Connected Backup

Software Version 8.11

Web Services Programming Reference

Document Release Date: October 2018
Software Release Date: October 2018

Legal notices

Copyright notice

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

You can check for more recent versions of a document through the MySupport portal. Many areas of the portal, including the one for documentation, require you to sign in with a Software Passport. If you need a Passport, you can create one when prompted to sign in.

Additionally, if you subscribe to the appropriate product support service, you will receive new or updated editions of documentation. Contact your Micro Focus sales representative for details.

Support

Visit the MySupport portal to access contact information and details about the products, services, and support that Micro Focus offers.

This portal also provides customer self-solve capabilities. It gives you a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the MySupport portal to:

- Search for knowledge documents of interest
- Access product documentation
- View software vulnerability alerts
- Enter into discussions with other software customers
- Download software patches
- Manage software licenses, downloads, and support contracts
- Submit and track service requests
- Contact customer support
- View information about all services that Support offers

Many areas of the portal require you to sign in with a Software Passport. If you need a Passport, you can create one when prompted to sign in. To learn about the different access levels the portal uses, see the Access Levels descriptions.

Contents

Chapter 1: Get started

This chapter describes the Web Services API for Micro Focus Connected Backup.

About the Web Services API

The Web Services API is an XML Web service with a SOAP API that allows you to make calls from your application to the Support Center to manage accounts, communities, and reports. You can use this collection of APIs to:

- Create and reserve accounts
- Get and set account information including name, address, telephone number, and e-mail
- Cancel accounts or place them on hold
- Validate accounts and change passwords
- Move accounts into new communities and change accounts' Agent setup
- Run reports

In this guide

This reference guide contains:

System requirements and permissions

The following table lists the requirements for using the Web Services API:

Requirement	Description
License	You must obtain and install a valid license to use the Web Services API.

Requirement	Description
agreement	
Software requirement	The Support Center server software version must be version 7.5 or higher.
Cookies	To enable multiple API calls per session, the client making an API call must support HTTP cookies and return any cookie sent to it by the server on any calls subsequent to the SessionLoginTechnician().
SSL	The Web Services API requires SSL certificates.

Use scripting permission

Individuals who are responsible for using these APIs must have the **Use Scripting** technician permission enabled. Permissions may be granted or revoked using Support Center.

Refer to Support Center help for information about granting technician permissions.

In addition to the **Use Scripting** permission, some APIs require additional technician permissions. For example, the AccountMoveToCommunity API requires that the technician making the call has the **Modify Communities** permission enabled in both the original and destination community. Before making a call to any API that sets or changes data, check its description to determine if it requires specific permissions.

Limitations

The following are important limitations you should be aware of when you use these APIs.

Required country values for input

There is currently no validation of country names to standardize user input. You must use the ISO standard for English Country Names. Click here to view a list of ISO standard names.

Empty array handling

Certain APIs, such as CommunityFindAccounts return an empty array when the call is successful but no matching results are found. Because C# cannot handle an empty array returned as part of a SOAP response, make sure to check for null values (if (array == null) {} before performing any operation on the returned array.

Password restrictions

Account password must be at least 6 characters long, cannot have leading and trailing space and cannot contain all the same characters. Technician password must be at least 8 characters long, including at least one numeric character. Passwords are case-sensitive. Passwords should not contain caret symbols (< or >).

Truncation of strings greater than Max Size

For each API that has string parameters, an additional column called Max String Size shows the maximum number of wide (Unicode) characters permitted for the string. Strings that exceed their set Max String Size are truncated.

Develop with the Web Services API

This section contains important information you should know before you begin working on your project.

Location

The Web Services API is exposed at the following URL:

`https://supportcentermachine/AdminAPI`

Get a copy of the Web Services API WSDL file

A WSDL file that describes the Web Services API is created on the server when the Web Services API is installed. To obtain copy of the AdminAPI.wsdl file, open it from the Web server using a browser.

For example:

`https://www.connected.com/AdminAPI/AdminAPI.wsdl`

Select **Save As** from the browser's **File** menu and save the file as an XML file.

> **NOTE:**
> If you plan to write clients in C#, you can use the WSDL file installed with the Web Services API to generate several C# wrapper classes.
>
> For more information, see Create C# wrapper classes, on page 76.

Chapter 2: APIs

This chapter describes the interface categories available for the Web Services API.

Account APIs

You can use the following APIs for account operations:

AccountDisableiRoam

Denies MyRoam access to the specified account by changing the MyRoam state to False.

Parameters

Name	Description	Type
AccountNumber	The account number of the account that you want to deny iRoam or MyRoam access.	xsd:int

Return Values

If successful, none.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1025	The Data Center is not licensed to use this feature.

Remarks

If iRoam or MyRoam access is already disabled for the specified account, nothing is done and no error messages are returned.

Example

```
[C# Example]

int intAccount = 101000401;

//Turn off access to iRoam or MyRoam so the user cannot
//use the Web interface, iRoam or MyRoam

AdminService.AccountDisableiRoam(intAccount);
```

AccountEnableiRoam, on the next page

AccountEnableiRoam

Enables MyRoam access to the specified account by changing the MyRoam state to True.

Parameters

Name	Description	Type
AccountNumber	The account number of the account that you want to grant iRoam or MyRoam access.	xsd:int

Return Values

If successful, none.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1025	The Data Center is not licensed to use this feature.

Remarks

If iRoam or MyRoam access is already enabled for the specified account, nothing is done and no error messages are returned.

Example

```
[C# Example]

int intAccount = 101000401;


//Turn iRoam or MyRoam access on so the user can

//retrieve files using the Web interface, iRoam or MyRoam

AdminService.AccountEnableiRoam(intAccount);
```

AccountEnableiRoam, above

AccountGetInfo, on page 15

AccountGetEncryptionKey

Discloses the encryption key for the specified account.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number.	xsd:int	
Justification	The reason why the encryption key is being disclosed	xsd:string	255

Return Values

Name	Description	Type
EncryptionKey	The unencrypted key for the specified account.	xsd:string

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1023	Justification cannot be blank.
1054	Access denied. Logged-in Technician does not have permission ‘Disclose Encryption Keys’.
1055	Unable to disclose encryption key. It is not escrowed.
1056	The call is not allowed for specified account since its agent version does not support this feature.

Remarks

The logged in technician must have the **Disclose Encryption Keys** permission.

Example

```
[C# Example]

int intAccount = 101000401;

//Get the account's EncryptionKey
string strEKey = AdminService.AccountGetEncryptionKey(intAccount, "Account holder
forgot encryption key");
Console.WriteLine("{0} has an Encryption key of: {1}", intAccount, strEKey);
```

AccountSetPassword, on page 30

AccountGetInfo, on the next page

AccountGetExtendedInfo

Get selected profile information for a specified account (AccountNumber). Information you can get includes:

- Account cancellation date
- Account deletion date
- Message code (selected by technician canceling the account or putting it on hold)
- Billing method code
- Other profile information

Parameters

Name	Description	Type
AccountNumber	The number of the account you want to get extended information for	xsd:int
FieldName	Type of profile field containing the requested data. Can be one of the following: • PROFILEFIELD_SECTION_NAME • PROFILEFIELD_ATTRIBUTE_NAME	Connected:PROFILEFIELD
FieldValue	Section or attribute for which you want to get information	xsd:string

Return Values

Name	Description	Type
ExtendedAccountInfo	A data structure that contains profile information for the specified account.	Connected: AdminAPIExtendedAccountInfo, on page 97

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
AdminAPIExtendedAccountInfo EAI = m_AAPI.AccountGetExtendedInfo(101000101,
PROFILEFIELD.PROFILEFIELD_SECTION_NAME, "Network");

Console.Writeline("Cancel Date: " + EAI.dtCancelDate.ToString());
Console.Writeline("Delete Date: " + EAI.dtDeleteDate.ToString());
Console.Writeline("Billing Method: " + EAI.nBillingMethod.ToString());
Console.Writeline("Message Code: " + EAI.nMsgCode.ToString());

int idx=0;
foreach ( AdminAPIProfileInfo Temp in EAI.ProfileInfo)
{
                Console.Writeline("Profile Info " + idx + ":");
        Console.Writeline("\tAttribute: " + Temp.strAttribute);
        Console.Writeline("\tSection: " + Temp.strSection);
        Console.Writeline("\tValue: " + Temp.strValue);
        idx++;
}
```

AccountGetInfo

Gets information for the specified account number. Information is returned as both individual values and structures containing values.

Parameters

Name	Description	Type
AccountNumber	The account for which you want to retrieve Agent, account size, or user information.	xsd:int

Return Values

Name	Description	Type
AdminAPIAccountInfo	This structure that contains information about the account including its start date, its Agent install path, Agent version, user information, account size and custom fields.	Connected: AdminAPIAccountInfo, on page 91

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

- If the currently logged–in technician has permission to view credit card data, the credit card type and number are also retrieved.
- If the logged–in technician does not have the **Provide Billing** permission, the card type is returned as 'unknown' and the card number is blank.
- For accounts created prior to version 7.1, the Agent Setup ID is -2.
- For accounts created using Support Center (prior to version 7.5), the Agent Setup ID is not set and its returned value is -1. This indicates the Agent Setup ID will be determined when the user registers the account.
- If Custom1 attribute name starts with `Dep` (or equivalent language translation meaning "Department"), the Custom1 field is not populated. In this case, the first element of the array has all the empty values. The User Department field is populated with the value of this attribute. If Custom2 is present, the second element of the array holds the values of it.

Example

```
[C# Example]

AdminAPIAccountInfo cAcntInfo = AdminService.AccountGetInfo(intAccount);

Console.WriteLine(cAcntInfo.dtStartDate.ToString());
Console.WriteLine(cAcntInfo.strAgentInstallPath);
Console.WriteLine(cAcntInfo.strAgentVersion);
Console.WriteLine(cAcntInfo.strComputerName);
```

AccountSetUserInfo, on page 26

AccountGetEncryptionKey, on page 13

AccountGetInfoEx

Gets information for the specified account number. Information is returned as both individual values and structures containing values. This call is similar to AccountGetInfo, on page 15, but this call includes both date and time information.

Parameters

Name	Description	Type
AccountNumber	The account for which you want to retrieve Agent, account size, or user information.	xsd:int

Return Values

Name	Description	Type
AccountInfoEx	This structure that contains information about the account including its start date, its Agent install path, Agent version, user information, account size and custom fields.	Connected: AdminAPIAccountInfoEx, on page 92

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.

Code	Reason
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

- If the currently logged–in technician has permission to view credit card data, the credit card type and number are also retrieved.
- If the logged–in technician does not have the **Provide Billing** permission, the card type is returned as 'unknown' and the card number is blank.
- For accounts created prior to version 7.1, the Agent Setup ID is -2.
- For accounts created using Support Center (prior to version 7.5), the Agent Setup ID is not set and its returned value is -1. This indicates the Agent Setup ID will be determined when the user registers the account.
- If Custom1 attribute name starts with `Dep` (or equivalent language translation meaning "Department"), the Custom1 field is not populated. In this case, the first element of the array has all the empty values. The User Department field is populated with the value of this attribute. If Custom2 is present, the second element of the array holds the values of it.

Example

```
[C# Example]

AdminAPIAccountInfoEx  cAcntInfo = AdminService.AccountGetInfoEx (intAccount);

Console.WriteLine(cAcntInfo.dtStartDate.ToString());
Console.WriteLine(cAcntInfo.strAgentInstallPath);
Console.WriteLine(cAcntInfo.strAgentVersion);
Console.WriteLine(cAcntInfo.strComputerName);
```

AccountSetUserInfo, on page 26

AccountGetEncryptionKey, on page 13.

AccountGetBackupDates

Gets information about the backup dates associated with a specified account. Information is returned as an array of the AdminAPIAccountBackupDateInfo structure.

Parameters

Name	Description	Type
AccountNumber	The account for which you want to get the last backup date	xsd:int

Return Values

Name	Description	Type
AdminAPIAccountBackupDateInfo	This structure that contains information about Tbackup dates, including its status, whether it was compacted, and the size of the backup.	Connected: AdminAPIAccountBackupDateInfo, on page 94

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1080	No backup dates for accounts.

AccountGetLastBackupDate

Gets the last backup date for specified account number.

Parameters

Name	Description	Type
AccountNumber	The account for which you want to get the last backup date	xsd:int

Return Values

Name	Description	Type
LastBackupDate	Date of the last backup for the specified account.	xsd:dateTime

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
DateTime dtLastBackUp = AdminService.AccountGetLastBackupDate(101000101);

Console.Writeline("Last Backup was on: " + dtLastBackup.ToString());
```

AccountGetMediaCount

This API enables you to find out how many units of recordable storage media (CDs, DVDs or NAS drives) are required to fulfil a media order. You can obtain this information before or after an account holder requests a copy of their backed-up data.

Parameters

Name	Description	Type
AccountNumber	The account number.	xsd:int

Return Values

Name	Description	Type
AdminAPIMediaCount, on page 97	A structure that contains the media type and count fields.	Connected:AdminAPIMediaCountList

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.

Code	Reason
1005	Access denied. Logged-in Technician does not have permission 'Order External Media'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

The currently logged in technician must have the **Order Media** permission.

AccountOrderMedia, on the next page

AccountMoveToCommunity

Move the specified account to a new community.

Parameters

Name	Description	Type
AccountNumber	The number of the account to move.	xsd:int
CommunityID	The ID of the target community.	xsd:int

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1003	Access denied. Logged-in Technician does not have permission 'Modify Communities'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1016	The specified account cannot be found on the system.
1024	Unable to perform required action. The destination community does not have enough licenses available.

Code	Reason
1036	An account cannot be moved to a community with a different Enterprise Directory configuration.
1037	Unable to move account to the data center level. An account can be moved only to a community.
1079	Access denied. Logged-in Technician does not have permission 'Move Accounts'.

Remarks

Logged in technician must have the **Modify Communities** permission for both the original and destination communities.

When an account is successfully moved, it consumes a license within the new community and releases the license in its old community. The license it used in the old community then becomes available to a new account.

Example

```
[C# Example]

int intAccount = 101000401;

//Move the account to a different community

AdminService.AccontMoveToCommunity(intAccount, intRootCmtyID);
```

CommunityGetSubCommunityIDs, on page 55

AccountGetInfo, on page 15

AccountOrderMedia

Place an order for all of the specified account's data to be written to the specified type of external media and shipped to the account address using the specified shipping priority.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number.	xsd:int	N/A
MediaType	Enumerated value. One of the following: • `Media_CD` (compact disc)	xsd:int	

Name	Description	Type	Max String Size
	• `Media_DVD` (digital video disc) • `Media_NAS` (network attached storage device)		

ShippingLabel	The name and address as it should appear on a shipping label. If this is not specified, it will be populated with:
	`FirstName LastName Telephone\n`
	`CompanyName\n`
	`Address1\n`
	`Address2\n`
	`City State ZIP Country`
		xsd:string	255

ShippingPriority	Enumeration value. One of: `High`, `Medium`, `Low`. This value is mapped to the appropriate default shipping method. • **High**. Next Day (no P.O. boxes) • **Medium**. 2nd Day (no P.O. boxes) • **Low**. Ground	Connected:Shipping Priority	

Return Values

When successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1005	Access denied. Logged-in Technician does not have permission ‘Order External Media’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

Logged-in Technician must have the **Order Media** permission.

ShippingPriority is saved in the database as a string representation of the ShippingPriority enumeration values.

AccountGetMediaCount, on page 20

AccountOrderMediaEx

Place an order for all of the specified account's data from a specified backup date to be written to the specified type of external media and shipped to the account address using the specified shipping priority. Use this API for version 8.0 Agents only.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number.	xsd:int	N/A
MediaType	Enumerated value. One of the following: • `Media_CD` (compact disc) • `Media_DVD` (digital video disc) • `Media_NAS` (network attached storage device)	Connected:MediaType	

ShippingLabel	The name and address as it should appear on a shipping label. If this is not specified, it will be populated with:
	`FirstName LastName Telephone\n`
	`CompanyName\n`
	`Address1\n`
	`Address2\n`
	`City State ZIP Country`
		xsd:string	255

ShippingPriority	Enumeration value. One of: `High`, `Medium`, `Low`. This value is mapped to the appropriate default shipping method. • **High**. Next Day (no P.O. boxes) • **Medium**. 2nd Day (no P.O. boxes) • **Low**. Ground	Connected:Shipping Priority	
BackupDate	A backup date.	xsd:date	

Return Values

When successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1005	Access denied. Logged-in Technician does not have permission 'Order External Media'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1081	Invalid Backup Date.
1082	No backup dates for Pre-8.0 Account.

Remarks

- Logged-in Technician must have the **Order Media** permission.
- *ShippingPriority* is saved in the database as a string representation of the ShippingPriority enumeration values.

AccountGetMediaCount, on page 20

AccountSendMessage

Sends a text message (Message) to a specified account (AccountNumber).

Parameters

Name	Description	Type
AccountNumber	The number of the account to which you want to send a text message.	xsd:int
Message	Message to send to the account.	xsd:string

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

Message content truncated after 1000 characters.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
AdminService.AccountSendMessage(101000101, "Please log off now.");
```

AccountSetUserInfo

Updates information for the specified account. To change the value for a single field, call the AccountGetInfo API to pre-set the other values before calling AccountSetUserInfo.

Parameters

Name	Description	Type
AccountNumber	The number of the account that you want to change information for.	xsd:int
AdminAPIUserInfo, on page 98	A structure that contains user account information fields.	Connected:AdminAPIUserInfo

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1012	Credit Card expiration Date is not a valid date.
1013	Credit Card type is invalid. Valid credit card types are Visa, MasterCard and AMEX.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1018	Credit Card number is invalid.
1045	Specified LoginID could not be located in Enterprise Directory.
1051	Access denied. Logged-in Technician does not have permission 'Change Enterprise Directory User'.
1052	Due to connection problems with the Enterprise Directory, the LoginID cannot be changed.

Remarks

- If you use AccountSetUserInfo to set the LoginID for an enterprise directory account, all other field values are ignored. These values will be obtained from the enterprise directory record for the account.
- To change enterprise directory login IDs, the technician must have the **Change User ID** permission.
- Custom field values cannot be set using this API.

  Valid credit card expiration date formats are:

  `mm/yy`

  `mm-yy`

  `mmyy`

  `mm/yyyy`

  `mm-yyyy`

  `mmyyyy`

  The valid date values are:

  - Month = `1` through `12`
  - Year is from 1 through 99 or 2001 through 2099. (Leading zeros are ignored, for example 003/009 is accepted and converted to 03/2009 date.)
- If date is submitted in any other form, the error 1012 is returned.
- When AccountGetInfo is called, the credit card expiration date is returned in the format `mm/yyyy`.
- To clear existing credit card information, submit empty credit card information. This clears the

number and date and the credit card type is set to CARD_TYPE::CARD_UNKNOWN.

- Either the complete credit card number or just the last four digits of the credit card number can be stored as a valid credit card number.
- When submitting a new credit card number, different information is required depending on whether the full credit card number or just the last four digits of the credit card number are stored:
  - If only the last four digits of the credit card number are stored, a valid expiration date and valid credit card type must be submitted.
  - If the full credit card number is stored, only a valid expiration date is required. The **Type** field is ignored since the type will be derived from the card number.
- Rules for changing credit card information are as follows:
  - If the submitted credit card number is the same or empty, the type is the same or unknown, but the expiration date is different, only the expiration date is changed.
  - If the submitted credit card number is the same or empty, the type is different and the date is different, the date is changed. The type is changed only if just the last four (4) digits of the credit card number are stored. Otherwise, type is ignored.
  - If the submitted credit card number and date are the same or empty, but the type is different, the type is changed only if just the last four (4) digits of the credit card number are stored. Otherwise, type is ignored.
  - If the submitted credit card number is different and the expiration date is valid, the existing credit card number and date is replaced. The type must be provided only if the existing type is invalid. If the new credit card number consists of four (4) digits and the stored credit card number is a full number, the full number is replaced with the four digit one. A different credit card number means the new number is not empty and if it is a full number, it doesn't match the existing one. If the new number is 4 digits and the stored number is a full number, just the last 4 digits of the existing number are compared and has to be different.

AccountGetInfo, on page 15

AccountSetStatus, on page 31

AccountSetPassword, on page 30

AccountSetAgentSetupID, below

AccountSetAgentSetupID

Changes the Agent Setup for the specified account. The Agent Setup is the Agent installation file. Each Agent installation executable can contain different features and permissions. The new Agent Setup must reside in the community or parent community of the specified account.

Parameters

Name	Description	Type
AccountNumber	The number of the user account that you want to assign a new Agent Setup to.	xsd:int
AgentSetupID	The ID number of the Agent Setup that you want to assign to the specified account. This determines which features and permissions are available to the account.	xsd:int

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1026	Community this account belongs to does not contain the Agent Setup specified.
1044	Access denied. Logged-in Technician does not have permission 'Change the Agent Setup of Accounts'.
1047	The Agent Setup of the account may not be changed because the account is deleted. Setups can be changed only for accounts that are active or on hold.

Remarks

- You can use this API to change Agent Setup for the reserved accounts, since it can be set when reserving the accounts in the CommunityReserveTicket API.
- Technician must have the **Change the Agent Configuration of Accounts** permission to change an account's Agent Setup.
- If you have the technician **Use Scripting** permission, you can get the community or Agent Setup ID number by hovering over the name of the current or parent community or agent setup in the Support Center interface.
- Use the `CommunityGetSubCommunityIDs` API to obtain a list of subcommunities in a specific

community. To determine the name of a community using its ID, use the `CommunityGetName` API.

- Use AccountGetInfo API to obtain an account's current Agent Setup ID. `AgentSetupID` is returned in `AdminAPIBaseAccount`.
- Optionally, you may use the Account C# helper class to get and set the `AgentSetupID`.

AccountGetInfo, on page 15

AccountSetPassword

Sets the password for the specified account.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number for the account that you want to change.	xsd:int	N/A
Password	The new password.	xsd:string	Refer to Password Restrictions for specific password limitations
Justification	The reason or justification for the password change.	xsd:string	255

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1022	The password provided does not conform to requirements. Account passwords must be at least 6 characters long, cannot have leading and trailing space and cannot contain all the same characters.
1023	Justification cannot be blank.
1053	Access denied. Logged-in Technician does not have permission ‘Reset Account Passwords'.

Remarks

The logged-in technician must have the **Reset Account Passwords** permission to use this API.

Example

```
[C# Example]

int intAccount = 101000401;


//Change the account password

AdminService.AccountSetPassword(intAccount, "NewPass1", "Account holder forgot
password, asked for new password");
```

AccountSetStatus, below

AccountVerifyUserCredentials, on page 34

AccountSetStatus

Sets the status of the specified account.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number for the account that you want to change.	xsd:int	
Status	New Status. The values defined in the Account Status table in CommunityFindAccounts API. Only one of three values from this table can be specified here: `Active`, `Cancelled`, `Onhold`	Connected:Account Status	
Justification	The reason or justification for the password change.	xsd:string	255
StatusCode	Status message code. For possible values refer to any Support Center Account status change page. If technician has **Use Scripting** permission each status message	xsd:int	

Name	Description	Type	Max String Size
	is prefixed with the status code. If 0 is submitted, the value is ignored and no error is produced.		

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.
1023	Justification cannot be blank.
1024	Unable to perform required action. The destination community does not have enough licenses available.
1038	Access denied. Logged-in Technician does not have permission 'Change the Status of Accounts'.
1039	The status of the account may not be changed at this time, as it is locked by another process.
1040	The status of the account may not be changed, as it is 'reserved'.
1041	The status of the account may not be changed, as it is 'deleted'.
1042	Cannot use the status specified. Status has to be one of the three values: `Active`, `Cancelled` or `OnHold`.
1060	Invalid status message code.
1061	The status of the account may not be change, as old status is invalid.

Remarks

- The Logged-in technician must have the permission **Change the Status of Accounts**.
- If the status of an account that is `Active`/`OnHold`/`Reserved` is changed to Cancelled/Deleted, the

license used by the account would becomes available for use by another individual.

- If the status of an account is Reserved, it can only be changed to 'Cancelled'.

Example

```
[C# Example]

int intAccount = 101000401;

//Put the account on hold

AdminService.AccountStatus(ACCOUNT_STATUS.HOLD);
```

AccountGetInfo, on page 15

AccountSetUserInfo, on page 26

AccountVerifyAgentInfoURLHash

Determines if hash included in the Agent Info URL is valid. You can use this API to validate the hash extracted from the URL. Validation is based on the contents of the hash and the date the hash was generated. If the contents are valid and the hash was generated the day or the day before the request, then the API determines the request is from an authorized and authentic Agent.

Parameters

Name	Description	Type	Max String Size
AccountNumber	The account number associated with the Agent.	xsd:int	
Hash	Hash string obtained from the Agent Info URL	xsd:string	

Return Values

Name	Description
HashCorrect	If the hash is valid, returns `True`; else returns `False`.

Error Codes

Code	Reason
1014	Access denied. Logged-in Technician is not authorized to access resources.
1016	The specified account cannot be found on the system.

Remarks

- The intent of this API is to provide a means to verify that the request was sent from a computer associated with an authorized, registered account within two days from the time the request was made.
- This verification does not guarantee that the URL originated from the Agent.
- This verification does not guarantee that the URL was not intercepted and reused by a third party.
- It is possible for a request to originate from the computer associated with the account, but not from the account holder. For example, an unauthorized person who has gained access to a registered user's computer. Use the AccountVerifyUserCredentials in addition to this API to verify the request is coming from a registered account holder.

AccountVerifyUserCredentials, below

AccountVerifyUserCredentials

Verifies whether a user with the specified account number or e-mail address exists in the Data Center and if the specified password matches the account password. If the password is verified, returns *Approved* set to `True`.

Parameters

Name	Description	Type
AccountNumber	The account number of the account to be verified.	xsd:int
Password	The user's account password.	xsd:string

Return Values

Name	Description	Type
Approved	Indicates acceptance of credentials. The return value is set to `True` if the password is valid. Otherwise it is set to `False`.	xsd:boolean

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.

Code	Reason
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1028	This account has been locked due to too many unsuccessful login attempts.
1068	Unable to authenticate user. Either the account or password is incorrect.

AccountGetInfo, on page 15

AccountSetStatus, on page 31

Community APIs

The available APIs for accessing community attributes include:

- CommunityChangeName, on the next page
- CommunityCreate, on page 37
- CommunityCreateInServerGroup, on page 38
- CommunityDisableRegistration, on page 39
- CommunityEnableRegistration, on page 40
- CommunityFind, on page 41
- CommunityFindAccounts, on page 43
- CommunityFindFedAuthAccounts, on page 44
- CommunityGetChangedAccounts, on page 45
- CommunityGetChangedAccountsEx, on page 46
- CommunityGetChangedCommunities, on page 48
- CommunityGetInstall, on page 49
- CommunityGetLicenseCount, on page 50
- CommunityGetName, on page 51
- CommunityGetParent, on page 53
- CommunityGetStatisticsInfo, on page 54
- CommunityGetSubCommunityIDs, on page 55
- CommunityGetTechnicians, on page 56
- CommunityReserveTicket, on page 57
- CommunityReserveTicketandFetch, on page 58
- CommunitySetLicenseCount, on page 60

CommunityChangeName

Changes the name of the specified community.

Parameters

Name	Description	Type	Max String Size
CommunityID	The ID of the community that you want to change.	xsd:int	
CommunityName	The new community name.	xsd:string	64

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1003	Access denied. Logged-in Technician does not have permission ‘Modify Communities’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1020	A community with the specified name already exists.
1021	Unable to authenticate user. Either the account or password is incorrect.
1029	Community names cannot include the character greater-than symbol, (>).

Remarks

- The logged-in technician must have the **Modify Communities** permission.
- Community names are not case-sensitive. This API allows changing capitalization of name.

Example

```
[C# Example]
```

```
int intCmtyId = 15;

//Change the name of our community

AdminService.CommunityChangeName(intCmtyId, "Changing name to this");
```

CommunityCreate, below

CommunityGetName, on page 51

CommunityCreate

Creates a subcommunity (`CommunityID`, `CommunityName`) in the specified parent community (`ParentCommunityID`). Logged-in technician must have the **Modify Communities** permission.

Parameters

Name	Description	Type	Max String Size
ParentCommunityID	The ID of the community where you want to create the subcommunity.	xsd:int	
CommunityName	The name of the new subcommunity.	xsd:string	64

Return Values

Name	Description	Type
CommunityID	The ID of the new subcommunity.	xsd:int

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1003	Access denied. Logged-in Technician does not have permission ‘Modify Communities’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1020	A community with the specified name already exists.

Code	Reason
1021	Unable to authenticate user. Either the account or password is incorrect.
1029	Community names cannot include the character greater-than symbol, (>).

Example

```
[C# Example]

//Create a new community

int intCmtyId = AdminService.CommunityCreate(intRootCmtyId, "This is my new
community");
```

Remarks

Logged-in technician must have the **Modify Communities** permission.

CommunityGetName, on page 51

CommunityChangeName, on page 36

CommunityCreateInServerGroup

Creates a subcommunity (`CommunityID`, `CommunityName`) in the specified server group (`ParentCommunityID`, `ServerGroup`). Logged-in technician must have the **Modify Communities** permission.

Parameters

Name	Description	Type	Max String Size
ParentCommunityID	The ID of the community where you want to create the subcommunity.	xsd:int	
CommunityName	The name of the new subcommunity.	xsd:string	64
ServerGroup	The name of the server group where you want to create the subcommunity.	xsd:int	

Return Values

Name	Description	Type
CommunityID	The ID of the new subcommunity.	xsd:int

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1020	A community with the specified name already exists.
1021	Unable to authenticate user. Either the account or password is incorrect.
1029	Community names cannot include the character greater-than symbol, (>).
1076	Invalid server group; the parent community does not exist in the specified server group.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
int nCommunityId = AdminService.CommunityCreateInServerGroup(5, "New Community
Name", 1);

Console.Writeline("New community created has an Id of " + nCommunityId);
```

CommunityDisableRegistration

Disables registration to the community (CommunityID) if it was enabled. Use this API to prevent any new users from registering to the community.

Parameters

Name	Description	Type
CommunityID	The ID of the community you want to disable.	xsd:int

Return Values

Name	Description
Success	If registration is disabled, returns `True`; if registration is already disabled or in case of any error listed below, returns `False`.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1003	Access denied. Logged-in Technician does not have permission 'Modify Communities'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Remarks

- Logged-in technician must have the **Modify Communities** permission.
- If registration is already disabled, the API does nothing and no error messages are returned.
- This API cannot be used to disable the root community (-1). Submitting -1 as the CommunityID returns error 1015 and does nothing.

Example

```
[C# Example]

int intCmtyId = 15;

//Turn off registration to the community

AdminService.CommunityDisableRegistration(intCmtyId);
```

CommunityEnableRegistration, below

AccountMoveToCommunity, on page 21

CommunityEnableRegistration

Enables registration to the community (CommunityID) if it was disabled. Logged-in technician must have the **Modify Communities** permission.

Parameters

Name	Description	Type
CommunityID	The ID of the community you want to enable.	xsd:int

Return Values

Name	Description
Success	If registration is enabled, returns `True`; if registration is already enabled or in case of any error listed below, returns `False`.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1003	Access denied. Logged-in Technician does not have permission 'Modify Communities'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Remarks

- Logged-in technician must have the **Modify Communities** permission.
- If registration is already enabled, the API does nothing and no error messages are returned.
- This API cannot be used to enable the root community (-1), which remains enabled by default. Submitting -1 as the CommunityID returns error 1015 and does nothing.

Example

```
[C# Example]

int intCmtyId = 15;

//Turn on registration to the community

AdminService.CommunityEnableRegistration(intCmtyId);
```

CommunityDisableRegistration, on page 39

AccountMoveToCommunity, on page 21

CommunityFind

Find all community IDs matching a specified parent community (ParentCommunityID) and community name (CommunityName).

Parameters

Name	Description	Type	Max String Size
ParentCommunityID	The parent community for which you want to find a community.	xsd:int	
CommunityName	The name of the community whose ID you want to find.	xsd:string	64

Return Values

Name	Description	Type
CommunityList	An array of community IDs.	Connected:CommunityFind_CommunityList_Array

An empty list is returned if the search is successful, but no matching results are found.

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
AdminAPICommunityInfo[] aCI = AdminService.CommunityFind(5, "Find Me");

foreach ( AdminAPICommunityInfo Temp in aCI)
{
        Console.Writeline("Community ID: " + Temp.nCommunityID):
        Console.Writeline("Parent Community ID: " + Temp.nParentId);
        Console.Writeline("Parent Name: " + Temp.strParentCommunityName);
}
```

CommunityFindAccounts

Find all accounts that match specified search criteria, including accounts in subcommunities of the specified community (CommunityID).

Parameters

Name	Description	Type	Max String Size
CommunityID	The community you want to find accounts in; the starting point of the search.	xsd:int	
FieldName	The field to match Must be either `LoginID` or `Email`.	Connected:Searchfield	
FieldValue	Text to match. Must be the LoginID supplied in registration or e-mail address. If this value is blank, no search is performed.	xsd:string (LoginID) xsd:string (Email)	64 100

Status	Find accounts with the specified status; the value of this field must be one of the following values:
	`ACCOUNT_NOSTATUS`
	`ACCOUNT_ANY`
	`ACCOUNT_INUSE` *****
	`ACCOUNT_DELETED`
	`ACCOUNT_RESERVED`
	`ACCOUNT_ONHOLD`
	`ACCOUNT_CANCEL`
	`ACCOUNT_ACTIVE`
	
	*****"`INUSE`" indicates the account is either Active or On Hold
		Connected: ACCOUNT_STATUS	

Return Values

If successful and matching results are found, this API returns an array called **AdminAPIBaseAccountInfoList** that contains AdminAPIBaseAccountInfo structures.

An empty list is returned if the search is successful, but no matching results are found.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

CommunityGetChangedAccounts, on the next page

CommunityFindFedAuthAccounts, below

AccountGetInfo, on page 15

AccountSetAgentSetupID, on page 28

AccountSetStatus, on page 31

AccountMoveToCommunity, on page 21

CommunityFindFedAuthAccounts

Find all accounts that match a particular federated authentication User ID in a specific community.

Parameters

Name	Description	Type	Max String Size
CommunityID	The community you want to find accounts in.	xsd:int	
AccountUID	The User ID whose accounts you want to find.	xsd:string	128

Return Values

If successful and matching results are found, this API returns an array called **AdminAPIBaseAccountInfoList** that contains AdminAPIBaseAccountInfo structures.

An empty list is returned if the search is successful, but no matching results are found.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

CommunityGetChangedAccounts, below

CommunityFindAccounts, on page 43

AccountGetInfo, on page 15

AccountSetAgentSetupID, on page 28

AccountSetStatus, on page 31

AccountMoveToCommunity, on page 21

CommunityGetChangedAccounts

Returns a list of all user accounts and the account information that changed after the specified date. An account is considered changed if there are any changes to the user information including Name, Address, Phone, as well as status changes and community assignment changes. You can use a bitmask to return a subset of user or account information.

Parameters

Name	Description	Type
CommunityID	The community where you want to search for accounts that have recently changed. Subcommunities of the specified community are included in the search.	xsd:int
Date	The date on or after which the account changes occurred; use with the *EndDate* parameter to search for accounts that changed within a specific date range.	xsd:string

Return Values

Name	Description	Type
AccountChange List	An array of account numbers. The array comprises a list of accounts that changed on or after the specified *Date*.	Connected:Community GetChangedAccounts_ AccountChangeList_Array
EndDate	Date of the last account change found.	xsd:date

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1069	The specified date is not a valid date.

Remarks

- Accounts in subcommunities of the specified community are also returned.
- May be used to notify another application or portal of changes to user information.
- Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:date type.

AccountGetInfo, on page 15

CommunityGetChangedAccountsEx

Returns a list of all user accounts and the account information that changed after the specified date and time. An account is considered changed if there are any changes to the user information including Name, Address, Phone, as well as status changes and community assignment changes. You can use a bitmask to return a subset of user or account information. This call is similar to CommunityGetChangedAccounts, on the previous page, but this call includes both date and time information.

Parameters

Name	Description	Type
CommunityID	The community where you want to search for	xsd:int

Name	Description	Type
	accounts that have recently changed. Subcommunities of the specified community are included in the search.	
DateTime	The date and time on or after which the account changes occurred; use with the *EndDateTime* parameter to search for accounts that changed within a specific date range.	xsd:dateTime

ChangeMask	Indicates the type of bitmask:
	`MODIFICATIONSBITMASK_ALL`
	`MODIFICATIONSBITMASK_OTHER`
	`MODIFICATIONSBITMASK_USER_INFO`
		Connected: MODIFICATIONSBITMASK

Return Values

Name	Description	Type
AccountChangeList	An array of account numbers. The array comprises a list of accounts that changed on or after the specified *DateTime*.	Connected:CommunityGetChangedAccounts_AccountChangeList_Array
EndDateTime	Date and time of the last account change found.	xsd:dateTime

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1069	The specified date is not a valid date.

Remarks

- Accounts in subcommunities of the specified community are also returned.
- May be used to notify another application or portal of changes to user information.
- Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:dateTime type.

AccountGetInfo, on page 15

CommunityGetChangedCommunities

Returns a list of all communities that changed during the specified period. Information includes any existing communities changes as well as new communities.

Parameters

Name	Description	Type
ParentCommunityID	The community where you want to start the search for communities that have recently changed. Subcommunities of the specified community are included in the search.	xsd:int
Date	The date on or after which the community changes occurred; use with the *EndDate* parameter to search for communities that changed within a specific date range.	xsd:date
EndDate	The date before which the community changes occurred; used in conjunction with the Date parameter when searching for communities that changed within a specified date range.	xsd:date

Return Values

Name	Description	Type
Community ChangeList	The community ID for each changed community. This information comes from the Registry.ChangedCommunity table.	Connected:CommunityGetChangedCommunities_ CommunityChangeList_ Array
Count	Size of CommunityChangeList.	xsd:int

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.

Code	Reason
1015	The community does not exist.
1075	Invalid date range.

Remarks

- Communities in subcommunities of the specified parent community are also returned.
- Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:date type.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
DateTime dtStartDate = DateTime.UtcNow.AddDays(-30);
DateTime dtEndDate = DateTime.UtcNow;
int[] anChangeCommunityIds AdminService.CommunityGetChangedCommunities(nRoot_
Community, dtStartDate.Date, dtEndDate.Date);
int[] anAccounts;
foreach (int nID in anChangeCommunityIds)
{
  int[] anData = AdminService.CommunityGetChangedAccounts(nID, dtStartDate.Date,
MODIFICATIONSBITMASK.MODIFICATIONSBITMASK_OTHER, out dtEndDate);
  foreach (int nAccount in anData)
  {
     Console.Writeline(nAccount);
  }
}
```

CommunityGetInstall

Returns data that can be used to create a PC Agent installation file for a selected parent community (`ParentCommunityID`) and Agent configuration (`ConfigurationID`).

Parameters

Name	Description	Type
ParentCommunityID	The parent community that contains the Agent configuration you want to get.	xsd:int
ConfigurationID	The ID of the Agent configuration you want to download.	xsd:int

Return Values

If successful, the API returns a binary byte array that you can save as an Agent Setup file (for example, `AgentSetup.msi`).

Remarks

This API lets you create an Agent Setup file for a PC Agent.

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1078	Community does not contain the Agent Setup specified.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
byte[] abInstall = AdminService.CommunityGetInstall(5, 9);

FileStream fsWriter = new FileStream("setup.msi", FileMode.Create);
foreach (byte bTemp in abInstall)
{
        fsWriter.WriteByte(Byte);
}
fsWriter.Close();
```

CommunityGetLicenseCount

Returns the number of licenses allocated to a community for PC Agents.

Parameters

Name	Description	Type
CommunityID	The community for which you want to obtain the number of allocated licenses for PC Agents.	xsd:int
ProductCode	Indicates the type of product: `PRODUCTCODE_PC_AGENT`	Connected: ProductCode

Return Values

Name	Description	Type
LicenseCount	The number of licenses allocated to the community for a specific product type. If access to the community is denied (no licenses are allocated to the community), the return value is SOAP fault `1070`. If the community inherits licenses from its parent community, the return value is `-1`. If the community has an unlimited number of licenses, the return value is `-2`.	xsd:int

Error Codes

Code	Reason
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1030	The Data Center is not licensed for this product.

Example

```
[C# Example]

//Get the license count for CommunityId 19 and print it
//Get the count for PC Agent licenses first

int nCmtyId = 19;

PRODUCTCODE ePCode = PRODUCTCODE.PRODUCTCODE_PC_AGENT;
int nCount = AdminService.CommunityGetLicenseCount(nCmtyId, ePCode);
Console.WriteLine("CommunityId {0} has PC license count of: {1}", nCmtyId, nCount);
```

CommunitySetLicenseCount, on page 60

CommunityGetName

Returns the full and short community names for the specified community ID (CommunityID).

Parameters

Name	Description	Type
CommunityID	The ID of the community you want to return the names of.	xsd:int

Return Values

If successful, returns **AdminAPICommunityNames**, which contains the following information:

Name	Description	Type	Max String Size
FullName	Canonical name of the specified community ID.	xsd:string	64
ShortName	The short community name (community name only).	xsd:string	64

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Example

```
[C# Example]

int intCmtyId = 15;

//Get the community name

AdminAPICommunityNames cAPICmtyNames = AdminService.CommunityGetName(intCmtyId);
Console.WriteLine("Canonical Name: {0}", cAPICmtyNames.strFullName);
Console.WriteLine("Short Name: {0}", cAPICmtyNames.strShortName);
```

CommunityGetSubCommunityIDs, on page 55

CommunityGetParent

Returns the parent (ParentCommunityID) of the given community.

Parameters

Name	Description	Type
CommunityID	The ID of the community you want to return the parent of.	xsd:int

Return Values

Name	Description	Type
ParentCommunityID	The ID of the community you want to return a list of subcommunities for	xsd:int

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Remarks

If *CommunityID* is `-1` (that is, the Data Center root community), the return *ParentCommunityID* will be `-1`.

Example

```
int nAccount = 101000001;
AdminAPIAccountInfo cAcntInfo = AdminService.AccountGetInfo(nAccount);
int nAccounts_CommunityID = cAcntInfo.BaseAccountInfo.nCommunityID;
int nAccounts_Parent_CommunityID = AdminService.CommunityGetParent(nAccounts_
CommunityID);
```

CommunityGetStatisticsInfo

Returns the following statistics for a given community (CommunityID):

- Number of accounts
- Number of licenses in use
- Number of licenses available
- Uncompressed tip revision size. (This information allows for identification of the relative size of the community data.)

Parameters

Name	Description	Type
CommunityID	The ID of the community you want to return statistics for.	xsd:int

Return Values

Name	Description	Type
CommunityStatisticsInfo	A data structure that contains statistics for the specified community, including: • number of accounts • number of licenses in use • number of licenses available • uncompressed tip revision size (allows for identification of the relative size of the community data)	Connected: AdminAPICommunityStatisticsInfo, on page 95

Error Codes

Code	Reason
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Example

```
int nRoot_Community = AdminService.SessionLogin(sTechName, sPassword);
AdminAPICommunityStatisticsInfo CSI = AdminService.CommunityGetStatisticsInfo(5);

Console.Writeline("Name: " + CSI.strCommunityName);
Console.Writeline("Account Count: " + CSI.nAccountCount);
Console.Writeline(" License Count Available: " + CSI.nLicenseCountAvailable);
Console.Writeline(" License Count in Use: " + CSI.nLicenseCountInUse);
Console.Writeline(" Tip Revision Uncompressed Size: " +
CSI.lTipRevisionUncompressedSize);
```

CommunityGetSubCommunityIDs

Returns a list of all the child or subcommunities of the specified parent community.

Parameters

Name	Description	Type
ParentCommunityID	The ID of the community you want to return a list of subcommunities for	xsd:int

Return Values

If successful, returns SubCommunityIDs, an array of CommunityIDs.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Example

```
[C# Example]

int intCmtyId = 15;

//Get a list of ids of subcommunities in this community
```

```
int[] intASubCmtyIDs =AdminService. CommunityGetSubCommunityIDs(intCmtyId);
if ( intASubCmtyIDs != null )
{
  foreach(int x in intASubCmtyIDs )
  {
     Console.Write("{0} ", x)
  }
}
```

CommunityCreate, on page 37

CommunityChangeName, on page 36

CommunityGetName, on page 51

CommunityGetTechnicians, below

CommunityGetTechnicians

Returns a list of all the technicians in the specified community.

Parameters

Name	Description	Type
CommunityID	The community for which you want to get a list of technicians.	xsd:int

Return Values

If successful, returns TechIDs, an array of type AdminAPITechnicianID, on page 98.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.

Example

```
[C# Example]

int intCmtyId = 15;
```

```
//Get a list of technicians in this community

AdminAPITechnicianID[] TechArray = AdminService.CommunityGetTechnicians(intCmtyId);
if ( intASubCmtyIDs != null )
{
  foreach(AdminAPITechnicianID T in TechArray )
  {
     Console.WriteLine("{0}:{1} ", T.strTechName, T.nCommunityID)
  }
}
```

CommunityCreate, on page 37

CommunityChangeName, on page 36

CommunityGetName, on page 51

CommunityGetTechnicians, on the previous page

CommunityGetSubCommunityIDs, on page 55

CommunityReserveTicket

> **DEPRECATED:**
> This API is deprecated. It uses the CommunityReserveTicketandFetch, on the next page API to reserve accounts and return the account number.

Reserves an account for future registration. Also sets the user information for the reserved account, including the account's community, Agent Setup and license code.

Parameters

Name	Description	Type
CommunityID	Community in which to reserve the account.	xsd:int
AgentSetupID	Agent Setup ID assigned for the future account. To use the default Agent Setup for the specified community, set AgentSetupID to zero (0).	xsd:int
AdminAPIUserInfo, on page 98	Information about the user of the reserved account. This info is optional except LoginID field, which identifies the reserved account. LoginID may be obtained from another source such as an enterprise directory server.	Connected:Admin APIUserInfo
ProductCode	Indicates the type of product license to reserve for the account: `PRODUCTCODE_PC_AGENT`	Connected:ProductCode

Return Values

If successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1012	Credit Card Expiration Date is not a valid date.
1013	Credit Card type is invalid. Valid credit card types are Visa, MasterCard and AMEX.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1018	Credit Card number is invalid.
1024	Unable to perform required action. The destination community does not have enough licenses available.
1026	Community this account belongs to does not contain the Agent Setup specified.
1035	Agent Setup ID is not enabled.
1063	Access denied. Logged-in Technician does not have permission 'Reserve Tickets'.
1066	Cannot reserve account for empty Logon ID.

Remarks

- Logged-in technician must have the **Reserve Accounts** permission enabled.
- If supplying the credit card information, see the rules for AccountSetUserInfo, on page 26.
- More than one ticket may be reserved for the same LoginID. This is useful if the individual account holder has more than one computer to back up.

CommunityCreate, on page 37

CommunityReserveTicketandFetch

Reserves an account for future registration and returns the account number for the reserved account. Also sets the user information for the reserved account, including the account's community, Agent Setup and license code.

Parameters

Name	Description	Type
CommunityID	Community in which to reserve the account.	xsd:int
AgentSetupID	Agent Setup ID assigned for the future account. To use the default Agent Setup for the specified community, set AgentSetupID to zero (0).	xsd:int
AdminAPIUserInfo, on page 98	Information about the user of the reserved account. This info is optional except LoginID field, which identifies the reserved account. LoginID may be obtained from another source such as an enterprise directory server.	Connected:Admin APIUserInfo
ProductCode	Indicates the type of product license to reserve for the account: `PRODUCTCODE_PC_AGENT`	Connected:ProductCode

Return Values

If successful, this API returns the reserved account number in an array called **AdminAPIBaseAccountInfoList** that contains AdminAPIBaseAccountInfo, on page 95 data structures.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1012	Credit Card Expiration Date is not a valid date.
1013	Credit Card type is invalid. Valid credit card types are Visa, MasterCard and AMEX.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1018	Credit Card number is invalid.
1024	Unable to perform required action. The destination community does not have enough licenses available.
1026	Community this account belongs to does not contain the Agent Setup specified.

Code	Reason
1035	Agent Setup ID is not enabled.
1063	Access denied. Logged-in Technician does not have permission 'Reserve Tickets'.
1066	Cannot reserve account for empty Logon ID.

Remarks

- Logged-in technician must have the **Reserve Accounts** permission enabled.
- If supplying the credit card information, see the rules for AccountSetUserInfo, on page 26.
- More than one ticket may be reserved for the same LoginID. This is useful if the individual account holder has more than one computer to back up.

CommunityCreate, on page 37

CommunitySetLicenseCount

Allocates a specified number of licenses in a community for PC Agents.

Parameters

Name	Description	Type
CommunityID	The community to which you want to allocate licenses for a specific product type.	xsd:int
ProductCode	Indicates the type of product license to reserve for the account: `PRODUCTCODE_PC_AGENT`	Connected:ProductCode
LicenseCount	The number of licenses that you want to allocate to the community for a specific product type. This value cannot exceed the number of unused licenses available to the community. To configure the community to inherit licenses from its parent community, specify 0 (zero) as the LicenseCount. To deny access to this community, do not specify a LicenseCount value.	xsd:int

Return Values

If successful, none.

Error Codes

Code	Reason
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1030	The Data Center is not licensed for this product.
1031	The allocated license count value is invalid.

Remarks

The logged-in technician must have the **Allocate Licenses to Subcommunities** permission enabled.

Example

```
[C# Example]

int nCmtyId = 19;

//Allocate 200 PC Agent licenses to the community

ePCode = PRODUCTCODE.PRODUCTCODE_PC_AGENT;
nCount = 200;
AdminService.CommunitySetLicenseCount(nCmtyId, ePCode, nCount);
```

CommunityGetLicenseCount, on page 50

Session APIs

The available interfaces for session features include:

- SessionLoginTechnician, below
- SessionLogoutTechnician, on page 63

SessionLoginTechnician

Starts a SOAP session with Support Center using the specified technician's name and password.

Parameters

Name	Description	Type	Max String Size
TechName	Technician log in name. It is case-insensitive.	xsd:string	64
Password	Technician password. It is case-sensitive.	xsd:string	See Password Restrictions

Return Values

Name	Description
CommunityID	Returns the root community ID of the logged in technician.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1030	Unable to authenticate technician. Either the Technician ID or password is incorrect, or there is more than one technician with submitted credentials.
1031	The current password has expired.

Remarks

- The technician account used to establish the session must have the **Use Scripting** permission to make calls to the APIs. Some APIs have additional permission requirements. Refer to the documentation for each API to determine if any additional permissions are required for the calls you want to make.
- To avoid exposing your password, make sure that the script you use gets the password from some secure location, such as an encrypted file on disk.
- The C# wrapper class APISession contains code to securely store the technician password in the OS the first time it is asked for, and then uses it thereafter if the user logged in is the same user who supplied the password.
- Use TechnicianGetPasswordExpiryDate to determine when your password expires and make sure you change it and the script before your password expires.
- If there are three unsuccessful attempts to log on, the account will be locked.

- Support Center allows creation of technicians using the same user name and password in different communities. This API assumes that there aren't any such duplicate technician user names. If the same user name is used for technicians in different communities, the duplicate technicians cannot access these APIs. The error 1030 is returned on attempt to authenticate a duplicate technician user ID.
- To prevent this problem, run the SQL query provided below to identify duplicate technicians. If duplicates exist, change one of the duplicate user names to a unique name.

```
select techid, min(permissionvalue), max(permissionvalue), min(rootcommunityid),
max(rootcommunityid),count(*) from techpermission
where permissiontype = 'pwhash1'
group by techid
having count(*) > 1
and min(permissionvalue) = max(permissionvalue)
```

Example

```
[C# Example]

//Login and set a cookie so that multiple calls
//can be made during a single session, display the
//root community ID of the logged-in technician,
//display the password expiration date, then logout.

AdminAPIService AdminService = new AdminAPIService()

AdminService.CookieContainer = new System.Net.CookieContainer();
AdminService.PreAuthenticate = true;
AdminService.Credentials = System.Net.CredentialCache.DefaultCredentials;

string strTechName = "druidia";
string strTechPassword = "Boston1822";
int intRootCmtyId = AdminService.SessionLoginTechnician (strTechName,
strTechPassword);

DateTime dtExp = AdminService.TechnicianGetPasswordExpiryDate();
Console.WriteLine("Technician {0} is logged into community ID:{1}.", strTechName,
intRootCmtyId);
Console.WriteLine("Password for technician {0} expires on {1}.", strTechName,
dtExp.ToString("MM/dd/yyyy"));

AdminService.SessionLogoutTechnician();
```

SessionLogoutTechnician, below

SessionLogoutTechnician

Log out of a session.

Parameters

None.

Return Values

None.

Remarks

If SessionLogoutTechnician is not called, it will be automatically abandoned by IIS after the session time out has passed (The default session time-out value is twenty (20) minutes.) However, calling this function to end a session is recommended, since it frees up resources on the Web server.

Example

```
[C# Example]

//Login and set a cookie so that multiple calls
//can be made during a single session, display the
//root community ID of the logged-in technician,
//display the password expiration date, then logout.

AdminAPIService AdminService = new AdminAPIService()

AdminService.CookieContainer = new System.Net.CookieContainer();
AdminService.PreAuthenticate = true;
AdminService.Credentials = System.Net.CredentialCache.DefaultCredentials;

string strTechName = "druidia";
string strTechPassword = "Boston1822";
int intRootCmtyId = AdminService.SessionLoginTechnician (strTechName,
strTechPassword);

DateTime dtExp = AdminService.TechnicianGetPasswordExpiryDate();
Console.WriteLine("Technician {0} is logged into community ID:{1}.", strTechName,
intRootCmtyId);
Console.WriteLine("Password for technician {0} expires on {1}.", strTechName,
dtExp.ToString("MM/dd/yyyy"));

AdminService.SessionLogoutTechnician();
```

SessionLoginTechnician, on page 61

Reports APIs

The available interfaces for reports include:

ReportTemplateRun

Runs a defined report in the specified community. This API returns the report name immediately, without waiting for all of the report results. Use the ReportGet API to get the actual report results.

Parameters

Name	Description	Type
AdminAPIReportTemplateID, on page 98	A data structure that contains the name of the report template to run and ID of the community where the report was created.	Connected:AdminAPIReportTemplateID
IncludeSubCommunites	Determines whether to include information from the subcommunities of the specified community.	xsd:boolean
ReportStartDate	The start of the date range to use when collecting report information.	xsd:dateTime
ReportEndDate	The end of the date range to use when collecting report information.	xsd:dateTime

Return Values

If the template is successfully run, returns the name of the report results for the specified community and report template.

Name	Description	Type	Max String Size
ReportName	Name of the report output. If the name was not passed in with the ReportName parameter, then the default format is used: <*CommunityID*>_8_<*NameOfReport*>.scr	xsd:string	64

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1004	Access denied. Logged-in Technician does not have permission 'Run Reports'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist
1057	A report with this name already exists, or is currently running in the queue.
1059	Specified date range is invalid: start date is after the end date.
1065	Report Template does not exist in requested community.

Remarks

- Logged-in technician must have the **Run Reports** permission.
- To include information from all subcommunities in the specified community, specify IncludeSubCommunites as True.
- A report with the requested name and CommunityID must exist in Support Center.
- The report can be self-contained, requires no additional information to run, or requires the date range. It also applies to the Custom reports that have Active Run Screen type.
- To use this API to run a custom report, make sure the custom report's Run Screen type is set to Custom and that all parameters for the run screen have default values (no validation for the parameters is performed). Refer to Support Center Help for more information about running custom Support Center reports.
- Report dates must be in GMT (UTC) time. All report times are converted to the server time before running the report. Support Center displays report data using server time.
- Start and end dates before 1970 are invalid. If invalid report dates are specified for reports that require a date, the report template defaults are used to generate the report. For reports that do not require dates, invalid dates are ignored and no dates are used to generate the report results.

  Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:date type.

ReportDelete, on the next page

ReportGet, below

ReportGet

Uses the report name returned by the ReportTemplateRun API to retrieve report results.

Parameters

Name	Description	Type
AdminAPIReportTemplateID, on page 98	A data structure that contains the name of the report template to run and ID of the community where the report was created.	Connected:AdminAPIReportID

Return Values

Name	Description	Type
ReportXML	An XML buffer for the requested report.	xsd:string

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1004	Access denied. Logged-in Technician does not have permission 'Run Reports'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist
1019	Report does not exist in requested community.
1100	Report is currently running in the queue.

Remarks

- The logged-in technician has must have the **Run Reports** permission.
- Error code 1100 does not indicate an error condition, just that the report has not finished running yet. Wait and call the API again to obtain the report results.

ReportDelete, below

ReportDelete

Deletes report output in the specified community.

Parameters

Name	Description	Type
AdminAPIReportTemplateID, on page 98	Structure that contains the report name and community ID.	Connected:AdminAPIReportID

Return Values

If report output is successfully deleted, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1004	Access denied. Logged-in Technician does not have permission ‘Run Reports’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist

Remarks

- The logged-in technician must have the **Run Reports** permission enabled.
- Reports deleted from a parent community are no longer available to any subcommunities that inherit them.
- If specified report does not exist, there is no error returned and API does nothing.

ReportTemplateRun, on page 65

Technician APIs

The technician APIs allow you to change values for technician IDs and passwords. The available interfaces are:

- AccountOrderMedia, on page 22
- TechnicianDelete, on page 71
- TechnicianGetPasswordExpiryDate, on page 71
- TechnicianGetPasswordExpiryDateTime, on page 73

TechnicianCreate

Creates a new technician user ID within the specified community and grants the same permissions as specified technician.

Parameters

Name	Description	Type	Max String Size
TechID	Identifies the technician. See the table below.	Connected: AdminAPI TechnicianID	64
TechPassword	Technician temporary password. If the community that the technician is being added to is an enterprise directory community, the password should be empty or it will be ignored. Instead, the API verifies that the new TechID exists in the enterprise directory server.	xsd:string	
SameAsTechID	The ID of an existing technician that possesses the same set of permissions that you want to grant to the new technician. The permissions granted to the new technician will be equal to or less than the set possessed by the currently logged in technician.	Connected: AdminAPI TechnicianID	
AdminAPI TechnicianID	*CommunityID* is the ID of the community where you want to create the technician (also known as the technician's "root community") *TechName* is the unique login name you want to assign to the technician.	xsd:int xsd:string	-- 64

Return Values

When successful, nothing is returned.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1002	Access denied. Logged-in Technician does not have permission ‘Modify Technician Permissions’.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist
1032	The Technician ID that you are trying to add is already associated with an existing technician.
1062	The Technician Login ID cannot be empty.
1063	The password provided does not conform to requirements. All passwords must be at least 8 characters long, including at least one numeric character.
1064	"SameAsTechID" does not exist.

Remarks

- The currently logged-in technician cannot grant any permissions that he himself does not have.
- If the community into which the technician is being added is an enterprise directory community, you may leave the password empty since it will be ignored. This API verifies that the new TechID exists in the enterprise directory server.

Examples

```
[C# Example]

//This is the new technician information:
AdminAPITechnicianID Create = new AdminAPITechnicianID();

Create.nCommunityID = 5;
Create.strTechName = "NewTechnician";

//This is the existing technician that we want to use
//as a model for the new technician's permissions:
AdminAPITechnicianID As = New AdminAPITechnicianID();

As.nCommunity = 5;
As.strTechName = "MyTechnicianLoginName";

adminService.TechnicianCreate(Create "NewPass1", As);
```

TechnicianGetPasswordExpiryDate, below

TechnicianDelete, below

TechnicianDelete

Deletes the specified technician from the specified community.

Parameters

Name	Description	Type
TechID	Identifies the technician.	Connected: AdminAPI TechnicianID

Return Values

Name	Description
Success	If the technician is deleted, returns `True`; if technician was not found or in case of any error listed below, returns `False`.

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.
1002	Access denied. Logged-in Technician does not have permission 'Modify Technician Permissions'.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1027	Unable to perform required action. A technician cannot modify him/herself.

Remarks

- The logged-in technician account that is making the call must have the **Modify Technician Permissions** password to delete another technician.
- If specified technician does not exist in the system, there is no error returned and API does nothing.

AccountOrderMedia, on page 22

TechnicianGetPasswordExpiryDate

Get the date that the currently logged in technician will expire.

Parameters

None.

Return Values

Name	Description	Type
Date	The date the password will expire.	xsd:date

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission 'Scripting'.

Remarks

Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:date type.

Example

```
[C# Example]
//Login and set a cookie so that multiple calls
//can be made during a single session, display the
//root community ID of the logged-in technician,
//display the password expiration date, then logout.

AdminServiceAPI adminService = new AdminServiceAPI()

adminService.CookieContainer = new System.Net.CookieContainer();
adminService.PreAuthenticate = true;
adminService.Credentials = System.Net.CredentialCache.DefaultCredentials;

string strTechName = "TechAccount1";
string strTechPassword = "NewPass1";
int intRootCmtyId = AdminService.SessionLoginTechnician (strTechName,
strTechPassword);

DateTime dtExp = AdminService.TechnicianGetPasswordExpiryDate();

Console.WriteLine("Technician {0} is logged into community ID:{1}.", strTechName,
intRootCmtId);
Console.WriteLine("Password for technician {0} expires on {1}.", strTechName,
```

```
dtExp.ToString("MM/dd/yyyy"));

AdminService.SessionLogoutTechnician();
```

TechnicianGetPasswordExpiryDateTime

Get the date and time that the currently logged in technician will expire. This call is similar to TechnicianGetPasswordExpiryDate, on page 71, but this call includes both date and time information.

Parameters

None.

Return Values

Name	Description	Type
DateTime	The date and time the password will expire.	xsd:dateTime

Error Codes

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.

Remarks

Refer to http://www.w3.org/TR/xmlschema-2/#date for a description of the xsd:dateTime type.

Example

```
[C# Example]
//Login and set a cookie so that multiple calls
//can be made during a single session, display the
//root community ID of the logged-in technician,
//display the password expiration date, then logout.

AdminServiceAPI adminService = new AdminServiceAPI()

adminService.CookieContainer = new System.Net.CookieContainer();
adminService.PreAuthenticate = true;
adminService.Credentials = System.Net.CredentialCache.DefaultCredentials;

string strTechName = "TechAccount1";
string strTechPassword = "NewPass1";
int intRootCmtyId = AdminService.SessionLoginTechnician (strTechName,
strTechPassword);

DateTime dtExp = AdminService.TechnicianGetPasswordExpiryDateTime ();

Console.WriteLine("Technician {0} is logged into community ID:{1}.", strTechName,
intRootCmtId);
Console.WriteLine("Password for technician {0} expires on {1}.", strTechName,
dtExp.ToString("MM/dd/yyyy HH:mm:ss"));

AdminService.SessionLogoutTechnician();
```

Chapter 3: C# class library

This chapter describes the information used with the class libraries in the Web Services API.

Use the C# class library

Using the WSDL file installed with the Web Interface Service, you can create several C# classes. These classes provide API wrappers that you can use to wrap the SOAP requests and helper methods that you can use to handle the SOAP results.

The API wrapper methods:

- Accept the parameters
- Create a SOAP request
- Send the request to Support Center for processing
- Wait for the SOAP response and store it as member variables

The helper methods provide a way to get and handle the SOAP results.

The main wrapper classes are Community and Account. Each class is responsible for the appropriate set of API calls. Each class constructor accepts existing APISession class object after the LoginTechnician is called.

System requirements

You can use the class library in applications that run on one of the following operating systems: Windows® XP, Windows 7, Windows 8, Windows Server® 2003, Windows Server 2008, Windows Server 2012, or Windows Server 2016. These operating systems use stored user names and passwords to associate a set of credentials with a single Windows user account, storing those credentials using the Data Protection API (DPAPI).

The C# classes included in this library use the Credential Management API function CredUIPromptForCredentials to prompt for the technician password and securely store it in the Credential Manager. The Credential Manger is only available on the operating systems previously listed.

These classes provide two kinds of methods, API wrappers and helper methods. The API wrapper methods accept parameters, create a SOAP request, and send it to Support Center. When a SOAP response is received, the wrapper methods store it as member variables. You can get the results through the utility methods provided.

Create C# wrapper classes

If you are using Microsoft Visual Studio to write clients in C#, you can use the Microsoft Web Services Description Language Tool wsdl.exe to create a proxy class for use in your project.

To create the proxy classes

1. Open the `AdminAPI.wsdl` file in a browser, then save a copy of it to an XML file on your local drive.
2. Run the following command:

   ```
   wsdl /n:YourNamespace YourDrive:\YourPath\AdminAPI.wsdl
   ```

   This command generates a C# file that contains the Web Service proxy class you can include in your project. This class has a hard-coded URL in the constructor. This is the URL in the soap:location field in the original AdminAPI.wsdl file.

   You must change this URL to point to the Web Interface Service server that you will be working with. To do this, set the URL property on the instance of the WebService class. The URL must be in this form:

   ```
   "https://DNS name/AdminAPI Virtual Directory/AdminAPI.dll?Handler=Default"
   ```

   ```
   C# Example:
   ```

   ```
   AdminAPIService WebService = new AdminAPIService();
   ```

   ```
   WebService.Url =
   "https://www.connected.com/AdminAPI/AdminAPI.dll?Handler=Default";
   ```

`AdminAPIService` is the proxy class generated using the wsdl.exe utility program.

Class listing

Account class

Class Hierarchy

Account

public class account;

File

`Account.cs`

Description

AdminAPIUtil.Account holds complete information about an account including the user information stored in the User class. It provides the methods to get the User object, get and change account settings, move the account and order CDs.

To use this class, create an Account class object passing APISession and AccountNumber, and call Load() on the object to populate account and user information.

Namespace

`AdminAPIUtil`

Properties

Property	Description
AccountNumber	The ID number that identifies an account.
AccountSizeInfo	Account size information for this account.
AccountStartDate	The date the account was registered.
AgentInstallPath	The installation path of the Agent.
AgentSetupID	The ID of the Agent Setup assigned to this account.
AgentVersion	The version of the Agent assigned to this account.
CommunityID	The ID of the community where the account was created.
ComputerName	The name of the computer associated with this account.
Status	The current status of this account.
UserInfo	The personal information of the account holder.

Methods

Method	Description
Account	The account number.
GetCustomInfo	Get the custom information values for the account.
GetEncryptionKey	Get the encryption key of the account.
GetMediaCount	Get the number of units of media required to fulfill a media order for this account (for example, the total number of CDs).
Load	Load the object.
Move	Move the account to another community.
OrderMedia	Order a copy of the account data on the specified type of storage media.
SavePassword	Save the account password.
SaveUser	Save the account user information.
SetAgentSetupID	Change the Agent setup that is assigned to this account.
SetiRoamOff	Disable access to iRoam.
SetiRoamOn	Enable access to iRoam.
SetStatus	Change the status of the account.

Remarks

If any of the Get methods (with the exception of `GetAccountNumber`() and `GetEncryptionKey`()), are called before `Load`(), the program checks to see if account information was already loaded. If not, it calls `Load`() before returning the data.

Examples

```
[C#]

// Creating a new account class

  class MyClass
  {
      public static int Main()
      {
          string strTech = "Admin";
          APISession Session = new APISession();
          Session.LoginTechnician(strTech);
          Account Acc = new Account(Session, 101000001);
          Acc.Load();
       }
  }
```

User class, on page 89

CustomInfo class, on page 88

Account Size class

Class hierarchy

AccountSize

```
public class AccountSize;
```

File

`AccountSize.cs`

Description

AdminAPIUtil.AccountSize class holds the backup sizes at some point in time for the account specified in the Account Class. It wraps the AdminAPIAccountSize structure.

Namespace

`AdminAPIUtil`

Properties

Property	Description
FirstBackup	Determines if this is the first backup (`True`) or a subsequent backup (`False`).
NumArchives	Gets the number of archives for the specified account.
NumFilesPool	Gets the number of non-pool file revisions.
NumFilesUnique	Gets the number of non-pool file revisions.
SizePool	Gets the compressed size of pool files.
SizePoolUncompressed	Gets the uncompressed size of non-pool file revisions.
SizeUnique	Gets the compressed size of all non-pool file revisions (equal to size of archives).
SizeUniqueDelta	Gets the uncompressed, post-delta size of all non-pool file revisions.
SizeUniqueUncompressed	Gets the uncompressed size of all non-pool file revisions.
SnapShotDate	Gets the date the sizes were recorded.
TipRevisionNumFiles	Gets the number of files in tip revision set.
TipRevisionUncompressed	Gets the uncompressed size of tip revision size.

AdminAPIException class

Class hierarchy

SoapException

AdminAPIException

```
public class AdminAPIException : SoapException;
```

File

`AdminAPIException.cs`

Description

`AdminAPIUtil.AdminAPIException` class is an exception object used by the wrapper classes. It is derived from .NET exception class System.SoapException. In addition to the members of the base class it provides the API Name, the Error Code and the Error Message members. When response from

the API call comes in as a SOAP Fault, the AdminAPIException class is created and populated with the API name, and error details from the Soap Fault message. Then it is thrown to the caller.

Namespace

`AdminAPIUtil`

Properties

Property	Description
APIName	The name of the API.
ErrorCode	The numeric error code number.
ErrorMessage	A string containing the error description.

Examples

```
[C#]

// This example shows how to throw this exception.

    public void APIWrapperMethod()
    {
         try
         {
              m_APISession.GetWebService().APICall();
         }
         catch (SoapException e)
         {
              throw new AdminAPIException("AccountDisableiRoam", e);
         }
    }


// This sample shows how to the caller can catch it.

    class MyClass
    {
         public static int Main()
         {
              try
              {
                   string strTech = "Admin";
                   APISession Session = new APISession();
                   Session.LoginTechnician(strTech);
              }
              catch (AdminAPIException e)
              {
                   string strAPIName = e.APIName;
                   string strErrCode = e.ErrorCode;
                   int nErrCode = e.GetErrorCode();
                   string strError = e.ErrorMessage;
                   Console.WriteLine(strError);
              }
         }
    }
```

APISession class, on the next page

APISession class

Class hierarchy

APISession

```
public class APISession;
```

File

`APISession.cs`

Description

AdminAPIUtil.APISession class is a starting point of using the Web Interface Service APIs. It logs in the technician and maintains the session state to handle all subsequent 3rd party application or user requests.

Namespace

`AdminAPIUtil`

Properties

Property	Description
RootCommunity	The community that the technician logged into.
WebService	The URL of the server where the Web Interface Service is installed.

Methods

Method	Description
APISession	Initiates a session.
LoginTechnician	Passes the specified technician's credentials.
LogoutTechnician	Terminates a session.
TechnicianGetPassword ExpiryDate	Gets the specified technician's password expiration date to determine if it is still valid.
VerifyUserCredentials	Verifies the Agent user's login ID and password.
VerifyAgentInfoURLHash	Verifies that the hash in a URL spawned by an Agent associated with the user is valid.

Remarks

- To avoid exposing credentials in clear text in a script, `LoginTechnician`() does not accept the technician name or password as a parameter. It accepts the name of the resource, called `Target Name`, by which the Credential Manager stores the credentials. The target name should be a server name (can be DNS name) to identify the server the credentials that should be used to create the session. There are no special requirements for the name. `LoginTechnician`() calls the `SessionLoginTechnician`() API and passes in the technician Login ID and password.
- The APISession class uses CredUIPromptForCredentials Win32 API to store and retrieve the password from the Credential Manager in the OS. Since the library Credui.lib is a C library, the call has to be wrapped in a C++ dll for C# program to use through the DllImport attribute.
- The behavior of retrieving and storing credentials is as follows:
  - The program tries to retrieve the credentials from the Credential Manager stored under specified Target Name. If successful, the name and password are returned to the caller.
  - If Target Name is not found, a prompt will ask the user for a login name and password. When user clicks **OK**, the program tries to verify the credentials by logging on to Support Center.
  - If technician with the provided name and password is accepted, the credentials are saved in the Credential Manager by the supplied Target Name.
  - If the program fails to log in, the credentials are not saved but still returned to the caller.

AdminAPIException class, on page 80

Community Class

Class hierarchy

Community

```
public class Community;
```

File

`Community.cs`

Description

AdminAPIUtil.Community class represents a unit of administration and corresponds to the community stored on the server. It provides functionality to manage technicians, search for accounts, search for changed accounts and change the community settings. Each Community method wraps its corresponding API call.

Namespace

`AdminAPIUtil`

Methods

Method	Description
ChangeName	Calls the `CommunityChangeName` API to change the community name.
CreateSubCommunity	Calls the `ComunityCreate` API to create a new subcommunity in the specified parent community.
CreateTechnician	Calls the `TechnicianCreate` API to create a new technician.
DeleteReport	Calls the `ReportDelete` API to delete the specified report.
DeleteTechnician	Calls the `TechnicianDelete` to delete the specified technician.
DisableRegistration	Calls the `CommunityDisableRegistration` API to disable registration to the specified community.
EnableRegistration	Calls the `CommunityEnableRegistration` API to enable registration to the specified community.
FindAccounts	Calls the `CommunityFindAccounts` API to search for accounts in a specific community and its subcommunities based on a set of given criteria.
GetChangedAccounts	Calls the `CommunityGetChangedAccounts` to get a list of accounts that have changed in the specified community and its subcommunities.
GetName	Calls the `CommunityGetName` API to get both the short and full canonical name for the specified community ID.
GetReport	Calls the `ReportGet` API to get report results.
GetSubCommunityIDs	Calls the `CommunityGetSubCommunityIDs` API to return a list of subcommunities within the specified parent community.
GetTechnicians	Calls the `CommunityGetTechnicians` API to return a list of all technicians in the specified community.
ReserveTicket	Calls the `CommunityReserveTicket` API to reserve an account in the specified community.

Examples

[C#]

```
public class Community
       {
           private APISession  m_APISession;
           private int         m_nCommunityID;

           // Class constructor. Initializes data members: m_APISession, m_nCommunityID.
           //
           public Community(APISession Session, int nCommunityID)
           {
               m_APISession = Session;
               m_nCommunityID = nCommunityID;

               // The following reserves an account for later registration and sets
               // the user information.
               // The parameter nAgentSetupID is the Agent Setup ID assigned for
               // the future account.
               // UseInfo contains information about the user of the reserved account.
               // This information is optional.
               // The LoginID field is required. LoginID is normally the LoginID
               // into a 3rd party system and is used to identify the reserved ticket.
               // eCode is the product code: PRODUCTCODE_PC_AGENT
               // If there is an error during the call to CommunityReserveTicket API,
               // AdminAPIException is thrown.

               public void ReserveTicket(int nAgentSetupID, User UseInfo,  PRODUCTCODE eCode)
               {
                  try
                  {
                        AdminAPIUserInfo APIUserInfo = new AdminAPIUserInfo();
                        APIUserInfo.strAddress1 = UseInfo.Address1;
                        APIUserInfo.strAddress2 = UseInfo.Address2;
                        APIUserInfo.strCity = UseInfo.City;
                        APIUserInfo.strCompany = UseInfo.Company;
                        APIUserInfo.strCountry = UseInfo.Country;
                        APIUserInfo.strDepartment = UseInfo.Department;
                        APIUserInfo.strEmail = UseInfo.Email;
                        APIUserInfo.strFirstName = UseInfo.FirstName;
                        APIUserInfo.strLastName = UseInfo.LastName;
                        APIUserInfo.strLoginID = UseInfo.LoginID;
                        APIUserInfo.strMiddleName = UseInfo.MiddleName;
                        APIUserInfo.strState = UseInfo.State;
                        APIUserInfo.strTelephone = UseInfo.Telephone;
                        APIUserInfo.strZip = UseInfo.Zip;
                        APIUserInfo.CreditCardInfo.eCCType =
                            UseInfo.CreditCardInfo.CCType;
```

```
            APIUserInfo.CreditCardInfo.strCCExpDate =
                UseInfo.CreditCardInfo.CCExpDate;
            APIUserInfo.CreditCardInfo.strCCNumber =
                UseInfo.CreditCardInfo.CCNumber;
            m_APISession.WebService.CommunityReserveTicket(
                m_nCommunityID, nAgentSetupID, APIUserInfo, eCode);
        }
        catch (SoapException e)
        {
            throw new AdminAPIException("CommunityReserveTicket", e);
        }
    }
```

Remarks

- To use this class, create Account class object passing APISession and AccountNumber, and call Load() on the object to populate account and user information.
- Refer to AdminAPIException Class for more information about SOAP faults and error handling.
- Refer to the UserAdminAPICustomInfo class for information on getting and setting values that appear in an Agent's custom fields.

Account class, on page 77

CreditCard class

Class hierarchy

CreditCard

```
public class CreditCard;
```

File

`CreditCard.cs`

Description

AdminAPIUtil.CreditCard class holds credit card information for a specific account. This is a wrapper for the AdminAPICreditCard structure.

Namespace

`AdminAPIUtil`

Properties

Property	Description
CCType	The credit card type (Master Card, VISA, etc.) for the specified account number.
CCNumber	String that contains the credit card number for the specified account number.
CCExpDate	String that represents the credit card expiration date for the specified account number.

User class, on the next page

CustomInfo class

Class hierarchy

CustomInfo

```
public class CustomInfo;
```

File

`CustomInfo.cs`

Description

The AdminAPIUtil.CustomInfo class describes the names and value of the custom fields defined in an Agent for the account specified in the Account class.

Namespace

`AdminAPIUtil`

Properties

Property	Description
Attribute	The name of the custom field.
CustomField	Enumerated value of the custom field.
Value	The value of the custom field.

User class, on the next page

User class

Class hierarchy

User

```
public class User;
```

File

`User.cs`

Description

`AdminAPIUtil.User` class holds user basic information such as the user name and address, and description of custom fields and credit card info. It provides sets and gets methods for the members and allows Partners to get the information stored on the server as well as to set it on the server.

Namespace

`AdminAPIUtil`

Properties

Property	Description
Address1	Registered Agent user's street address.
Address2	Registered Agent user's street address, second line.
City	Registered Agent user's city.
Company	Registered Agent user's company.
Country	Registered Agent user's country.
CreditCardInfo	Structure containing Registered Agent user's credit card number, type and expiration date. (See CreditCard class, on page 87).
Department	Registered Agent user's department.
Email	Registered Agent user's e-mail address.
FirstName	Registered Agent user's first name.
LastName	Registered Agent user's last name.

Property	Description
LoginID	Registered Agent user's login name.
MiddleName	Registered Agent user's middle name or initial.
State	Registered Agent user's state.
Telephone	Registered Agent user's telephone.
Zip	Registered Agent user's zip code.

CreditCard class, on page 87

Chapter 4: Data structures

This chapter describes the data structures that the Web Services API specifies or returns.

- Structure listing, below

> **IMPORTANT:**
> AdminAPI is a paid feature. To use it, contact Account Management.

Structure listing

The Web Services API uses several data structures:

AdminAPIAccountInfo

This structure that contains information about the account including its start date, its Agent install path, Agent version, user information, account size and custom fields.

Name	Description	Type	Max String Size
AdminAPIBase	A collection of values common to	Connected:	

Name	Description	Type	Max String Size
AccountInfo	all accounts.	AdminAPIBaseAccountInfo, on page 95	
StartDate	The date the account was registered. This value is empty if the account status is "Reserved".	xsd:date	
AgentInstallPath	The installation path of the Agent; where it was installed. This value is empty if the account status is "Reserved".	xsd:string	255
AgentVersion	The Agent version (typically software and/or language version) assigned to the account. This value is empty if the account status is "Reserved".	xsd:string	
ComputerName	The name of the computer associated with the account. This value is empty if the account status is "Reserved".	xsd:string	255
AdminAPICustom Info	A collection of values returned for any defined custom fields in use.	Connected: AdminAPICustomInfo, on page 97	
AdminAPIUserInfo	A collection of personal information values for the account holder.	Connected: AdminAPIUserInfo, on page 98	
AdminAPIAccountSize	A collection of account size and usage statistics.	Connected: AdminAPIAccountSize, on the next page	

AdminAPIAccountInfoEx

This structure that contains information about the account including its start date and time, its Agent install path, Agent version, user information, account size and custom fields.

Name	Description	Type	Max String Size
AdminAPIBase AccountInfo	A collection of values common to all accounts.	Connected: AdminAPIBaseAccountInfo,	

Name	Description	Type	Max String Size
		on page 95	
StartDateTime	The date the account was registered. This value is empty if the account status is "Reserved".	xsd:dateTime	
AgentInstallPath	The installation path of the Agent; where it was installed. This value is empty if the account status is "Reserved".	xsd:string	255
AgentVersion	The Agent version (typically software and/or language version) assigned to the account. This value is empty if the account status is "Reserved".	xsd:string	
ComputerName	The name of the computer associated with the account. This value is empty if the account status is "Reserved".	xsd:string	255
AdminAPICustom Info	A collection of values returned for any defined custom fields in use.	Connected: AdminAPICustomInfo, on page 97	
AdminAPIUserInfo	A collection of personal information values for the account holder.	Connected: AdminAPIUserInfo, on page 98	
AdminAPIAccountSize	A collection of account size and usage statistics.	Connected: AdminAPIAccountSize, below	

AdminAPIAccountSize

Name	Description	Type
SnapShotDate	Date the backed up file was recorded in the database. Since the type of this field is xsd:date, it cannot be empty. However it can contain an invalid date, which is 0001-01-01.	xsd:date
NumArchives	Number of archives created for this account.	xsd:int

Name	Description	Type
NumFilesUnique	Number of unique files backed up.	xsd:int
SizeUnique	Total file size of all unique files.	xsd:long
SizeUniqueUncompressed	Total uncompressed file size of all unique files.	xsd:long
SizeUniqueDelta	Uncompressed, post-delta size of all non-pool file revisions. This is the size of the data before it is compressed.	xsd:long
NumFilesPool	Number of backed up files that are in the SendOnce pool of shared files.	xsd:int
SizePool	Total size of backed up files located in the SendOnce pool.	xsd:long
SizePoolUncompressed	Total uncompressed size of backed up files located in the SendOnce pool.	xsd:long
TipRevisionNumFiles	Number of changed files backed up during the last backup session.	xsd:int
TipRevisionUncompressed	Total uncompressed size of changed files backed up during the last backup session.	xsd:long
IsFirstBackup	If True, signifies that the number and size of files backed up reflects data for the account's first backup. If false, signifies that the number and size of files backed up reflects a normal backup. Typically, the number and total size of files backed up during a first backup can be significantly larger than a normal backup.	xsd:Boolean

AdminAPIAccountBackupDateInfo

Name	Description	Type
BackupDate	A date of a backup.	xsd:date

Status	The current status of the account:
	`NOSTATUS`
	`RESERVED`
	`ACTIVE`
	`ONHOLD`
	`CANCELLED`
		Connected: ACCOUNT_STATUS

Name	Description	Type
Compacted	If `True`, indicates the backup was compacted. If `False`, indicates the backup was not compacted.	xsd:Boolean
MediaSizeInBytes	The size of the backup in bytes.	xsd:Long

AdminAPIBaseAccountInfo

Name	Description	Type
AccountNumber	The number of the account	xsd:int
CommunityID	The ID of the community to which the Account is registered	xsd:int

Status	The current status of the account:
	`NOSTATUS`
	`RESERVED`
	`ACTIVE`
	`ONHOLD`
	`CANCELLED`
		Connected: ACCOUNT_ STATUS

AgentSetupID	The ID of the Agent Setup assigned to this account	xsd:int

AdminAPICommunityStatisticsInfo

Name	Description	Type
CommunityName	The name of the community	xsd:string
PCAccountCount	The number of PC accounts in this community	xsd:int
SVAccountCount	The number of Server accounts in this community	xsd:int
PCLicenseCountInUse	The number of PC licenses in use in this community	xsd:int
SVLicenseCountInUse	The number of Server licenses in use in this community	xsd:int
PCLicenseCountAvailable	The number of PC licenses available in this community (that is, the total number of PC licenses allocated to the community minus the number of PC licenses in use by	xsd:int

Name	Description	Type
	this community and its subcommunities)	
SVLicenseCountAvailable	The number of Server licenses available in this community (that is, the total number of Server licenses allocated to the community minus the number of Server licenses in use by this community and its subcommunities)	xsd:int
PCTipRevisionUncompressedSize	The relative size of the data for PC accounts in this community	xsd:long
SVTipRevisionUncompressedSize	The relative size of the data for Server accounts in this community	xsd:long

AdminAPICreditCard

Name	Description	Type	Max String Size

Type	The credit card type; enumerated value:
	`CARD_AMEX`
	`CARD_DISCOVER`
	`CARD_VISA`
	`CARD_MASTERCARD`
	`CARD_OTHER`
		Connected:AdminAPICreditCard	

Number	The credit card number that is billed for the specified account.	xsd:string	16
ExpiryDate	The credit card expiration date	xsd:string	16

Initialize `AdminAPICreditCard` before calling the `CommunityReserveTicket` method.

For example:

```
AdminAPIUserInfo oUserInfo = new AdminAPIUserInfo();
oUserInfo.CreditCardInfo = new AdminAPICreditCard();
oUserInfo.strLoginID = ‘ABC123’;

CommunityReserveTicket(3,0,oUserInfo, PRODUCTCODE_PC_AGENT);
```

AdminAPICustomInfo

Name	Description	Type	Max String Size

Section	One of the three custom fields available in the Agent, represented as Enumerated value:
	`CUSTOM1`
	`CUSTOM2`
	`CUSTOM3`
		Connected: CUSTOMFIELD	

Attribute	The name of the custom field. For this call this value is already set in the agent options and will be ignored	xsd:string	32
Value	The value of the custom field (of the Attribute above)	xsd:string	255

AdminAPIExtendedAccountInfo

Name	Description	Type
CancelDate	Date account was canceled.	xsd:dateTime
DeleteDate	Date account was deleted	xsd:dateTime
MsgCode	Message selected by technician was canceling account or putting it on hold	xsd:int
BillingMethod	Account's billing method	xsd:int
ProfileInfo	A data structure that contains account profile information	Connected: AdminAPIProfileInfo, on the next page

AdminAPIMediaCount

Name	Description	Type

eType	Enumerated value. One of the following:
	`MEDIA_CD`
	`MEDIA_DVD`
	`MEDIA_NAS`
		Connected:MediaType

nCount	The number of media units required to complete a media order.	xsd:int

AdminAPIProfileInfo

Name	Description	Type
Section	Section of account profile.	xsd:string
Attribute	Account profile attribute name.	xsd:string
Value	Account profile attribute value.	xsd:string

AdminAPIReportTemplateID

Name	Description	Type	Max String Size
CommunityID	The ID of the community where a report was created	xsd:int	
Name	The name of the report output	xsd:string	64

AdminAPITechnicianID

Name	Description	Type	Max String Size
CommunityID	Community where the technician was created (also known as the technician's root community).	xsd:int	
TechName	Technician's login name	xsd:string	64

AdminAPIUserInfo

Name	Description	Type	Max String Size
LoginID	The account holder's login ID	xsd:string	64
FirstName	First name of the account holder	xsd:string	32
MiddleName	Middle name or initial of the account holder	xsd:string	16

Name	Description	Type	Max String Size
LastName	Last name of the account holder	xsd:string	64
Telephone	Given telephone number for the account holder	xsd:string	32
Company	Account holder's employer or place of business	xsd:string	64
Address1	Account holder's given street address	xsd:string	40
Address2	Account holder's given street address, suite, apartment or PO box number	xsd:string	40
City	Account holder's city	xsd:string	32
State	Account holder's state	xsd:string	20
Zip	Account holder's zip	xsd:string	11
Email	Account holder's e-mail address	xsd:string	100
Country	Account holder's country	xsd:string	32
Department	Account holder's department or group	xsd:string	64
AdminAPICreditCard, on page 96	A collection of credit card fields including Type, Number and Expiration date. For more details, see AccountSetUserInfo, on page 26.	Connected: AdminAPICreditCard	

Chapter 5: Reference

This chapter describes the terms used in this document, as well as the error messages used by the Web Services API.

Terminology

Term	Description
account	An individual Agent subscriber. Accounts are identified by a unique 9-digit account number. An account is established at the Data Center when the Agent registers with the Data Center; an account is a prerequisite to first backup.
Agent	The client program installed on a computer that assembles the backup and sends it to the Data Center.
Agent configuration	The Agent version, rule set, Agent settings and Website settings that you can enable when creating Agent Setups.
AgentInfoURL	A URL created within the Agent that links to an informational Web site or portal. This is an optional feature for legacy PC Agents. This feature is configured in the Agent Configuration using the Agent Configuration Editor (ACE) through Support Center. The AccountVerifyAgentInfoURLHash API enables verification of requests coming into a Website by determining if the hash in the URL originated from valid Agent and the computer on which it is registered.
Agent Setup	The program that installs the Agent. Agent Setups are created using Support Center or theAccount Management Website. Each Agent Setup has a unique ID. The AccountSetAgentSetupID API enables you to change the Agent Setup assigned for a specified account ID. The CommunityGetInstall API enables you to download an Agent Setup.
Agent version	The language or software version of an Agent.
community	The basic organizational unit for accounts on the Data Center. A community is a group of accounts that can be managed collectively. When the Data Center is installed, one “default” community is created to receive new accounts. The CommunityCreate API enables the creation of new subcommunities within a given community.

Term	Description
canonical name	The form of a full community name that includes the path, starting from technician root community. The path is displayed in this format: `TechnicianRootCommunityName>SubcommunityName>Subcommunity`
encryption key	A key is a variable value used in the encryption and decryption of data. Every backup account has one encryption key. The encryption key is a series of letters and numbers, either randomly chosen or selected by the user. The encryption key is used to automatically encrypt and decrypt data on the user's computer. Once established, the encryption key for an account cannot be changed for the lifetime of the account. All data is encrypted on the user's computer by the Agent before it is transmitted to the Data Center. Starting in version 8.0, encryption keys are generated automatically and are not part of the user interface.
iRoam / MyRoam	An optional feature that provides secure access to backed up data via a Web interface. iRoam is accessible using any Web browser. Access to iRoam may be managed using the AccountDisableiRoam and AccountEnableiRoam APIs. Starting in version 8.0, iRoam is renamed MyRoam and can be accessed from the Account Management Website.
registration	Process by which an account is established at the Data Center during Agent installation on a computer.
Support Center	A Web-based application that allows users of version 6.1 and later to manage communities and accounts, create and deploy Agents, and run reports to monitor backup activity.
technician	Someone who has permission to access Support Center to manage, monitor and report on accounts. Technicians have one or more permissions that allow them to perform various administrative tasks using Support Center, the Web Interface Service, the Account Management Website, and the Agent user interface (Retrieve).
ticket	A uniquely-generated ID number that, when employed with an Agent configured to use reserved accounts, ensures that the Agent Setup will only establish one account, for the owner of the ticket. The API enables you to reserve accounts for a specific community.

Common error messages

The following is a list of error messages in common use by one or more APIs.

Code	Reason
1000	Failed execution due to database time-out issue.
1001	Access denied. Logged-in Technician does not have permission ‘Scripting’.
1002	Access denied. Logged-in Technician does not have permission ‘Modify Technician Permissions’.
1003	Access denied. Logged-in Technician does not have permission ‘Modify Communities’.
1004	Access denied. Logged-in Technician does not have permission ‘Run Reports’.
1012	Credit Card Expiration Date is not a valid date.
1013	Credit Card type is invalid. Valid credit card types are Visa, MasterCard and AMEX.
1014	Access denied. Logged-in Technician is not authorized to access resources.
1015	The community does not exist.
1016	The specified account cannot be found on the system.
1018	Credit Card number is invalid.
1019	Report does not exist in requested community.
1020	The community name cannot be blank.
1021	A community with the specified name already exists.
1022	The password provided does not conform to requirements. Account passwords must be at least 6 characters long, cannot have leading and trailing space and cannot contain all the same characters.
1023	Justification cannot be blank.
1024	Unable to perform required action. The destination community does not have enough licenses available.
1025	The Data Center is not licensed to use this feature.
1026	Community this account belongs to does not contain the Agent Setup specified.
1027	Unable to perform required action. A technician cannot modify him/herself.
1028	This account has been locked due to too many unsuccessful login attempts.
1030	The Data Center is not licensed for this product.
1056	The call is not allowed for specified account since its agent version does not support this feature.
1059	Specified date range is invalid: start date is after the end date.

Code	Reason
1069	The specified date is not a valid date.
1075	Invalid date range.
1076	Invalid server group; the parent community does not exist in the specified server group.
1077	Credit Cards are not supported for this account.
1078	Community does not contain the Agent Setup specified.
1079	Access denied. Logged-in Technician does not have permission ‘Move Accounts’.

Index

T

U

V

W

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Web Services Programming Reference (Micro Focus Connected Backup 8.11)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.ConnectedBackup.DocFeedback@microfocus.com.

We appreciate your feedback!