        stateSetDate

        userid          (if assigned)

        orgName

        type            (as Text)

        container

        rented          (boolean)

        hardwareInit    (boolean, true if it can be initialized in the
                        console)
```

```
assignable      (boolean, true if the token can be directly assigned
                without enrollment)

ice             (boolean, true if the tokens is an ICE MP)

stateInt        (token state as an int. Directly maps to the token
                state input of this function)
```

BSIDCA

GetTotalTokens(System.Nullable{DataLayer.Entity.Token.TokenState},System.Nullable{DataLayer.Entity.Token.TokenType},System.String,System.String,System.String)

* *Summary:* Gets the token number of tokens that match the given filter. Used with GetTokens to assist with determining the number of tokens you want returned.
* *Param:* **state:** Required token state. (Optional)
* *Param:* **type:** Required token type. (Optional)
* *Param:* **serial:** Serial number. (Wildcards supported, optional)
* *Param:* **container:** Container the tokens are in. (Optional)
* *Param:* **organization:** Account.
* *Returns:*

```
Number of tokens that match the search criteria.
```

BSIDCA

GetTokenFile(System.String,System.String,System.String@)

* *Summary:* Gets a token file. Used for issuing MP tokens that are assigned to users. NOTE: Calling this function will re-initialize the MP on the server.
* *Param:* **serial:** Serial number of the token to download.
* *Param:* **organization:** Account
* *Param:* **fileName:** Will return the name that the file should have. If so desired, another name can be used, but the extension must be kept.
* *Returns:*

```
Token file.
      To be used, the user will require the initial PIN (or transport PIN for tokens
with no PIN).
```

BSIDCA

GetToken(System.String,System.String)

* *Summary:* Get a token.
* *Param:* **serial:** Serial number of the token.
* *Param:* **organization:** Account
* *Returns:*

```
The token object
```

BSIDCA

GetFixSerial(System.String,System.String,System.String)

* *Summary:* Converts a user input serial number into a server serial number using any custom type tokens that a user has as a reference point. Used when asking a user to input a serial number into a self service style web page.
* *Param:* **userName:** User Name of the user that is using the self service style page.
* *Param:* **serial:** The serial as input by the user.
* *Param:* **organization:** Account
* *Returns:*

```
The serial number as formatted to match any custom token the user has.
```

BSIDCA

MoveTokens(System.Collections.Generic.List{System.String},System.String,System.String)

* *Summary:* Move tokens from one container to another. Assigned tokens can not be moved.
* *Param:* **serials:** List of serial numbers to be moved to the new container.
* *Param:* **newContainer:** Container to move the tokens to.
* *Param:* **organization:** Account
* *Returns:*

```
List of results, one per token, in the same order as the list of serials provided.
      Values are:

         Moved
         CantMoveOwned     (Token is assigned and cannot be moved.)
         Failed
```

BSIDCA

SuspendToken(System.String,ControlLibrary.TokenControl.suspendSelection,System.String,System.String,System.Boolean,System.String,System.Int32)

* *Summary:* Suspends a user's token and optionally assigns a temporary static password.
* *Param:* **serial:** Serial of the token to suspend
* *Param:* **selection:** Action you are taking: No_Static_Password, Accept_LDAP_Password (Direct LDAP Only), Set_Temp_Static_Password
* *Param:* **comment:** Message to attach to the token.
* *Param:* **tempPassword:** Temporary password if Set_Temp_Static_Password was specified.
* *Param:* **changePasswordOnFirstUse:** True to force the user to change the password on first use.
* *Param:* **organization:** Account
* *Param:* **passwordExpiryInMinutes:** Number of minutes the password will be valid for. 0 for no expiry.
* *Returns:*

BSIDCA

GetTokenTemplateForToken(System.String,System.String)

* *Summary:* Get the template settings for a token.
* *Param:* **serial:** Serial number of the token.
* *Param:* **organization:** Account
* *Returns:*

```
Token template that describes the function of the token.
```

BSIDCA

ImportTokens(System.Byte[],System.Boolean,System.Boolean,System.String,System.String)

* *Summary:* Import a BTK file of tokens. Can optionally apply the server side PIN settings to all tokens with no PIN. Can optionally overwrite all existing tokens of the same serial number.
* *Param:* **btkFileContents:** The contents of the BTK file.
* *Param:* **mergeServerSidePINSettings:** True to use the current token template server PIN settings for all tokens that have no PIN in the BTK file.
* *Param:* **overwriteExistingTokens:** True to overwrite any tokens that already exist with the same serial number.
* *Param:* **destinationContainer:** The container to import into.
* *Param:* **organization:** Account
* *Returns:*

```
List of import results. One row per token that was in the BTK file.
      Each import result contains:
      Serial
      TypeName (Name of the token type)
      Result
         Added             New token was added
         Updated           Existing token was updated
         Failed            Token could not be loaded
         Purged            Existing token was deleted and the replacement token
                           could not be added.
         AlreadyExists     This token already exists in the system and
                           overwrite was false.
```

BSIDCA

RemoveToken(System.String,System.String)

* *Summary:* Remove a token from the system. This only applies to hardware tokens that you own.
* *Param:* **serial:** Serial of the token to remove.
* *Param:* **organization:** Account
* *Returns:*

```
True if successful
```

BSIDCA

GetServerPINPolicy(System.String)

* *Summary:* Get the server side PIN policy for tokens.
* *Param:* **organization:** Account
* *Returns:*

```
Server PIN Policy
```

BSIDCA

GetTokenChallenge(System.String,System.String,System.String,System.String@)

* *Summary:* Gets a challenge to use for resyncing a token.
* *Param:* **serial:** Serial number of the token.
* *Param:* **userName:** User the token is assigned to.
* *Param:* **organization:** Account
* *Param:* **state:** The challenge state.
* *Returns:*

```
Challenge String
```

BSIDCA

ResyncToken(System.String,System.String,System.String,System.String,System.String)

* *Summary:* Resyncs a token against a given challenge. For Timebased, OATH or SecurID tokens, pass in one OTP in the challenge, and the next OTP in the OTP. This does not work for RADIUS tokens.
* *Param:* **serial:** Serial of the token to resync.
* *Param:* **userName:** User the token is assigned to.
* *Param:* **challenge:** Challenge for quicklog tokens. First OTP for time based, OATH or SecurID tokens.
* *Param:* **OTP:** OTP for quicklog tokens. Second OTP for time based, OATH or SecurID tokens.
* *Param:* **organization:** Account
* *Returns:*

```
success
fail
error
```

BSIDCA

TestToken(System.String,System.String,System.String,System.String,System.String@,System.String@)

* *Summary:* Function to test tokens. If the return code is not success or failure, all subsequent calls must include the state that was returned from the first call. Note that this API call is meant for testing purposes only, and not intended for regular production authentications. The Management API is not provided with the same level of service availability as authentication services.
* *Param:* **userName:** User name to test.
* *Param:* **organization:** Account
* *Param:* **serial:** Serial number of the token to test.
* *Param:* **OTP:** OTP to test.
* *Param:* **state:** Request state. Use null on new tests. If presented with a challenge/PIN change/outer window, you must pass back the value that was returned on the first call.
* *Param:* **challenge:** Challenge message for return values that require it.
* *Returns:*

```
AUTH_FAILURE            Authentication Failed

AUTH_SUCCESS            Authentication Success

CHALLENGE               A challenge has been returned in the
                        challenge argument. Call this function again
                        with the response in the OTP argument.

SERVER_PIN_PROVIDED     The server has initiated a PIN change. Your
                        new PIN is in the challenge argument.
                        Authenticate again using this new PIN.

USER_PIN_CHANGE         You are required call this function again
                        providing a new PIN.

OUTER_WINDOW_AUTH       Your token is out of sync. You must call
                        this function again with the next OTP.

CHANGE_STATIC_PASSWORD  You must call this function again and
                        provide a new static password.
```

```
STATIC_CHANGE_FAILED    Password change failed.

PIN_CHANGE_FAILED       PIN change failed.
```

BSIDCA

GetChallengeImage(System.String,System.String)

* *Summary:* Returns a JPG image for custom token challenges (if applicable).
* *Param:* **userName:** User to get the challenge for.
* *Param:* **organization:** Account
* *Returns:*

```
A JPG image if the user has a token that returns challenge images.
```

BSIDCA

RevokeToken(System.String,System.String,System.String,ControlLibrary.TokenControl.RevokeSelection,System.Boolean,System.String)

* *Summary:* Revoke a token from a user.
* *Param:* **serial:** Serial number of the token. (Serial of "0" for static passwords)
* *Param:* **userName:** User to revoke the token from.
* *Param:* **comment:** Message to attach to the token.
* *Param:* **revokeMode:** One of: ReturntoInventory_Initialized, ReturntoInventory_NotInitialized, Faulty, Lost
* *Param:* **revokeStaticPassword:** True to also revoke the user's static password.
* *Param:* **organization:** Account
* *Returns:*

```
successRevokeTemp  Successfully revoked a static password

failRevokeTemp     Failed to revoke a static password

BothSuccess        Revoked the token and removed the static password

SuccessFail        Revoked the token but failed to revoke the static
                   password

Success            Revoked the token

Fail

LastAuth           User is an operator or account manager. Their last
                   authentication method cannot be removed.
```

BSIDCA

SetTokenPIN(System.String,System.String,System.Boolean,System.String,System.Boolean,System.String)

* *Summary:* Set a token's server side PIN, optionally requiring a PIN change.
* *Param:* **serial:** Serial of the token to set the PIN for
* *Param:* **newPIN:** The new PIN
* *Param:* **changeOnFirstUse:** True if the user much change it on first use
* *Param:* **comment:** Comment to attach to the token
* *Param:* **notifySMSUsers:** True if you want to notify an SMS token user that their PIN has changed
* *Param:* **organization:** Account

* *Returns:*

```
success

fail

complexityFail    New PIN to set does not meet the server side PIN
                  complexity requirements.

error
```

BSIDCA

GetRandomTokenPIN(System.String)

* *Summary:* Get a random server side PIN that complies with the current server side PIN policy.
* *Param:* **organization:** Account
* *Returns:*

```
The random PIN
```

BSIDCA

RequestSMS(System.String,System.String,System.String)

* *Summary:* Used to request that a new SMS message be sent to a user. This requires their PIN for validation.
* *Param:* **userName:** User the SMS is for
* *Param:* **PIN:** PIN of the SMS token
* *Param:* **organization:** Account
* *Returns:*

```
True if an SMS message has been sent
```

BSIDCA

AddUser(DataLayer.Entity.User,System.String)

* *Summary:* Adds a new user to the given organization. User name must be unique in the organization to which it is added.
* *Param:* **user:** User object to add
* *Param:* **organization:** Account
* *Returns:*

```
True if the user was added.
```

BSIDCA

AddUserWithAlias(DataLayer.Entity.User,System.Collections.Generic.List{System.String},System.String)

* *Summary:* Adds a new user with alias to the given organization. User name must be unique in the organization to which it is added.
* *Param:* **user:** User object to add
* *Param:* **Alias:** Alias
* *Param:* **organization:** Account
* *Returns:*

```
True if the user was added.
```

BSIDCA

GetUser(System.String,System.String)

* *Summary:* Gets a user.
* *Param:* **userName:** User name of the user to get.
* *Param:* **organization:** Account
* *Returns:*

```
The user
```

BSIDCA

GetUserInOperatorOrganization(System.String)

* *Summary:* Gets a user in the same organization as the current operator.
* *Param:* **userName:** User name of the user to get.
* *Returns:*

```
The user
```

BSIDCA

GetUserAlias(System.String,System.String)

* *Summary:* Gets a user's alias.
* *Param:* **userName:** User name of the user to get.
* *Param:* **organization:** Account
* *Returns:*

```
The user's alias
```

BSIDCA

GetUsers(System.String,System.String,DataLayer.Entity.AuthMethod,System.String,System.Int32,System.Int32,System.String)

* *Summary:* Gets a table of user information matching the provided filters.
* *Param:* **userName:** User Name (Wild card support, optional)
* *Param:* **lastName:** Last Name (Wild card support, optional)
* *Param:* **authMethod:** Authentication method: Any, Token, Password, External Credentials, Token Or Password, None
* *Param:* **container:** Container (Wild card support, optional)
* *Param:* **firstRecord:** First record to return
* *Param:* **pageSize:** Number of records to return
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      Lastname
      Firstname
      userid
      custom1
      authmethod
      attributes
      authstate
      container
      isDormant
```

BSIDCA

UpdateUser(System.String,DataLayer.Entity.User,System.String)

* *Summary:* Updates an existing user in the system.
* *Param:* **userName:** User name of the user to update. If changing the user name, this is the old user name. New user name is in the user object
* *Param:* **user:** Updated user
* *Param:* **organization:** Account
* *Returns:*

```
True if updated
```

BSIDCA

UpdateUserWithAlias(System.String,DataLayer.Entity.User,System.Collections.Generic.List{System.String},System.String)

* *Summary:* Updates an existing user with alias in the system.
* *Param:* **userName:** User name of the user to update. If changing the user name, this is the old user name. New user name is in the user object
* *Param:* **user:** Updated user
* *Param:* **Alias:** Updated use' alias
* *Param:* **organization:** Account
* *Returns:*

```
True if updated
```

BSIDCA

RemoveUser(System.String,System.String,ControlLibrary.TokenControl.revokeSelection,System.String)

* *Summary:* Removes an existing user and revokes their tokens.
* *Param:* **userName:** User name of the user to remove
* *Param:* **organization:** Account
* *Param:* **tokenOption:** One of: ReturntoInventory_Initialized, ReturntoInventory_NotInitialized, Faulty, Lost
* *Param:* **comment:** Message to attach to the tokens that are being revoked
* *Returns:*

```
            Deleted
            FailedToDeassignTokens
            FailedToRemoveUser
            UnknownError
```

BSIDCA

MoveUsers(System.Collections.Generic.List{System.String},System.String,System.String)

* *Summary:* Moves user and their tokens to a new container
* *Param:* **userNames:** User names of all users to move to the new container.
* *Param:* **newContainer:** Name of an existing container to move the users to.
* *Param:* **organization:** Account
* *Returns:*

```
True if the users are moved
```

BSIDCA

AddGroup(System.String,System.String,System.String)

* *Summary:* Add a group to the account
* *Param:* **groupName:** Group Name
* *Param:* **description:** Group Description
* *Param:* **organization:** Account
* *Returns:*

```
 True if added
```

BSIDCA

UpdateGroup(System.String,System.String,System.String,System.String)

* *Summary:* Update a group in the account. Applies to writable groups only.
* *Param:* **groupName:** Name of the group as it exists
* *Param:* **newName:** New name of the group
* *Param:* **newDescription:** New description of the group
* *Param:* **organization:** Account
* *Returns:*

```
True if updated
```

BSIDCA

RemoveGroup(System.String,System.String)

* *Summary:* Remove a writable group from the account.
* *Param:* **groupName:** Name of group to remove
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetGroups(System.String,DataLayer.Entity.GroupKind)

* *Summary:* Returns a table of the groups in the system.
* *Param:* **organization:** Account
* *Param:* **kind:** Read Only, Writable or Both
* *Returns:*

```
A table with the columns:
      Groupname
      Groupdescription
      removable
```

BSIDCA

GetUsersForGroup(System.String,System.Boolean,DataLayer.Entity.GroupKind,System.String,System.String,System.String)

* *Summary:* Returns a table of users that are or are not members of a group.
* *Param:* **groupName:** Name of the group
* *Param:* **isAMember:** If they are a member or not
* *Param:* **kind:** Read Only, Writable or Both
* *Param:* **userName:** User Name filter
* *Param:* **lastName:** Last Name filter

* *Param:* **organization:** Account

* *Returns:*

```
A table with the columns:
      Firstname
      Lastname
      Username
      custom1
```

BSIDCA
AddUsersToGroup(System.Collections.Generic.List{System.String},System.String,System.String)
* *Summary:* User becomes a member of the group.
* *Param:* **userNames:** Names of users to add to group
* *Param:* **groupName:** Name of group to add users to
* *Param:* **organization:** Account
* *Returns:*

```
 True if users are members
```

BSIDCA
RemoveUserFromGroup(System.Collections.Generic.List{System.String},System.String,System.String)
* *Summary:* Removes users group membership.
* *Param:* **userNames:** Users to remove from the group
* *Param:* **groupName:** Group name to remove membership from
* *Param:* **organization:** Account
* *Returns:*

```
True if users have their membership removed
```

BSIDCA
AddContainer(System.String,System.String,System.String)
* *Summary:* Add a container to the account
* *Param:* **containerName:** New container Name. Must be unique to the account
* *Param:* **description:** Container description.
* *Param:* **organization:** Account
* *Returns:*

```
 True if container added
```

BSIDCA
UpdateContainer(System.String,System.String,System.String,System.String)
* *Summary:* Update the container
* *Param:* **containerName:** Container to be updated
* *Param:* **newName:** New container name
* *Param:* **newDescription:** New description
* *Param:* **organization:** Account
* *Returns:*

```
True if container updated
```

BSIDCA

RemoveContainer(System.String,System.String)

* *Summary:* Remove a container from an account
* *Param:* **containerName:** Container name to remove
* *Param:* **organization:** Account
* *Returns:*

```
True if container removed
```

BSIDCA

GetContainers(System.String)

* *Summary:* Get a table of container information
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      Container
      description
```

BSIDCA

GetUsersForContainer(System.String,System.String)

* *Summary:* Get a table of users in a container
* *Param:* **containerName:** Container to return users in
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      Lastname
      Firstname
      Username
      custom1
```

BSIDCA

GetRADIUSAttributesForUser(System.String,System.String)

* *Summary:* Get a list of RADIUS attributes associated with a user. Multi value attributes appear as multiple attributes of the same type/ID with different values.
* *Param:* **userName:** User name
* *Param:* **organization:** Account
* *Returns:*

```
RADIUS Attributes
```

BSIDCA

AddRADIUSAttributeToUser(DataLayer.Entity.RadiusAttribute,System.String,System.String)

* *Summary:* Add a RADIUS attribute to a user. To add a multi value attribute, add the attribute once for each value of the attribute. To add an attribute specific to auth nodes, assign the alias name within the **attribute** parameter.
* *Param:* **attribute:** RADIUS Attribute to add
* *Param:* **userName:** User to add the attribute to
* *Param:* **organization:** Account
* *Returns:*

```
True if attribute was added
```

BSIDCA

RemoveRADIUSAttributeFromUser(DataLayer.Entity.RadiusAttribute,System.String,System.String)

* *Summary:* Remove a RADIUS attribute from a user. Exact value and auth node name to remove must be specified.
* *Param:* **attribute:** RADIUS attribute to remove
* *Param:* **userName:** User name
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetRADIUSAttributesForGroup(System.String,System.String)

* *Summary:* Get a list of RADIUS attributes associated with a group. Multi value attributes appear as multiple attributes of the same type/ID with different values.
* *Param:* **groupName:** Group Name
* *Param:* **organization:** Account
* *Returns:*

```
RADIUS Attributes
```

BSIDCA

AddRADIUSAttributeToGroup(DataLayer.Entity.RadiusAttribute,System.String,System.String)

* *Summary:* Add RADIUS attribute to a group. To add a multi value attribute, add the attribute once for each value of the attribute. To add an attribute specific to auth nodes, assign the alias name within the **attribute** parameter.
* *Param:* **attribute:** Attribute to add to the group
* *Param:* **groupName:** Group to add attribute to
* *Param:* **organization:** Account
* *Returns:*

```
True if added
```

BSIDCA

RemoveRADIUSAttributeFromGroup(DataLayer.Entity.RadiusAttribute,System.String,System.String)

* *Summary:* Remove a RADIUS attribute from a group. Exact value and auth node name to remove must be specified.
* *Param:* **attribute:** Attribute to be removed
* *Param:* **groupName:** Group to remove the attribute from
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetRADIUSVendors()

* *Summary:* Gets a list of all vendors for which RADIUS attributes are available.
* *Returns:*

```
List of vendor names
```

BSIDCA

GetRADIUSAttributeForVendor(System.String)

* *Summary:* Get a list of all RADIUS attributes for the vendor.
* *Param:* **vendorName:** Vendor to get attributes for
* *Returns:*

```
 List of attributes supported for a vendor
```

BSIDCA

GetRADIUSAttribute(System.String,System.String)

* *Summary:* Get a RADIUS attribute by vendor name and attribute name.
* *Param:* **vendorName:** Vendor name
* *Param:* **attributeName:** Attribute name
* *Returns:*

```
 RADIUS Attribute
```

BSIDCA

AssignStaticPassword(System.String,System.String,System.Boolean,System.DateTime,System.String)

* *Summary:* Assigns a static password to a user.
* *Param:* **userName:** User to assign the password to
* *Param:* **staticPassword:** The static password
* *Param:* **changeOnFirstUse:** True if the user is required to change the password the first time they authenticate
* *Param:* **revokeDate:** Date that the static password will no longer work.
* *Param:* **organization:** Account
* *Returns:*

```
True if assigning the password has succeeded
```

BSIDCA

RevokeStaticPassword(System.String,System.String)

* *Summary:* Revokes a user's static password
* *Param:* **userName:** User to revoke the password of
* *Param:* **organization:** Account
* *Returns:*

```
True if password revoked
```

BSIDCA

ProvisionUsers(System.Collections.Generic.List{System.String},DataLayer.Entity.ProvisioningEntry.TokenOption,System.String,System.String)

* *Summary:* Provisions a list of users a token of a given class. SMS tokens will be instantly provisioned, all other types will have provisioning tasks added for the users.
* *Param:* **userNames:** Names of the users to provision tokens to
* *Param:* **tokenClass:** The type of token to provision: Software, Oath, SMS, Password, KT, RB, ICE, GOLD, eToken
* *Param:* **description:** Provisioning task description
* *Param:* **organization:** Account
* *Returns:*

List of results. Order is the same as order of user names provided.

FailedToAddToBatch	Failed to add user to provisioning task
EmailSent	Success
SMSSent	Success for SMS tokens
UserHasNoEmail	Can't provision to the user as they have no e-mail address
UserHasNoMobileNumber	Can't provision an SMS token, as they have no mobile number
FailedToSendEmail	Failed to send out the e-mail. Check e-mail settings for the account.
FailedToSendSMS	Failed to send out an SMS. Check SMS settings for the account.
FailedToSendSMS_NoCredits	The account does not have enough SMS credits to send the message.
CouldntGetToken	Couldn't find an available SMS or MP token to use for SMS provisioning.
CouldntAssignToken	User could not be assigned an SMS token.
UserHasActiveToken	Cannot provision password as the user has an active token.

BSIDCA

ProvisionUsersGrIDsureTokens(System.Collections.Generic.List{System.String},System.String,System.String)

* *Summary:* Provisions a list of users GrIDsure tokens.
* *Param:* **userNames:** Names of the users to provision tokens to
* *Param:* **description:** Provisioning task description
* *Param:* **organization:** Account

* *Returns:*

```
List of results. Order is the same as order of user names provided

        FailedToAddToBatch  Failed to add user to provisioning task

        EmailSent           Success

        UserHasNoEmail      Can't provision to the user as they have no e-mail
                            address

        FailedToSendEmail   Failed to send out the e-mail. Check e-mail
                            settings for the account.

        CouldntGetToken     Couldn't find an available SMS or MP token to use
                            for SMS provisioning.

        CouldntAssignToken  User could not be assigned an SMS token.

        UserHasActiveToken  Cannot provision password as the user has an
                            active token.
```

BSIDCA

GetProvisioningTasks(System.String,System.Int32,System.Int32)

* *Summary:* Gets a table of information on provisioning tasks
* *Param:* **organization:** Account
* *Param:* **startRecord:** First record
* *Param:* **numberOfRecords:** Number of records
* *Returns:*

```
A table with the columns:
      Tasked
      Operator
      Startdate
      Count
      tokenoption
      stopdate
```

BSIDCA

GetProvisioningTasksForUser(System.String,System.String,System.Int32,System.Int32)

* *Summary:* Gets a table of information on provisioning tasks
* *Param:* **user:** The user to get provisioning tasks for
* *Param:* **organization:** Account
* *Param:* **startRecord:** First record
* *Param:* **numberOfRecords:** Number of records
* *Returns:*

```
A table with the columns:
      taskid
      operator
      startdate
      count
      tokenoption
      stopdate
```

BSIDCA

GetProvisioningTaskCount(System.String)

* *Summary:* Gets the number of provisioning tasks in the organization.
* *Param:* **organization:** Account
* *Returns:*

```
Number of provisioning tasks in the account
```

BSIDCA

GetProvisioningTasksForUserCount(System.String,System.String)

* *Summary:* Gets the number of provisioning tasks in the organization.
* *Param:* **user:** User to get task count for
* *Param:* **organization:** Account
* *Returns:*

```
Number of provisioning tasks in the account
```

BSIDCA

GetProvisioningTaskDetails(System.Int32,System.String)

* *Summary:* Returns a table of provisioning task details
* *Param:* **taskID:** Task to get details for
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      userid
      username
      startdate
      stopdate
      status
      serialnumber
      description
```

BSIDCA

RemoveProvisioningTask(System.Int32,System.Boolean,System.String)

* *Summary:* Removes an existing provisioning task, optionally sending a notification e-mail to all the users that this affects.
* *Param:* **taskID:** Task ID to remove
* *Param:* **sendNotificationEmail:** True to send a notification e-mail
* *Param:* **organization:** Account
* *Returns:*

BSIDCA

RemoveUsersFromProvisioningTask(System.Int32,System.Collections.Generic.List{System.String},System.Boolean,System.String)

* *Summary:* Removes specific users from a given provisioning task, optionally sending a notification e-mail to them.
* *Param:* **taskID:** Task ID to remove users from
* *Param:* **userNames:** List of user names to remove
* *Param:* **sendNotificationEmail:** True to send a notification e-mail
* *Param:* **organization:** Account
* *Returns:*

```
True if users are removed
```

BSIDCA

UpdateProvisioningStopDate(System.Int32,System.Collections.Generic.List{System.String},System.DateTime,System.String)

* *Summary:* Updates the expiry date of a provisioning task for the given users. Used to extent a provisioning task that users may not have had the time to complete.
* *Param:* **taskID:** Task ID to update
* *Param:* **userNames:** List of user names to update the expiry date for
* *Param:* **newStopDate:** The new expiry date
* *Param:* **organization:** Account
* *Returns:*

```
True if the expiry dates were updated
```

BSIDCA

GetAuthenticationMetrics(System.String,System.String)

* *Summary:* Get authentication metrics.
* *Param:* **userName:** Optional user to filter by
* *Param:* **organization:** Account
* *Returns:*

```
A table with 3 rows (Pass, Fail, All) in the columns:
      result
      today
      weektodate
      lastweek
      monthtodate
      lastmonth
      yeartodate
      lastyear
```

BSIDCA

GetAuthenticationActivity(System.String,System.String,DataLayer.Entity.Authentication.AuthResult,System.Int32,System.Int32)

* *Summary:* Gets a table of authentication activity.
* *Param:* **userName:** User name filter. (Optional)
* *Param:* **organization:** Account
* *Param:* **authResult:** Result filter. (Optional)
* *Param:* **firstRecord:** First record.
* *Param:* **pageSize:** Number of records to retrieve.
* *Returns:*

```
A table with the columns:
      actDate     (Time of the event)
      username    (User it was for)
      ActionText  (What the request was for)
      ResultText  (Result)
      Serial
      sourceIP    (Client IP/Auth Node)
      agentID     (SAS Agent that was used for authentication)
      orgName
      message
```

BSIDCA

ProcessEnrollmentWithHost(System.String,System.String,System.Nullable{System.Int32},System.String,System.Nullable{TV.TokenValidator.ReturnCode}@,System.String@,System.String@,ControlLibrary.Enrollment.ProvisioningInformation@,ControlLibrary.Enrollment.TokenFileInformation@,ControlLibrary.Enrollment.CustomTokenInformation@)

* *Summary:* This function can be used to enroll a provisioned token to a user. To begin, call with the enrollment code. Return codes will prompt for more information for next call or end in a terminating state.
* *Param:* **Code:** Enrollment code provided to the user. This is generally part of the URL in the e-mail that we send out.
* *Param:* **Serial:** Serial of the token. Only required if prompted by this function to get it from the user.
* *Param:* **hostNumber:** Host number for multi host tokens. Provide only if prompted for.
* *Param:* **OTP:** OTP to authenticate with. Only required if prompted to authenticate by this function.
* *Param:* **ResultCode:** Authentication result. See TestToken return codes for their meanings.
* *Param:* **AuthenticationState:** Authentication state. Pass in null unless replying to a continue authentication, where you will pass in what you recieve.
* *Param:* **AuthenticationChallenge:** Challege message from authentication. See TestToken for more details.
* *Param:* **Info:** Provides information about the provisioning request. This information will be used with the return code to process the next step.
* *Param:* **TokenInfo:** Provides a software token and all information pertaining to it. The provided token must be loaded to continue.
* *Param:* **CustomInfo:** Provides information about a custom token including an enrollment image and instructions to present to the user.
* *Returns:*

Error	Process cannot continue. Unexpected input or server issue.
Success	Enrollment has completed and the user can authenticate with their token.
CodeDoesNotExist	The code provided cannot be found. Prompt the user to ensure it has been entered correctly, if they entered it manually.
ProvisioningNotActive	The code entered was found, but it is not available for use at this time.
NotEnoughTokens	The account does not have enough tokens of the type required to process this enrollment.
RequiresSerial	A serial number for a hardware token is required. Call again with the serial number.
InvalidSerial	The serial number provided by the user is not valid for provisioning.
TokenDoesNotExist	The serial number that has been provided does not exist in the system.
TokenNotAssignable	A token cannot be assigned to the user. They may have an existing token that is not compatible.

RequiresAuthentication	Call again with the next OTP from the token.
FailedAuthentication	The OTP provided was not valid. Another attempt can be made.
ContinueAuthentication	A challenge has been provided. Prompt for a response and call again with the response in the OTP field.
TooManyAttempts	There have been too many failed attempts to authenticate. This provisioning task has been locked.
OutsideOfTimeWindow	This provisioning task cannot be used at this time.
SelectSoftwareTarget	Prompt the user to select a token target to install their token to. Targets are available by calling: GetTokenTargets(String code) Select a target by calling: SelectEnrollmentSoftwareType
SendSMS	You must call SendEnrollmentSMS once for each SMS message in the ProvisioningInfo Info. Default values will be used if not overridden in the call to SendEnrollmentSMS
SendEMail	You must call SendEnrollmentEMail once for each e-mail message in the ProvisioningInfo Info. Default values will be used if not overridden in the call to SendEnrollmentEMail
RequiresCustomEnrollment	Present the image and instructions in the CustomInfo and call again with the OTP/Authentication code in the OTP field.
FailedCustomEnrollment	The custom enrollment information provided was not valid. Another attempt can be made.
ChoosePassword	Call again with a password in the OTP field. Must meet the requirements in the ProvisioningInfo Info.
DoesNotMeetRequirements	Password does not meet the requirements in the ProvisioningInfo object. Try again.

ProvisioningLocked	The provisioning code provided is for a locked provisioning task and cannot be used at this time.
ProvisioningExpired	The provisioning code provided is for an expired provisioning task and cannot be used at this time.
ProvisioningCancelled	The provisioning code provided is for a cancelled provisioning task and cannot be used at this time.
ProvisioningCompleted	The provisioning code provided is for an already completed provisioning task and cannot be used at this time.
OutOfBand	The user has been SMS or e-mailed an out of band validation code. Prompt them for this and provide it in the OTP field.
RequiresHost	The user is enrolling a GOLD token and must provide the host number they are attempting to enroll.
OutOfBandNotAvailable	The provisioning task requires out of band validation, but it cannot be processed at this time.

BSIDCA

GetTokenTargets(System.String)

* *Summary:* This function can be used to get a list of available targets for a software token when prompted to select a token target.
* *Param:* **code:** Enrollment code
* *Returns:*

```
List of token targets available for this provisioning code
```

BSIDCA

GetTokenSubTargets(DataLayer.Entity.Token.TokenTargets,System.String)

* *Summary:* This function can be used to get a list of available sub targets for the desired software target.
* *Param:* **target:** Target to check sub targets for.
* *Param:* **code:** ENrollment code
* *Returns:*

```
List of sub targets for a given target type.
```

BSIDCA

SelectEnrollmentSoftwareType(System.String,ControlLibrary.Enrollment.SoftwareType,ControlLibrary.Enrollment.ProvisioningInformation@,ControlLibrary.Enrollment.TokenFileInformation@)

* *Summary:* This function is used to select the kind of deployment to use for software token provisioning.
* *Param:* **enrollmentCode:** Enrollment code
* *Param:* **type:** The token type to select: iPhone, Android, Blackberry, Windows7Phone, MP, MacOSX
* *Param:* **pInfo:** Information about the provision request
* *Param:* **tfInfo:** Information about the token if the target you have selected gives you a token file.
* *Returns:*

Error	Process cannot continue. Unexpected input or server issue.
Success	Enrollment has completed and the user can authenticate with their token.
CodeDoesNotExist	The code provided cannot be found. Prompt the user to ensure it has been entered correctly, if they entered it manually.
TokenDoesNotExist	The serial number that has been provided does not exist in the system.
DoesNotMeetRequirements	Password does not meet the requirements in the ProvisioningInfo object. Try again.
SendSMS	You must call SendEnrollmentSMS once for each SMS message in the ProvisioningInfo Info. Default values will be used if not overridden in the call to SendEnrollmentSMS
SendEMail	You must call SendEnrollmentEMail once for each e-mail message in the ProvisioningInfo Info. Default values will be used if not overridden in the call to SendEnrollmentEMail
RequiresAuthentication	Call again with the next OTP from the token.
TokenNotAssignable	A token cannot be assigned to the user. They may have an existing token that is not compatible.

BSIDCA

SendEnrollmentSMS(System.String,ControlLibrary.Enrollment.MessageType,System.String,System.String,System.String,ControlLibrary.Enrollment.ProvisioningInformation@)

* *Summary:* This function is used to send any SMS messages that you were told to send when either selecting the software token type or while processing the enrollment. The message and mobile are optional and defaults will be used if they are not provided.
* *Param:* **enrollmentCode:** Enrollment code
* *Param:* **type:** Message type you were prompted to send
* *Param:* **message:** Optional override of the default message
* *Param:* **mobileNumber:** Optional override of the users mobile number if permitted.
* *Param:* **shortCodeURL:** URL is a URL of a web site that the user may be redirected to from their mobile device. If you provide this URL, the page indicated must take a post parameter of 'sc' for the short code. This page should then send the contents of the file provided by GetFileForShortCode()
* *Param:* **pInfo:** Infomration about the provisioning task. May contain a subsequent message to send
* *Returns:*

Success	
Failure	
SendSMS	Call this function again with the SMS message in the pInfo parameter
SeedEMail	Call SendEnrollmentEMail with the email message in the pInfo parameter

BSIDCA

SendEnrollmentEMail(System.String,ControlLibrary.Enrollment.MessageType,System.String,System.String,System.String,System.String,ControlLibrary.Enrollment.ProvisioningInformation@)

* *Summary:* This function is used to send any email messages that you were told to send when either selecting the software token type or while processing the enrollment. The message and mobile are optional and defaults will be used if they are not provided.
* *Param:* **enrollmentCode:** Enrollment code
* *Param:* **type:** Message type you were prompted to send
* *Param:* **message:** Optional override of the default message
* *Param:* **mobileNumber:** Optional override of the users email if permitted.
* *Param:* **shortCodeURL:** URL is a URL of a web site that the user may be redirected to from the email they receive. If you provide this URL, the page indicated must take a post parameter of 'sc' for the short code. This page should then send the contents of the file provided by GetFileForShortCode()
* *Param:* **pInfo:** Infomration about the provisioning task. May contain a subsequent message to send
* *Returns:*

Success	
Failure	
SendSMS	Call SendEnrollmentSMS with the SMS message in the pInfo parameter
SendEMail	Call this function again with the email message in the pInfo parameter

BSIDCA

GetFileForShortCode(System.String,System.String,System.String@)

* *Summary:* This function is used to provide the token files for download for any mobile devices. The short code will be passed as a parameter to the web page that uses this function. The resource URL is the web directory in which support files for mobile tokens (such as the Blackberry JAuthenticator.jar) is located. Pass in null to use the default resource URL.
* *Param:* **shortCode:** Short code that was used when the page was requested
* *Param:* **resourceURL:** web directory in which support files for mobile tokens (such as the Blackberry JAuthenticator.jar) is located. Pass in null to use the default resource URL.
* *Param:* **fileName:** File name for the returned file
* *Returns:*

```
File to deliver to the client that is requesting it.
```

BSIDCA

GetSelfEnrollmentPolicy(System.String)

* *Summary:* Get the self enrollment policy that affects the given enrollment request
* *Param:* **enrollmentCode:** Enrollment code
* *Returns:*

```
SelfEnrollment Policy
```

BSIDCA

AddProvisioningRequest(DataLayer.Entity.User,DataLayer.Entity.ProvisioningEntry.TokenOption,System.Boolean,System.String@)

* *Summary:* Adds a user's request to be provisioned a token. Gets added to the pending provisioning request list.
* *Param:* **u:** User requesting the token.
* *Param:* **tokenType:** Token type being requested
* *Param:* **existingUser:** True if user already exists
* *Param:* **message:** Message to the user in case of a failure
* *Returns:*

```
True if success
```

BSIDCA

AddCustomTypeProvisioningRequest(DataLayer.Entity.User,System.Int32,System.Boolean,System.String@)

* *Summary:* Adds a user's request to be provisioned a token. Gets added to the pending provisioning request list.
* *Param:* **u:** User requesting the token.
* *Param:* **subType:** Token sub type being requested
* *Param:* **existingUser:** True if user already exists
* *Param:* **message:** Message to the user in case of a failure
* *Returns:*

```
True if success
```

BSIDCA

UpdateProvisioningRequest(System.Int32,DataLayer.Policy.TokenAuthority)

* *Summary:* Approves a provisioning request.
* *Param:* **requestID:** Request ID to update
* *Param:* **authority:** What level to approve the request at.
* *Returns:*

```
True if updated
```

BSIDCA

DenyProvisioningRequest(System.Int32)

* *Summary:* Denies a pending provisioning request.
* *Param:* **requestID:** Request to deny
* *Returns:*

```
True if denied
```

BSIDCA

GetPendingProvisioningRequests()

* *Summary:* Gets a table of all pending provisioning requests for the organization you are logged in to.
* *Returns:*

```
A table with the columns:
      request
      userName
      FirstName
      LastName
      email
      token
      processState
      requestedOn
      newUser
      expiryDate
```

BSIDCA

GetDelegationCodes(System.Int32,System.Int32,System.String)

* *Summary:* Gets a table of delegation code information in an account.
* *Param:* **firstRecord:** First record
* *Param:* **pageSize:** Number of records to return
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      code
      orgToManage
      codeState
      codeDate
      establishedDate
      accountGroup
      canActivate
```

BSIDCA

`CreateDelegationCode(System.String,System.String)`

* *Summary:* Create a new delegation code for external management of another account.
* *Param:* **`accountGroup`:** Account group to add the code in.
* *Param:* **`organization`:** Account
* *Returns:*

```
The delegation code
```

BSIDCA

`ActivateDelegationCode(System.String,System.String)`

* *Summary:* Activate a delegation code that has been used by another account and is waiting for approval.
* *Param:* **`code`:** The code to activate
* *Param:* **`organization`:** Account
* *Returns:*

```
True if activated
```

BSIDCA

`RemoveDelegationCode(System.String,System.String)`

* *Summary:* Remove a delegation code and all associated rights from the account.
* *Param:* **`code`:** Code to remove
* *Param:* **`organization`:** Account
* *Returns:*

```
True if removed
```

BSIDCA

`GetAccounts(System.String,System.String,System.Int32,System.Int32)`

* *Summary:* Get a table of all accounts under the account that the current operator belongs to. This list will be automatically filtered for the operator scope.
* *Param:* **`organizationFilter`:** Account Filter
* *Param:* **`customFieldFilter`:** Custom field filter
* *Param:* **`firstRecord`:** First record to return
* *Param:* **`pageSize`:** Number of records to return
* *Returns:*

```
A table with the columns:
      Orgname
      custom1
      class
      activated
      expires
      billing
      capacity
      unused
      active
```

BSIDCA

`GetAccount(System.String)`

* *Summary:* Get the account details of a single child account
* *Param:* **`organization`:** Account
* *Returns:*

BSIDCA

UpdateOrganization(DataLayer.Entity.Organization)

* *Summary:* Update the account details of a child account
* *Param:* **organization:** Account to update
* *Returns:*

BSIDCA

AddOrganization(DataLayer.Entity.Organization,System.String)

* *Summary:* Add a new organization under the current account. Current account must be a service provider. Note: To set the capacity for a specific organization, call AddOrganization first, and then call AllocateCapacity. The capacity value of organization objects is not used.
* *Param:* **organization:** New organization to create.
* *Param:* **accountGroup:** Account group to create the organization in.
* *Returns:*

```
Success
NoOrganizationName
ParentNotAServiceProvider
ParentNotEnoughCapacity
RootOrganizationAlreadyExists
NoParent
NonExistantParent
FailedToAddToDatabase
GeneralError
```

BSIDCA

RemoveOrganization(System.String)

* *Summary:* Remove an existing organization from the system. All tokens and capacity must be de-allocated before being removed.
* *Param:* **organization:** Name of the organization to remove
* *Returns:*

Success	
Fail_HasChildOrganizations	Remove sub accounts first.
Fail_HasTokens	De-allocate tokens first.
Fail_IsSystem	Cannot remove required system accounts.
Fail_HasCapacity	De-allocate capacity first.
Fail_CouldntRemove	Couldn’t remove the organization from the database.
Fail_Error	An error occurred while attempting to remove the organization.

BSIDCA

AllocateCapacity(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.String)

* *Summary:* Allocate capacity to a child account.
* *Param:* **quantity:** Capacity to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateICE(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.String)

* *Summary:* Allocate ICE capacity to a child account. ICE tokens will automatically be allocated to the child account if required and available to be allocated.
* *Param:* **quantity:** ICE Capacity to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateHardware(System.Collections.Generic.List{System.String},DataLayer.Entity.Token.TokenType,DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* Allocate hardware tokens to a child account. This includes KT, RB and OATH tokens.
* *Param:* **serials:** List of serial numbers to allocate
* *Param:* **tokenType:** Token type of the allocation
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **withCapacity:** True to automatically add license capacity with the tokens
* *Param:* **isSale:** True if this is a sale, false for rental.
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateSoftware(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* Allocate software tokens to a child account. This is for MP tokens only. (SMS tokens are MP tokens issued to an SMS end point.)
* *Param:* **quantity:** Number of tokens to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **withCapacity:** True to automatically add license capacity with the tokens
* *Param:* **isSale:** True if this is a sale, false for rental.
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateRSA(System.Collections.Generic.List{System.String},DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* Allocate RSA SecurID tokens to a child account
* *Param:* **serials:** List of serial numbers to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **withCapacity:** True to automatically add license capacity with the tokens
* *Param:* **isSale:** True if this is a sale, false for rental.
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateGrIDsure(System.Collections.Generic.List{System.String},DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* Allocate GrIDsure tokens to a child account.
* *Param:* **serials:** List of serial numbers to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **withCapacity:** True to automatically add license capacity with the tokens
* *Param:* **isSale:** True if this is a sale, false for rental.
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateSMSCredits(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.String)

* *Summary:* Allocate SMS credits to a child account. SMS credits are not only required for SMS tokens, but for all SMS notification generated by the system. If your account has it's own SMS configuration, allocation of SMS credits does not require SMS credits.
* *Param:* **quantity:** Number of credits to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

AllocateMobilePASS(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* Allocate software tokens to a child account. This is for MobilePASS tokens only.
* *Param:* **quantity:** Number of tokens to allocate
* *Param:* **billingStyle:** Billing style of the allocation: Allocation, Activation, Authentication or Transfer
* *Param:* **withCapacity:** True to automatically add license capacity with the tokens
* *Param:* **isSale:** True if this is a sale, false for rental.
* *Param:* **toOrganization:** Account to allocate to
* *Returns:*

```
True if allocated successfully
```

BSIDCA

DeallocateCapacity(System.Int32,System.String,System.String)

* *Summary:* De-allocate capacity from a child account.
* *Param:* **quantity:** Capacity to de-allocate
* *Param:* **fromOrganization:** Account to de-allocate from
* *Param:* **intoContainer:** Container to move any tokens into
* *Returns:*

```
True if de-allocated
```

BSIDCA

DeallocateICE(System.Int32,System.String,System.String)

* *Summary:* De-allocate ICE capacity from a child account.
* *Param:* **quantity:** Capacity to de-allocate
* *Param:* **fromOrganization:** Account to de-allocate from
* *Param:* **intoContainer:** Container to move any tokens into
* *Returns:*

```
True if de-allocated
```

BSIDCA

DeallocateHardware(System.Collections.Generic.List{System.String},DataLayer.Entity.Token.TokenType,ControlLibrary.AccountsControl.TokensSelection,System.Boolean,System.String,System.String)

* *Summary:* De-allocate hardware tokens from a child account. This includes KT, RB and OATH tokens.
* *Param:* **serials:** List of serials to de-allocate from the child account
* *Param:* **tokenType:** Token type to de-allocate
* *Param:* **deallocationState:** State to return tokens to: ReturntoInventory_Initialized, ReturntoInventory_NotInitialized, Faulty, Lost
* *Param:* **isSale:** True if tokens are being sold back up the chain.
* *Param:* **intoContainer:** Container to place the tokens into
* *Param:* **fromOrganization:** Account to move tokens from
* *Returns:*

```
True if tokens are de-allocated
```

BSIDCA

DeallocateSoftware(System.Int32,ControlLibrary.AccountsControl.TokensSelection,System.Boolean,System.Boolean,System.String,System.String)

* *Summary:* De-allocate software tokens from a child account. This is for MP tokens only. (SMS tokens are MP tokens issued to an SMS end point.)
* *Param:* **quantity:** Number of tokens to de-allocate
* *Param:* **deallocationState:** State to return tokens to: ReturntoInventory_Initialized, ReturntoInventory_NotInitialized, Faulty, Lost
* *Param:* **withCapacity:** True to automatically include capacity
* *Param:* **isSale:** True if tokens are being sold back up the chain.
* *Param:* **intoContainer:** Container to place the tokens into
* *Param:* **fromOrganization:** Account to move tokens from
* *Returns:*

```
True if tokens are de-allocated
```

BSIDCA

DeallocateRSA(System.Collections.Generic.List{System.String},System.Boolean,System.String,System.String)

* *Summary:* De-allocate RSA SecurID tokens from a child account.
* *Param:* **serials:** List of serials to de-allocate from the child account
* *Param:* **isSale:** True if tokens are being sold back up the chain.
* *Param:* **intoContainer:** Container to place the tokens into
* *Param:* **fromOrganization:** Account to move tokens from
* *Returns:*

```
True if tokens are de-allocated
```

BSIDCA

DeallocateGrIDsure(System.Collections.Generic.List{System.String},System.Boolean,System.String,System.String)

* *Summary:* De-allocate GrIDsure tokens from a child account.
* *Param:* **serials:** List of serials to de-allocate from the child account
* *Param:* **isSale:** True if tokens are being sold back up the chain.
* *Param:* **intoContainer:** Container to place the tokens into
* *Param:* **fromOrganization:** Account to move tokens from
* *Returns:*

```
True if tokens are de-allocated
```

BSIDCA

DeallocateSMSCredits(System.Int32,System.String)

* *Summary:* De-allocate SMS credits from a child account.
* *Param:* **quantity:** Number of credits to de-allocate
* *Param:* **fromOrganization:** Account to take the credits from
* *Returns:*

```
True if the credits have be de-allocated
```

BSIDCA

DeallocateMobilePASS(System.Int32,DataLayer.Entity.Transaction.BillingStyle,System.Boolean,System.Boolean,System.String)

* *Summary:* De-allocate software tokens from a child account. This is for MobilePASS tokens only.
* *Param:* **quantity:** Number of tokens to de-allocate
* *Param:* **isSale:** True if tokens are being sold back up the chain.
* *Param:* **intoContainer:** Container to place the tokens into
* *Param:* **fromOrganization:** Account to move tokens from
* *Returns:*

```
True if tokens are de-allocated
```

BSIDCA

AddOperator(DataLayer.Entity.User,DataLayer.Entity.ProvisioningEntry.TokenOption,System.Nullable{System.Int32},System.String)

* *Summary:* Adds an operator to a child account. If the account is a service provider, the created operator will have administrator permissions on the account as well.
* *Param:* **user:** User to add as an operator. Must be a new user.
* *Param:* **tokenType:** The type of token/password to provision to the user.
* *Param:* **modelNumber:** Model Number for custom tokens. GrIDsure = 10001, SecurID = 10002, RADIUS = 10004
* *Param:* **organization:** Account
* *Returns:*

```
Success
UnknownError
UserAlreadyExists
FailedToCreateUser
FailedToProvision
EmailAddressAlreadyInUse
FailedToSendEnrollmentEmail
NotEnoughSMSTokens
FailedToSendSMSMessage
```

BSIDCA

AddAuthNode(BlackShield.BSIDCA.AuthNode,System.String)

* *Summary:* Adds an auth node to the account. IP addresses associated with the auth node must be system wide unique. Nodes can be used to add realms and provide common authentication points for child accounts.
* *Param:* **node:** AuthNode to add
* *Param:* **organization:** Account
* *Returns:*

```
True if added
```

BSIDCA

RemoveAuthNode(System.String,System.String)

* *Summary:* Remove an auth node from the account.
* *Param:* **nodeName:** Name of the AuthNode to remove
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

UpdateAuthNode(System.String,BlackShield.BSIDCA.AuthNode,System.String)

* *Summary:* Update an auth node in the account
* *Param:* **oldName:** Name of the auth node to update. If changing the node name, this is it's old name
* *Param:* **node:** Updated AuthNode
* *Param:* **organization:** Account
* *Returns:*

```
True if updated
```

BSIDCA

GetAuthNodes(System.String)

* *Summary:* Get a table of the auth nodes in the account.
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      index
      make
      host
      ipaddress
      status
      freeradius
```

BSIDCA

GetAuthNode(System.String,System.String)

* *Summary:* Get an auth node in the account.
* *Param:* **nodeName:** Name of the node to get
* *Param:* **organization:** Account
* *Returns:*

```
AuthNode that was requested if it exists
```

BSIDCA

AddShibbolethNode(BlackShield.BSIDCA.ShibbolethNode,System.String,System.String, System.String)

* *Summary:* Adds a Shibboleth service provider.
* *Param:* **node:** ShibbolethNode to add
* *Param:* **entityIdString:** Metadata that specifies the unique URI identifier of the SAML entity whose metadata is described by the element's contents.
* *Param:* **locationString:** Metadata required URI attribute that specifies the location of the endpoint. The allowable syntax of this URI depends on the protocol binding.
* *Param:* **organization:** Account
* *Returns:*

```
True if added
```

BSIDCA

RemoveShibbolethNode(System.String,System.String)

* *Summary:* Removes a Shibboleth service provider.
* *Param:* **node:** ShibbolethNode to remove
* *Param:* **relyingId:** The URL that the Shibboleth agent uses to communicate with SAS regarding any changes related to the addition or removal of a service provider.
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

AddSamlExtension(DataLayer.Entity.SAMLUserExtension,System.String,System.String)

* *Summary:* Adds a SAML user extension to a user.
* *Param:* **extension:** SAML user extension to add
* *Param:* **userName:** User Name
* *Param:* **organization:** Account
* *Returns:*

```
True if added
```

BSIDCA

GetBmcFile(System.String)

* *Summary:* Get the BMC file for the organization. This file contains the keys that are used to encrypt the data transferred between the Synchronization Agent and SAS.
* *Param:* **organization:** Organization name
* *Returns:*

```
BMC file as binary array if LDAP Sync Server settings are configured, otherwise return
null.
```

BSIDCA

GetPasswordPolicy(System.String)

* *Summary:* Return the static password policy
* *Param:* **organization:** Account
* *Returns:*

```
PasswordPolicy for the account
```

BSIDCA

GetAvailableReports(DataLayer.Entity.ReportLevel,System.Nullable{DataLayer.Entity.ReportClass},System.Int32,System.Int32,System.String)

* *Summary:* Get All report available at a given level to customize. Optionally filter by report class.
* *Param:* **level:** Report level
* *Param:* **type:**
* *Param:* **startRecord:**
* *Param:* **PageSize:**
* *Param:* **organization:**
* *Returns:*

```
A table with the columns:
      reportName
      reportType
      description
```

BSIDCA

GetAvailableReport(DataLayer.Entity.ReportLevel,System.String,System.String)

* *Summary:* Get an available report to edit and save. Note: Report columns and report filter definitions are read only except for the report columns .IsIncluded property.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report to get
* *Param:* **organization:** Account
* *Returns:*

```
The report if it exists
```

BSIDCA

GetSavedReports(DataLayer.Entity.ReportLevel,System.Int32,System.Int32,System.String)

* *Summary:* Get a table of all previously saved reports.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **startRecord:** First record to return
* *Param:* **pageSize:** Number of records to return
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      reportName
      reportType
      description
```

BSIDCA

GetSavedReport(DataLayer.Entity.ReportLevel,System.String,System.String)

* *Summary:* Get a previously saved report.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report
* *Param:* **organization:** Account
* *Returns:*

```
The saved report, if it exists
```

BSIDCA

SaveReport(DataLayer.Entity.Report,System.String)

* *Summary:* Save a customized report. This allows the report to scheduled. All reports must have at least one operator/account manager/external operator. Note: Report columns and report filter definitions are read only except for the report columns .IsIncluded property.
* *Param:* **report:** Report to save
* *Param:* **organization:** Account
* *Returns:*

```
True if saved
```

BSIDCA

UpdateReport(DataLayer.Entity.Report,System.String)

* *Summary:* Update a saved report with new information. Note: Report columns and report filter definitions are read only except for the report columns .IsIncluded property.
* *Param:* **report:** Report to update
* *Param:* **organization:** Account
* *Returns:*

```
True if the report is updated
```

BSIDCA

RemoveReport(DataLayer.Entity.ReportLevel,System.String,System.String)

* *Summary:* Remove a report that had been previously saved.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report to remove
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetScheduledReports(DataLayer.Entity.ReportLevel,System.Int32,System.Int32,System.String)

* *Summary:* Get a table of all report scheduled to run.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **startRecord:** First record to return
* *Param:* **pageSize:** Number of records to return
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      reportName
      frequency
      runtime
      expires
```

BSIDCA

GetScheduledReport(DataLayer.Entity.ReportLevel,System.String,System.String)

* *Summary:* Get a report that was previously scheduled
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the scheduled report
* *Param:* **organization:** Account
* *Returns:*

```
The scheduled report, if it exists
```

BSIDCA

ScheduleReport(DataLayer.Entity.ScheduledReport,System.String)

* *Summary:* Schedule a report to run at a given time. Note: Report columns and report filter definitions are read only except for the report columns .IsIncluded property
* *Param:* **report:** The report to schedule
* *Param:* **organization:** Account
* *Returns:*

```
True if scheduled
```

BSIDCA

ScheduleReportToRunNow(DataLayer.Entity.ScheduledReport,System.String)

* *Summary:* Schedule a report to run as soon as possible
* *Param:* **report:** Report to schedule
* *Param:* **organization:** Account
* *Returns:*

```
True if scheduled
```

BSIDCA

UpdateScheduledReport(DataLayer.Entity.ScheduledReport,System.String)

* *Summary:* Update a previously scheduled report. Note: Report columns and report filter definitions are read only except for the report columns .IsIncluded property.
* *Param:* **report:** Scheduled Report to update
* *Param:* **organization:** Account
* *Returns:*

```
True if updated
```

BSIDCA

RemoveScheduledReport(DataLayer.Entity.ReportLevel,System.String,System.String)

* *Summary:* Remove a scheduled report
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report to remove
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetFinishedReports(DataLayer.Entity.ReportLevel,System.Int32,System.Int32,System.String)

* *Summary:* Get a table of all reports that have been run.
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **startRecord:** First record to return
* *Param:* **pageSize:** Number of records to return
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      reportname
      scheduledTime
      status
      timestamp
      downloadable      (True if the operator credentials in use have permission to
      download)
      removable         (True if the operator credentials in use have permission to
      remove)
```

BSIDCA

RemoveFinishedReport(DataLayer.Entity.ReportLevel,System.String,System.DateTimeOffset,System.String)

* *Summary:* Remove a finished report from the system. NOTE: The results of this report will no longer be available after removal. The scheduledTime parameter must be a string in "yyyy-MM-ddTHH:mm:ss.fffffffzzzz" format (for example, 2015-07-16T12:18:07.1556784-04:00).
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report to remove
* *Param:* **scheduledTime:** The time the report ran
* *Param:* **organization:** Account
* *Returns:*

```
True if removed
```

BSIDCA

GetReportOutput(DataLayer.Entity.ReportLevel,System.String,System.DateTimeOffset,System.String,ControlLibrary.ReportControl.OutputFormat)

* *Summary:* Get the results of a report. Returned information is UTF-8 encoded text. Output can be in CSV, TSV, or HTML Table format. The scheduledTime parameter must be a string in "yyyy-MM-ddTHH:mm:ss.fffffffzzzz" format (for example, 2015-07-16T12:18:07.1556784-04:00).
* *Param:* **level:** Report level that the report exists at. Subscriber, ServiceProvider
* *Param:* **reportName:** Name of the report to get
* *Param:* **scheduledTime:** The time the report ran
* *Param:* **organization:** Account
* *Param:* **format:** Desired output format: HTML, CSV or TAB (Tab delimited)
* *Returns:*

```
UTF8 encoded document in the format requested
```

BSIDCA

GetAccountManagers(System.String)

* *Summary:* Get a list of account managers in the given account.
* *Param:* **organization:** Account
* *Returns:*

```
List of account manager names
```

BSIDCA

GetOperators(System.String)

* *Summary:* Get a list of operators in the given account.
* *Param:* **organization:** Account
* *Returns:*

```
List of operator names
```

BSIDCA

GetExternalOperators(System.String)

* *Summary:* Get a list of eternal operators in the given account.
* *Param:* **organization:** Account
* *Returns:*

```
List of external operator. (Account names)
```

BSIDCA

GetGrIDsurechallenge(System.String,System.String)

* *Summary:* Get a challenge string for a GrIDsure token. This string must be processed into an image and presented to the user.
* *Param:* **userName:** User who is requesting the challenge
* *Param:* **organization:** Account
* *Returns:*

```
GrIDsure challenge string
```

BSIDCA

AuthenticateGrIDsureToken(System.String,System.String,System.String)

* *Summary:* Authenticate a GrIDsure token
* *Param:* **userName:** User with the token
* *Param:* **response:** Reponse
* *Param:* **organization:** Account
* *Returns:*

```
True if successful
```

BSIDCA

ResetGrIDsureTokenPIP(System.String,System.Byte[],System.String,System.String)

* *Summary:* Reset the PIP for a GrIDsure token using an initialization key from getGrIDsureTokenInitKey and the correct response.
* *Param:* **userName:** User with the token
* *Param:* **initKey:** Initialization key from getGrIDsureTokenInitKey
* *Param:* **response:** Reponse to the initialization key
* *Param:* **organization:** Account
* *Returns:*

```
True if the PIP was reset
```

BSIDCA

getGrIDsureTokenInitKey(System.String,System.String)

* *Summary:* Get the initialization Key for a GrIDsure token.
* *Param:* **userName:** User with a GrIDsure token
* *Param:* **organization:** Account
* *Returns:*

```
Initialization key
```

BSIDCA

getGrIDsureToken(System.String,System.String)

* *Summary:* Get the GrIDsure token for a given user.
* *Param:* **userName:** User name
* *Param:* **organization:** Account
* *Returns:*

```
The user's GrIDsure token (if they have one)
```

BSIDCA

GetTokenRequestTable(System.String)

* *Summary:* Gets a table of token types that are requestable.
* *Param:* **organization:** Account
* *Returns:*

```
A table with the columns:
      text
      tokentype
      enable
      subtarget
```

BSIDCA

RequestToken(System.String,DataLayer.Entity.ProvisioningEntry.TokenOption,System.Nullable{System.Int32},System.Nullable{DataLayer.Entity.Token.TokenTargets},System.String)

* *Summary:* Request a token.
* *Param:* **userName:** User that is requesting the token
* *Param:* **tokenoption:** Token type they are requesting
* *Param:* **modelnumber:** Model Number for custom tokens. GrIDsure = 10001, SecurID = 10002, RADIUS = 10004
* *Param:* **MPTarget:** MP target if requesting an MP and want to choose now instead of during enrollment. Options: HardDrive, Blackberry, iPhone, Android, WindowsPhone7, MacOSX
* *Param:* **organization:** Account
* *Returns:*

```
Request ID for the token request, or 0 for a failure
```

BSIDCA

ConfirmRequestToken(System.Int32,System.String,System.String)

* *Summary:* Mark a token request as confirmed.
* *Param:* **requestID:** Request ID to confirm
* *Param:* **userName:**
* *Param:* **organization:** Account
* *Returns:*

```
True if confirmed
```

BSIDCA

RevokeAllSIMTokens(System.String,System.String)

* *Summary:* Revokes all of a SIM's tokens. NOTE: Only revokes tokens that belong to the given organization or any of their decendants.
* *Param:* **userName:** MSISDN to revoke tokens from
* *Param:* **organization:** Organization the user is in
* *Returns:*

BSIDCA

ReprovisionAllSIMTokens(System.String,System.String)

* *Summary:* Reprovision all of a SIM's tokens. NOTE: Only reprovisions tokens that belong to the given organization or any of their decendants.
* *Param:* **userName:** MSISDN to reprovision tokens to.
* *Param:* **organization:** Organization the user is in
* *Returns:*

BSIDCA

GetMobilePASSProvisioningActivationCode(System.String,System.Int32,System.String)

* *Summary:* Gets the base64 activation code for a user's MobilePASS provisioning task.
* *Param:* **userName:** User with the task.
* *Param:* **taskID:** Task ID for the specific provisioning task.
* *Param:* **organization:** Organization the user is in.
* *Returns:*

BSIDCA
GetEnrollmentURL(System String,System.Int32,System.String)
* *Summary:* Gets the self-enrollment URL for a given user's provisioning task.
* *Param:* **userName:** User with the task.
* *Param:* **taskID:** Task ID for the specific provisioning task.
* *Param:* **organization:** Organization the user is in.
* *Returns:* Self enrollment URL for the specific provisioning task

Support Contacts

If you encounter a problem while installing, registering or operating this product, please make sure that you have read the documentation. If you cannot resolve the issue, contact your supplier or Gemalto Customer Support. Gemalto Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Gemalto and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Contact Method	Contact Information	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017, USA	
Phone	US	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://serviceportal.safenet-inc.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.	