Personalization Server API

Version 2019.1

Contents

Introduction

The Personalization API is primarily used by the Environment Manager console to maintain and monitor the personalization database. It can also be used by customer-written .net applications or powershell scripts to provide additional functionality as documented here.

For Environment Manager version 10.0, significant database changes mean that existing scripts using earlier versions of the supplied proxy dlls (PSProxy3.dll and PSProxy4.dll) are not guaranteed to work. These changes are described in the next sections.

Breaking Changes in 10.1 FR2

This version introduces the GeoSync feature, where selected data from a publisher server can be regularly synchronized with one or more subscriber servers. This feature is disabled by default and must be configured by dedicated powershell cmdlets documented elsewhere. GeoSync configuration performed by the EM console can be programmed by this API, and the API calls are documented here.

One new feature which changes existing calls is the capability for subscriber servers to have user groups, windows settings groups and application groups with the duplicate names, where one name comes from the publisher and the other is local to the subscriber. Technically the key to each of these tables is now a composite of the name and the source system id - an integer that distinguishes between a published object and a local object.

As a result many functions searching by name now need a source system id parameter to resolve any ambiguity on a subscriber. For normal user (with no GeoSync) a source system id of null or zero means ‘local system’.

Until existing scripts are upgraded they may continue to work by using the proxy dll from the previous version of the product. Affected existing API calls are marked *(10.1 FR2)* in this document.

Breaking Changes in 10.0

The following changes to the 10.0 database have been made:

1. User profiles are no longer owned by both the personalization group (user group) and the user - a profile is owned only by the user. This means that a user can no longer have separate data for the same application in different user groups. Instead the same data follows the user if his group membership changes. If on upgrade duplicate data for the same user is found, the oldest data is deleted.
2. Only application groups may be assigned to user groups - previously an application could be directly assigned to a user group. On upgrade any directly-assigned applications are converted to application groups consisting of a single application.
3. Discovered profiles have been removed - similar functionality is provided by the data collection feature.

Changes from previous releases are marked with *(10.0)*

Proxy DLL

Communication with the Personalization Server is performed via a proxy dll which contains the class **ProfileManagementClient**. Two versions are provided - PSProxy3.dll (which uses .Net framework 3.5) and

PSProxy4.dll (for .Net framework 4). The legacy dll (ProfileManagement.dll) is still included in the installation for legacy code. Although the API can be used from any .NET language, PowerShell is discussed below.

Getting Started (Powershell)

To use the API the appropriate proxy dll must be loaded into the PowerShell environment. There are a number of ways to do this:

- The Server Configuration Utility PowerShell cmdlet **Import-ApsInstanceModule** automatically imports both the import and export PowerShell cmdlets and PSProxy4.dll into the PowerShell environment. Import-ApsInstanceModule is installed to the default location when the personalization server is installed and is thus immediately available. This requires PowerShell 3 or later to be installed.
- The import/export PowerShell modules may be loaded explicitly with **Import-Module**. This automatically loads PSProxy4.dll. A typical command would be;

  ```
  import-module "c:\program files\AppSense\Environment Manager\Personalization Server\API\Modules\EMPImportExport"
  ```

  (This also requires PowerShell 3 or later)
- The appropriate module can be loaded directly with **Import-Module**. Use PSProxy3.dll for PowerShell 2, and PSProxy4.dll for PowerShell 3 or later. The can also be done from a different machine by simply copying the dll to that machine.

Getting Started (.Net)

From a .net language, add a reference to PSProxy3.dll or PSProxy4.dll according to which version of the net framework is being used by the application.

Using the Proxy DLL

Once the proxy dll is loaded (see 'Getting Started' above), one of the connect functions is called to return a proxy object (of type **ProfileManagementClient** in the global namespace). All the API calls documented here are methods on this proxy object.

The proxy contains three static connect functions:

- **PSProxy.ConnectAnonymous** - connects to a PS (personalization server) using anonymous authentication
- **PSProxy.ConnectWindows** - connects to a PS using windows authentication
- **PSProxy.Connect** - connects to either (by trying windows first then anonymous)

Each function takes four parameters:

Parameter	Details
Server	String. Name of server to connect to. If a non-standard port is used, append it to the server name preceded by a colon (e.g. PServer:8000), Required.
Https	Boolean. True if ssl is to be used (optional, default false)
Username	String. Username used to connect to server (optional - default is to use current logged-in user). Specify as domain\username.
Password	String. Password (specify if username is used).

In PowerShell:

```
PS> import-module "<path to psproxy dll>"
```

```
PS> $pmClient = [PSProxy]::Connect(<server>,[<https>[,<username>, <password>]])
```

The Connect function returns a **ProfileManagementClient** object in $pmClient. The API functions are methods on this object.

Note

When creating multiple client objects it is important to close them after use, by calling the Close() function thus:

```
PS> $pmClient.Close()
```

Setting $pmClient to $null or overwriting it with another object will not close the connection - it will remain open in the background.

Authorization and Security

The API is in the same security context as the EM Console - the user specified in the connect call (or the current user if username and password are omitted) must be an authorized user of the console. Users only authorized for the web console cannot use the API. Note that regardless of the server authentication type, users must be valid Windows users.

Limitations

This API was designed to be used by the Environment Manager Console to monitor and modify the database. Because of this certain methods are not documented for the following reasons:

- They are not used by the console and are thus untested
- They return data which is of no use to a third party
- They require complex validation which must be done by the console software
- There are some “get” routines which modify the database and can’t be used by third party software

Nomenclature

Note that the internal name for “Personalization Analysis” is “Profile Analysis”, and the internal name for a “Personalization Group” is a “User Group”. Methods and fields which are new for 10.0 are marked “*(10.0)”*.

Well-known GUIDs

Names for Special Applications

Methods requiring application names (in the ProfileAnalysis section) require string GUIDs to represent special applications. Names for these applications can potentially be localized, so the API returns them as GUIDs and expects them as GUIDS. GUIDs supplied to API functions must have enclosing braces. Special application names are:

Windows Settings (formerly DesktopSettings)	{5DC1123B-3983-451C-A4CF-6CF169B639EB}
Session Data (OBSOLETE)	{A4977E45-9CF6-43F9-8DCD-111915E9A00A}
Certificates (OBSOLETE)	{D3AC9732-C951-4AF6-8799-83C4C5CBD6A4}

Other Well-Known GUIDs

Default Users (user group)	{00000000-0000-0000-1111-000000000000}
Default Site	{00000000-0000-0000-3333-000000000000}
Parent of global paths/GlobalID	{10732D21-EA06-4588-8144-D79EBD0EB2FF}
Parent of data collection paths	{10160847-0383-4D8E-911B-51DBD1BA7CB6}

API Classes

This section provides an alphabetical list of all classes used by the API.

Application

Remarks

Represents an application defined in the Application table.

profilemanagement.profileservice.datacontracts.Application

ApplicationId	Guid	Guid of application
Name	string	Application Name
Executable	string	Executable of application
VersionRegEx	string	Regular expression matching executable version
OSRegEx	string	Regular expression matching operating system version
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
Registry	Registry	Registry object - lists of application-specific included and excluded registry keys
Folder	Folder	Folder objects - lists of application-specific included and excluded folder paths
File	FilePaths	FilePaths objects - list of application-specific included and excluded file paths
OriginalName	string	Original name of application at creation time used by clients

(10.0) Application Category removed

ApplicationData

Remarks

Represents a row of the ApplicationData table. This row represents either a folder or a file in a user profile. For a file the record contains the actual file data.

profilemanagement.profileservice.datacontracts.ApplicationData

ApplicationProfilePK	Guid	Guid of owning application profile
ApplicationDataPK	int	Hash of folder and filename. This forms part of the primary key of this table (with ApplicationProfilePK)
Filename	string	Name of file, or "." if the row represents a folder
RelativePath	string	Path to file, with leading CSIDL value denoting the folder (e.g. {CSIDL_APPDATA}\Microsoft\Office). Can contain the reserved strings "Device" or "Registry" for the rename table and registry settings file (fbr) respectively.
StoredSize	int	Size of data in database (zero for folders) (bytes)
Size	int	Uncompressed size of file (zero for folders) (bytes)
Attributes	int	Win32 attribute flags
ACLDetails	string	Not used
ModifiedTime	DateTime?	Modified time from file when saved
InsertionTime	DateTime?	Time row was inserted into database
Data	byte[]	Data from database (null for folder). May be gzipped (compressed) for larger files

ApplicationGroupAssignment

Remarks

Represents a link between an application and an application group - the application is a member of the application group.

profilemanagement.profileservice.datacontracts.ApplicationGroupAssignment

ApplicationId	Guid	Guid of application
ApplicationGroupId	Guid	Guid of application group
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user

ApplicationGroupManagedFolder

Remarks

Represents a managed folder assigned to an application group.

profilemanagement.profileservice.datacontracts.ApplicationGroupManagedFolder

ApplicationGroupId	Guid	Guid of application group
Folder	string	Folder path
OSRegEx	string	Regular expression matching operating system version
Recursive	Boolean	Indicates whether subdirectories of the folder are to be managed

ApplicationPassiveTrace

Remarks

Represents a summary of the collected trace information for an application

profilemanagement.profileservice.datacontracts.ApplicationPassiveTrace

TraceApplicationID	int	Id of application
ApplicationPath	string	Path to application
Executable	string	Name of application
Version	string	Application version (from application's resources) - major version only (e.g. 6.*)
Company	string	Company (from resources)
Description	string	Description (from resources)
Accesses	ApplicationPassiveTraceAccess[]	not used
AccessCount	Int64	Total accesses since data collection started

ApplicationPassiveTraceAccess

Remarks

Represents summary of access for a particular folder or registry key (associated with an application)

profilemanagement.profileservice.datacontracts.ApplicationPassiveTraceAccess

Path	string	Path accessed (registry key or folder)
IsRegistry	Boolean	Path is key
IsWrite	Boolean	A write access has occurred
Size	Int64	Peak size of data in folder/key (across all clients) (bytes)
AccessCount	Int64	Total accesses (all clients)
AverageSize	Int64	Average size of the data contained in the key or folder (bytes)

Argument

Remarks

Arguments and operator for an Expression.

profilemanagement.profileservice.datacontracts.Argument

ArgumentPK	Guid	Guid of argument
Name	string	Type of argument
Type	string	Data type of value
Value	string	Value for an operand
ExpressionPK	Guid	Parent expression guid
OrderNo	int	Argument sequence
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

AuthorizedUser

Remarks

Represents a user authorized to run the EM Console. (Unused fields not shown)

profilemanagement.profileservice.datacontracts.AuthorizedUser

UserId	Guid	User identity GUID
Name	string	User name (domain\name)
SID	string	SID of user/AD group
IsGroup	bool	True if name is an AD Group
Roles[]	AuthorizedUserRole[]	User role(s)
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

AuthorizedUserRole

Remarks

Role owned by an AuthorizedUser

profilemanagement.profileservice.datacontracts.AuthorizedUserRole

UserId	Guid	Ownin user id
Role	AuthorizedUserRoleEnum	Role: User (0), Administrator (1), WebUser (2), WebAdministrator (3), SelfService (4), SupportConsole (5), ApplicationTeam(6)

BlacklistedApplication

Remarks

Represents an application in the global Application Exclusions list.

profilemanagement.profileservice.datacontracts.BlacklistedApplication

ApplicationId	Guid	Guid of application
Name	string	Application Name
Executable	string	Executable of application
VersionRegEx	string	Regular expression matching executable version
OSRegEx	string	Regular expression matching operating system version
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
Action	IUDActionEnum	The action to perform on the application {0=None,1=Insert,2=Update,3=Delete}

ContactDatabase

Remarks

Return from ContactDatabase call

profilemanagement.profileservice.datacontracts.ContactDatabase

Response	int	1 = OK 6 = version mismatch
ExpectedVersion	string	DALSchema version of PS

DataCheck

(10.0) Removed

Expression

Remarks

Expression associates with Membership's ExpressionList member. Represents a condition in a membership. (Unused members not shown.)

profilemanagement.profileservice.datacontracts.Expression

ExpressionPK	Guid	Guid of expression
Name	string	Text representation of condition
Plugin	string	Type of condition (plugin name)
MembershipFK	Guid	Associated membership
Arguments	Argument[]	Argument list
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

FilePaths

Remarks

Represents a list of file inclusions and exclusions associated with an application or application group

profilemanagement.profileservice.datacontracts.FilePaths

Exclude	Path[]	Excluded file paths
Include	Path[]	Included file paths

Folder

Remarks

Represents a list of folder inclusions and exclusions associated with an application or application group

profilemanagement.profileservice.datacontracts.Folder

Exclude	Path[]	Excluded folder paths
Include	Path[]	Included folder paths

FolderSetting

Remarks

Used only by OperatingSystemSetting object to represent managed folders/files in a custom setting

profilemanagement.profileservice.datacontracts.FolderSetting

Folder	string	Folder name
FileName	string	File name if row refers to a file
Exclude	bool	*(10.0)* Folder is excluded from custom setting

ForceReplicationNow

Remarks

Object returned from ForceReplicationNow method

profilemanagement.profileservice.datacontracts.ForceReplicationNow

Response	int	1 = OK, 2 = Replication not installed, 3 = Cannot start replication from subscriber

GeoCommand

Remarks

Command sent to initiate a GeoSync operation. Commands are

```
public enum GeoSyncCommand
{
    StartBatchSync = 0,     // Start one-off sync
    StartConfigOnlySync,    // Start one-off configuration sync
    StartContinuousSync,    // NOT IMPLEMENTED
    CancelSync,             // Cancel current sync
    Disconnect,             // NOT IMPLEMENTED
    None,                   // No command available
    Reset                   // Reset (aka uninitialize)
}
```

profilemanagement.profileservice.datacontracts.GeoCommand

Command	Constants.GeoSyncCommand	Supported commands in this release are StartBatchSync, StartConfigOnlySync and CancelSync
ServiceName	string	Name of service. This refers to the 'ServiceName' in the Geo.RemoteService table (not the display name)

GeoCommandResponse

Remarks

Response to GeoSync command, Status enum as follows:

```
public enum GeoSyncCommandStatus
{
    Accepted = 0,       // Command accepted and started
    Completed,          // Command successfully completed synchronously
    Busy,               // Service busy with previous command
    ServerException,    // Server exception
    InvalidCommand,     // Something wrong with command
    UnknownCommand,     // Never heard of command
    UnknownResponse,    // PS didn't understand BS response
    NotResponding,      // Background service not responding
    SyncNotActive      // Cancel when sync not active,

}
```

profilemanagement.profileservice.datacontracts.GeoCommandResponse

Status	Constants.GeoSyncCommandStatus	Enum, status 0 = OK
AdditionalInfo	string	Additional response info (English)

GeoRemoteService

Remarks

Contains information on the publisher about a remote service. This consists of fixed fields and statistics about the current or last synchronization.

profilemanagement.profileservice.datacontracts.GeoRemoteService

DisplayName	string	Display name of service
ServiceName	string	Service Name, used to identify service in other calls
Server	string	Remote SQL server name
Instance	string	Remote SQL Instance
Database	string	Remote database
ServiceState	Constants.GeoRemoteServiceState	Service State enum
LastSyncTime	DateTime?	Time last sync completed (null if none)
SyncStartTime	DateTime?	Time sync started
UserToSynchronize	int?	Total users to be synchronized
UsersSynchronized	int?	Users synchronized so fare
ProfilesSynchronized	int?	Profile synchronized
UserRetries	int?	Number of retries to sync user record
ProfileRetries	int?	Number of retries to sync profile
UsersFailed	int?	Users failed after retry
ProfilesFailed	int?	Profiles failed after retry
UsersToInitialize	int?	User to be initailized
UsersInitialized	int?	Users initialized
UserGroups	GeoUserGroup[]	List of groups with users to be sent to this service
LastError	string	Last error as string resource name (used by console to look up language string)
LastErrorParameters	string[]	Parameters to be inserted in error (if any)

LastErrorTime	DateTime?	Time of Last Error

GeoRemoteServiceList

Remarks

List of remote services.

profilemanagement.profileservice.datacontracts.GeoRemoteServiceList

GeoSyncEnabled	bool	True if GeoSync enabled on database
Services	GeoRemoteService[]	Array of remote services

GeoSyncSchedule

Remarks

Schedule information for a remote service

profilemanagement.profileservice.datacontracts. GeoSyncSchedule

ServiceName	string	Service Name (key)
DisplayName	string	Display name
ScheduledSyncEnabled	bool	Schedule is enabled
ScheduledSyncTime	TimeSpan	Time of day of sync
RepeatScheduledSync	bool	Repeating (one-off if false)
LastScheduledSyncTime	DateTime?	Last time it ran
NextScheduledSyncTime	DateTime?	Next due time if repeating schedule

GeoUserGroup

Remarks

User group for GeoSync functions

profilemanagement.profileservice.datacontracts. GeoSyncSchedule

UserGroupId	Guid	User group id
Name	string	Group name
ConfigurationOnly	bool	Sync is configuration-only (no user data)

Global

Remarks

Global folder and registry paths. Unused fields not shown

profilemanagement.profileservice.datacontracts.Global

RootFolder	string	Root for virtual caches on client machines
RootRegistry	string	not used
Registry	Registry	Global registry inclusions and exclusions
Folder	Folder	Global folder inclusions and exclusions
GlobalId	Guid	Guid that is parent of global folder inclusions and exclusions in the Path table ({10732d21-ea06-4588-8144-d79ebd0eb2ff})
CreationUser	string	Creating user name
CreationTIme	DateTime	Time the database was installed
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified
File	FilePaths	Global file inclusions and exclusions
PassiveId	Guid	Guid that is parent of folder and registry inclusions and exclusions for passive data collection in the Path table ({10160847-0383-4d8e-911b-51dbd1ba7cb6})
PassiveRegistry	Registry	List of registry inclusions/exclusions for data collection
PassiveFolder	Folder	List of folder inclusions/exclusions for data collection

GlobalProperty

Remarks

Represents a value from the GlobalProperty table.

profilemanagement.profileservice.datacontracts.GlobalProperty

Name	string	Name of property
Value	string	Value of property
Description	string	Text description for non-system settings. For system settings, name of a console resource providing the text description
IsSystemSetting	bool	True if factory value
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

GroupBasedDesktopSettingState *(OBSOLETE)*

Remarks

Relates to an individual desktop setting and its RoamPlan value. The RoamPlans supported are ‘Off’, ‘Separate’, ‘Shared’.

profilemanagement.profileservice.datacontracts.GroupBasedDesktopSettingState

UserGroupFK	Guid	Guid of personalization group
NewUserGroupFK	Guid	Used to set to a group or global
Setting	string	Desktop setting
Group	string	Desktop setting group
RoamPlan	sting	‘Off’, ‘Shared’, ‘Separate’
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

IdAndModifiedTimePair

Remarks

Encapsulates a combination of object id and its modified time for use when deleting a list of objects (applying concurrency checks)

profilemanagement.profileservice.datacontracts.IdAndModifiedTimePair

Id	Guid	Identity of object
ModifiedTime	DateTime?	Modified time associated with object

ManagedApplicationGroup

Remarks

Details of an application group.

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup

ApplicationGroupId	Guid	Guid of application group
Name	string	ApplicationGroup Name
Description	string	User description
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
ManagedGroupApplications	Application[]	Applications in the group
Registry	Registry	List of registry inclusions/exclusions
Folder	Folder	List of folder inclusions/exclusions
ManagedFolders	ApplicationGroupManagedFolder[]	List of managed folders
File	FilePaths	List of file inclusions/exclusions
OriginalName	string	Original name of application group at creation time used by clients
InstallerCondition	string	Regular expression used only by migration to check if the application group is installed on the endpoint. Can only be entered/modified by import

Membership

Remarks

Row of membership table. Memberships represent conditions for membership of a user group or site.

Note

Unused members are not listed.

profilemanagement.profileservice.datacontracts.Membership

MembershipPK	Guid	Guid of membership
ParentFK	Guid?	Guid of parent membership
ExpressionList	Expression[]	List of expressions
AssignmentFK	Guid?	Parent row (UserGroup or Site)
CreationUser	string	Creating user name
CreationTime	DataTime	Time created
ModifiedUser	string	Modifying user
ModifedTime	DateTime	Time modified

OperatingSystemSettingInformation

Remarks

Used by WindowsSettingsSetting object to represent managed folders and registry keys for an operating system

profilemanagement.profileservice.datacontracts. OperatingSystemSettingInformation

OperatingSystem	string	Name of OS
Folders	FolderSetting[]	Array of folder settings
Registry	RegistrySetting[]	Array of registry settings

Path

Remarks

Represents a registry or folder path from the Path table (contains inclusions/exclusions)

profilemanagement.profileservice.datacontracts.Path

Value	string	Path
Include	bool	true if an include, false if an exclude
Type	int	0=registry key, 1=folder, 2=file, 3=registry value
ParentId	Guid	Guid of parent record
PathId	Guid	Guid of this path
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user

ProfileAnalysisApplication

Remarks

Application group as returned by Profile Analysis methods

profilemanagement.profileservice.datacontracts.ProfileAnalysisApplication

Name	string	Name of application (string GUID for special applications)
ApplicationPK	Guid	Guid of application in appropriate table (or well-known guid)

(10.0) IsGroup removed - all applications are application groups here

ProfileAnalysisArchive

Remarks

Structures returned by FetchArchiveReport, representing archives stored for a user for a particular user.

profilemanagement.profileservice.datacontracts.ProfileAnalysisArchive

Application	string	**Name of application (string GUID for special applications)**
Archives	ProfileAnalysisArchiveDetail[]	Array of detail records
ApplicationType	enum	Type of application: 1 = ManagedApplicationGroup 5 = ManagedDesktop (i.e. WindowsSettings) 6 = ManagedCertificate (i.e. Legacy Certificates) 7 = ManagedSessionData (i.e. Legacy Session Data)
SourceSystemId	int?	Owner of archive. 0 or null = local system, >0 archive is for a profile on a subscriber that was synchronized from a publisher
DuplicateName	bool	True it the archive is of an application group which is duplicated in the profile (where one group is local and the other published) *(10.1 FR2)*

ProfileAnalysisArchiveDetail

Remarks

Represents a single archive for a user/application. Only one archive per user can be set as protected.

profilemanagement.profileservice.datacontracts.ProfileAnalysisArchiveDetail

ArchiveDate	DateTime	Time archive saved
SizeKB	float	Size of data in KB
ArchiveID	int	Archive identifier for restore/delete
IsProtected	Boolean	Protected against automatic deletion
Description	String	Description of archive.
ApplicationType	enum	(see above)

ProfileAnalysisPlotPoint

Remarks

Represents a point on a profile analysis graph

profilemanagement.profileservice.datacontracts.ProfileAnalysisPlotPoint

Name	string	Name of point. If the point represents a special application, a GUID is returned as a string
PlotValue	float	Value of point - units depend on graph type
Rank	int	Rank of point in graph. Note: Rank starts from 2 for the first point on the graph, as the PS uses Rank 1 internally to calculate the summary
SourceSystemId	int?	Owner of application group. 0 or null = local system, >0 application group on a subscriber that was synchronized from a publisher
DuplicateName	bool	True if application group is duplicated in the profile *(10.1 FR2)*

(10.0) DiscoveredApplication & DiscoveredHasData removed

ProfileAnalysisPlotSummary

Remarks

Represents the summary of a profile analysis graph

profilemanagement.profileservice.datacontracts.ProfileAnalysisPlotSummary

TotalValue	float	Total sum of all points on graph
IntervalSize	int	obsolete - not used
MaxRank	int	Total items in graph

ProfileAnalysisUser

Remarks

Represents a user name in profile analysis - this is simply the user name. Profile analysis (personalization analysis) always works with names rather than GUIDs, so that name changes at other consoles cause operations to fail (providing concurrency protection)

profilemanagement.profileservice.datacontracts.ProfileAnalysisUser

Name	string	User name (domain\loginname)

ProfileAnalysisWindowsSettingsGroupArchiveDetail

Remarks

Represents name, description and size of data in a windows settings group archive

profilemanagement.profileservice.datacontracts.ProfileAnalysisWindowsSettingsGroupArchiveDetail

Name	string	Original name of windows setting group in archive
Description	string	Current description of windows setting group
DisplayName	string	Current display name if group has been renamed, otherwise same as Name
SizeKB	float	Size of files in group in the archive

ProfileAnalysisWindowsSettingsGroupDetail

Remarks

Represents name, description and display name for data in a windows settings group profile

profilemanagement.profileservice.datacontracts.ProfileAnalysisWindowsSettingsGroupArchiveDetail

Name	string	Original name of windows setting group in profile
Description	string	Current description of windows setting group
DisplayName	string	Current display name if group has been renamed, otherwise same as Name

Registry

Remarks

Represents a list of registry key or value inclusions and exclusions associated with an application or application group

profilemanagement.profileservice.datacontracts.Registry

Exclude	Path[]	Excluded registry keys/values
Include	Path[]	Included registry keys/values

RegistrySetting

Remarks

Used only by OperatingSystemSetting object to represent managed keys/values in a custom setting

profilemanagement.profileservice.datacontracts.RegistrySetting

Key	string	Key name
Value	string	Value name if row refers to a value
Exclude	bool	Setting is an exclusion

Result

Remarks

General result object

profilemanagement.profileservice.datacontracts.Result

ErrorCode	int	Error code

Server

Remarks

Represents a row in the Server table.

profilemanagement.profileservice.datacontracts.Server

ServerId	Guid	Guid of row
NetbiosName	string	Netbios name of server
Url	string	Url of server sent to clients
Port	int	TCP port number sent to clients
Description	string	Text description
DistinguishedName	string	AD name of server
ADObjectId	Guid?	AD object guid
SiteId	Guid	Id of owning Site
FQDN	string	Fully qualified domain name
DNSSuffix	string	DNS suffix of FQDN
ServerType	ServerTypeEnum	**0 = physical server, 1 = virtual host. A virtual host represents a load-balanced Url**
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
OrderNo	int	Order precedence.(low number = higher priority)
VirtualHostRetries	int	Number of retries the server will perform on a failed virtual host before moving to the next one in the list.
VirtualHostTimeOut	int	Interval in milliseconds between retries the server will perform on a failed virtual host.

SessionData *(OBSOLETE)*

Remarks

Represents a registry key in the SessionData table. IUDActionEnum = {0=None,1=Insert,2=Update,3=Delete}

profilemanagement.profileservice.datacontracts.SessionData

SessionDataPK	Guid	Guid of row
UserGroupFK	Guid	UserGroup Guid or GlobalID
Path	string	Registry key
Type	SessionDataTypeEnum	Type of data (always Registry (0))
Action	IUDActionEnum	Type of action
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user

Site

Remarks

Represents a row in the Site table. An OrderNo of 2147483647 represents the default site.

profilemanagement.profileservice.datacontracts.Site

SiteID	Guid	Guid of row
Name	string	Site name
Description	string	Text description
OrderNo	int	**Order precedence (low number = higher priority)**
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
Servers	Server[]	Array of servers associated with Site

SystemInfo

Remarks

Contains the system information that is presented on the home screen

profilemanagement.profileservice.datacontracts.SystemInfo

Server	string	Database server name
SQLVersion	string	SQL Server version information
PersonalizationServerVersion	string	Version of the Personalization Server. May not match PersonalizationServerVersionExternal if the software has been patched.
DBName	string	Personalization database name
DBSize	double	Size of database in MB/GB
ManagedApplications	int	Number of applications defined in “User” category
ApplicationGroups	int	Total number of application groups defined
PersonalizationGroup	int	Total number of user groups defined (including default users)
Users	int	Number of endpoint users
Sites	int	Number of sites
PersonalizationServerVersionExternal	string	Product identifier of the server software
DBVersion	string	Database Schema Version, e.g. “8.6.20”
GeoSyncEnabled	bool	GeoSync feature enabled on database

(10.0) DiscoveredApplications removed

User

Remarks

Represents a user from the User table. Unused fields are omitted

profilemanagement.profileservice.datacontracts.User

UserId	Guid	Guid of user
Name	string	User login name (Domain\Name)
SID	string	User's SID
LastLogin	DateTime	Time of last login (server time)
CreationTime	DateTime	Time record created
ModifedTime	DateTime	Time modified

UserGroup

Remarks

UserGroup (internal name for Personalization Group) from the UserGroup table. An OrderNo of 2147483647 represents the default Personalization Group

profilemanagement.profileservice.datacontracts.UserGroup

GroupId	Guid	Guid of user group
Name	string	UserGroup Name
Description	string	User description
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user
OfflineMode	bool	Offline mode
OfflineResiliency	bool	Offline resilience
Certificate	bool	Manage certificates
WhitelistApplicationGroups	ManagedApplicationGroup[]	ApplcationGroups in whitelist
OrderNo	int	Priority (and display order on console), lower number higher priority
Desktop	bool?	Desktop settings managed *(OBSOLETE)*
SessionData	bool	Session data managed *(OBSOLETE)*

PassiveMode	bool	Trace data collected (for configuration assistant)
InheritSessionData	bool	Group inherits global session data *(OBSOLETE)*
OfflineRulesMode	bool	Offline rules mode
DynamicOffline		Dynamic offline mode
InheritSessionData	bool	Inherit global session data
PreCache	bool	Precache mode- endpoint loads all data at logon
SyncOnLogoff	bool	WindowsSettings sync performed on logoff
SyncOnSessionLock	bool	WindowsSettings sync performed on session lock
SyncOnSessionDisconnect	bool	WindowsSettings sync performed on session disconnect
WhitelistApplicationGroupsCount	int	Number of application groups assigned to user group
MembershipsCount	int	Number of membership rules assigned to user group
WindowsSettingsGroupCount	int	Count of windows settings groups assigned to user group
EndpointSelfServiceRulesMode	enum	Mod for Endpoint Self-Service tool: 0 = Disabled 1 = EnabledForGroup (unconditionally) 2 = EnabledWithGroupRules
MigrationEnabled	bool	Migration mode enabled
MigrationSequenceNumber	int?	Identity of current migration - incremented when a new migration is started. Null if no migration performed May still contain a value if migration is not enabled
MigrationType	int	Type of migration 1 = import 2 = export
MigrationForce	bool	Migration force mode (valid for import only)
MigrationToFrom	int	Migration source (import only): 0 = client profile 1 = MigrationPath

MigrationPath	string	source for migration data (network share)
MigrationAppDataPath	string	If non empty, separate path for AppData
GeoSyncEnabled	bool	GeoSync enabled on database (This does NOT imply anything about the specific user group - it's a global setting)

(10.0) Obsolete properties removed

UserGroupAssignedApplication

(10.0) Removed

UserGroupAssignedApplicationGroup

Remarks

Represents the association between a user group and an application group from the UserGroupAssignedApplicationGroup table.

profilemanagement.profileservice.datacontracts.UserGroupAssignedApplicationGroup

ApplicationGroupId	Guid	Guid of application group
UserGroupId	Guid	Guid of user group
Blacklist	bool	OBSOLETE - always returns false
CreationTime	DateTime	Time record created
CreationUser	string	Creating user name
ModifedTime	DateTime	Time modified
ModifiedUser	string	Modifying user

UserGroupGeoSyncList

Remarks

GeoSync configuration for a user group

profilemanagement.profileservice.datacontracts.UseGroupGeoSyncList

UserGroupId	Guid	Guid of user group
ModifiedTime	DateTime?	Time last modified
AvailableRemoteServices	UserGroupRemoteService[]	Remote services available to be assigned

GeoSyncPartitions	UserGroupGeoSyncPartition[]	Geo sync info partitions (service & conditions). Only one partition per user group is supported in this release

UserGroupGeoSyncPartition

Remarks

A partition contains a list of services and conditions assigned to a user group. In the current release only one partition is allowed per user group (for simplicity). Partition names are currently not visible in the user interface.

profilemanagement.profileservice.datacontracts.UseGroupGeoSyncPartition

Name	string	Name of partition
RemoteServices	UserGroupRemoteService[]	List of services
Conditions	Membership[]	List of memberships forming the conditions

UserGroupRemoteService

Remarks

Remote service associated with user group (returned in UserGroupGeoSyncPartition object)

profilemanagement.profileservice.datacontracts.UserGroupRemoteService

DisplayName	string	Service display name
ServiceName	string	Service key name
ServerName	string	Database server (inc instance)
Database	string	Database name
ConfigurationOnly	bool	Config-only sync

UserGroupsUsingWindowsSettingGroup

Remarks

List of user group names associated with a windows setting group

profilemanagement.profileservice.datacontracts.UserGroupsUsingWindowsSettingGroup

WindowsSettingsGroupId	Guid	Identity of windows settings group
UserGroupNames	string[]	List of user group names

WindowsSettingsCondition

Remarks

Represents a condition applied to a WindowsSettingsGroup

profilemanagement.profileservice.datacontracts.WindowsSettingsCondition

Id	Guid	Condition identity
Name	string	Condition name
Description	string	Text description
UEMConditions	string	Xml condition in UEM format
EMXConditions	string	Xml condition in EMX format
AvailableForReuse	bool	Condition is reusable
CreationUser	string	Creation user
CreationTime	DateTime	Creation time
ModifiedUser	string	Modified user
ModifiedTime	DateTime	Modified time
DeletedCondition	bool	Internal use only

WindowsSettingsConditionGroupAssociation

Remarks

Represents an association between a WindowsSettingsGroup and a WIndowsSettingsCondition

profilemanagement.profileservice.datacontracts. WindowsSettingsConditionGroupAssociation

ConditionID	Guid	Condition identity
GroupDisplayName	string	Display name of associated group
GroupID	Guid	Windows Settings Group identity
GroupModifiedTime	DateTime	Modified time of group

WindowsSettingsGroup

Remarks

Represents a Windows Settings Group

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup

Id	Guid	Group identity
DisplayName	string	Display name
OriginalName	string	Original name (used at client)
Description	string	Text descripition
CreationUser	string	Creation user
CreationTime	DateTime	Creation time
ModifiedUser	string	Modified user
ModifiedTime	DateTime	Modified time
Settings	WindowsSettingsGroupSetting[]	Array of associated settings (read-only)
Condition	WindowsSettingsCondition	Associated condition if any (read-only)
LegacyGroup	bool	Created by upgrade process

WindowsSettingsGroupSetting

Remarks

Represents a setting associated with a windows settings group.

profilemanagement.profileservice.datacontracts.WindowsSettingsGroupSetting

SettingId	Guid	Setting identity
SettingName	string	Setting name
SettingDescription	string	Text description
IsCustom	bool	True if setting is a user-entered custom setting
SettingData	string	Xml representation of OperatingSystemSettingInfomation for a custom setting (not used for fixed setting)
OperatingSystemSetting-Information	OperatingSystemSetting-Information[]	For a custom setting, lists per-OS folder/file/registrykey/registry value data
CreationTime	DateTime	Creation time
CreationUser	string	Creation user
ModifiedTime	DateTime	Modified time
ModifiedUser	string	Modified user
GroupsUsingSetting	string[]	Display names of wsgs using this setting (read-only)
IsCloneable	bool	True if the setting can be cloned

WindowsSettingsGroupsUsingSetting

Remarks

Represents a list of windows settings group names associated with a windows setting.

profilemanagement.profileservice.datacontracts.WindowsSettingsGroupSetting

WindowsSettingId	Guid	Setting identity
WindowsSettingsGroupNames	string[]	List of windows settings group names

API Descriptions

Application_ConvertDiscoveredApplication

(10.0) Removed

Application_ConvertDiscoveredApplication2

(10.0) Removed

Application_Delete

Remarks

Deletes an application by its Guid identifier. You must supply the modified time so that the PS can check that it hasn't changed since you read the application (optimistic concurrency). Throws any error.

Declaration

profilemanagement.profileservice.datacontracts.Result
Application_Delete(Guid ApplicationID, DateTime ModifiedTime)

Returns

Result - error code 0. Throws an exception on any error.

Caution

It is possible to delete any application with this function, even if it is assigned to an application group or a user group. The delete will automatically remove the associations from the database. Use Application_GetAllDependencies to check first.

Application_Get

Remarks

Returns details of an application if you know the Id. Joins to the path table to populate the Registry, Folder and File members of the returned Application object. Note that both registry keys and values are returned to the Registry property - they can be distinguished by the Type field in the Path object.

Declaration

profilemanagement.profileservice.datacontracts.Application Application_Get(System.Guid ApplicationId)

Returns

profilemanagement.profileservice.datacontracts.Application

Parameter

ApplicationId	Guid of application from Application Table

Application_GetAll

Remarks

Returns list of all applications in the Application table. **The registry, file and folder lists of the returned application objects are not populated with this method.**

Declaration

profilemanagement.profileservice.datacontracts.Application[] Application_GetAll()

Application_GetAllDependencies

Remarks

Returns lists of application groups (return value) and user groups (out parameter) which contain the specified application. Used to check it if it is safe to delete the application. **The lists of sub-objects in the returned objects are not populated with this method.** This applies to ManagedGroupApplications in ManagedApplicationGroup and Whitelists in UserGroup.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup[]
Application_GetAllDependencies(System.Guid ApplicationId,
out profilemanagement.profileservice.datacontracts.UserGroup[] UserGroups)

Returns

List of application groups containing this application

Parameters

ApplicationId	Guid of application from Application Table (in)
UserGroups	List of user groups referencing the application (out)

Application_GetAllDiscoveredApplications

(10.0) Removed

Application_GetAllUnassignedByApplicationGroup

Remarks

Returns list of applications NOT assigned to the specified application group.

Note

Registry, file and folder exclusions are not returned by this method.

Declaration

profilemanagement.profileservice.datacontracts.Application[]
Application_GetAllUnassignedByApplicationGroup(Guid ApplicationGroupId)

Returns

List of applications not assigned to Application Group ApplicationGroupId

Parameters

ApplicationGroupId	Guid of application group from ApplicationGroup table (in)

Application_GetAllUnassignedByUserGroup

(10.0) Removed

Application_GetByName

Remarks

Returns an application by name.

Note

Registry, file and folder exclusions are not returned by this method - to get these extract the guid and use Application_Get.

Declaration

profilemanagement.profileservice.datacontracts.Application Application_GetByName(string ApplicationName)

Returns

Application if found. Throws error if not

Parameters

Application name (not case-sensitive). This is the name of the application, not the name of the executable.

Application_GetDiscoveredApplication

(10.0) Removed

Application_Insert

Remarks

Inserts a new Application into the Application table, and returns the created details. The combination of executable, versionregex and osregex must be unique in the Application table.

To add application-specific folder and registry inclusions or exclusions, use the Path_Insert function.

Caution

The application inserted must have a name which does not match any other application, application group, discovered application or reserved application name. This is not checked by this function - it only checks that the name is not already an application.

Declaration

```
profilemanagement.profileservice.datacontracts.Application
                    Application_Insert (string Name,
                                        string Executable,
                                        string OSRegEx,
                                        string VersionRegEx)
```

Returns

Application record inserted, including modified time and new Guid. On error an exception is thrown.

Parameters

Name	Application name, as it appear on the console and in the client cache
Executable	Name of executable, including .exe
OSRegEx	Regular expression to match operating system version - version must match for application to be managed
VersionRegEx	Regular expression to match application version

Application_InsertWithOriginalName

Remarks

(Advanced use only). Similar to Application_Insert but attempts to set the OriginalName field. (Application_Insert sets the OriginalName to the same as the Name field, unless there is a name clash, where Application_Insert will generate a new OriginalName.) If the original name specified already exists on another application or application group, the name is modified with a suffix.

Declaration

profilemanagement.profileservice.datacontracts.Application
Application_InsertWithOriginalName (string Name,
string OriginalName,
string Executable,
string OSRegEx,
string VersionRegEx)

Returns

Application record inserted, including modified time and new Guid. On error an exception is thrown.

Parameters

Name	Application name, as it appear on the console
OriginalName	Original name of application as used by clients
Executable	Name of executable, including .exe
OSRegEx	Regular expression to match operating system version - version must match for application to be managed
VersionRegEx	Regular expression to match application version

Application_Update

Modifies an existing Application in the Application table, and returns the new details. The combination of executable, versionregex and osregex must be unique in the Application table.

Caution

If the application is renamed the new name must not match any other application, application group, discovered application or reserved application name. This is not checked by this function - it only checks that the new name is not already an application.

Declaration

profilemanagement.profileservice.datacontracts.Application
Application_Update (Guid ApplicationId,
string Name,
string Executable,
string OSRegEx,
string VersionRegEx,
DateTime ModifiedTime)

Returns

Application record modified, including new modified time. On error an exception it thrown. See warning above about namespaces

Parameters

ApplicationId	Guid of application to modify
Name	Application name
Executable	Name of executable, including .exe
OSRegEx	Regular expression to match operating system version - version must match for application to be managed
VersionRegEx	Regular expression to match application version
ModifiedTime	Current timestamp for concurrency checking. If the record's timestamp does not match, the update is not performed and an exception is thrown.

Application_UpdateWithOriginalName

(Advanced use only). Similar to Application_Update but sets the OriginalName field explicitly. (Application_Update doesn't change the OriginalName field.) If the original name specified already exists on another application or application group, the name is modified with a suffix.

Declaration

profilemanagement.profileservice.datacontracts.Application
Application_UpdateWithOriginalName
(Guid ApplicationId,
string Name,
string OriginalName,
string Executable,
string OSRegEx,
string VersionRegEx,
DateTime ModifiedTime)

Returns

Application record modified, including new modified time. On error an exception it thrown. See warning above about namespaces

Parameters

ApplicationId	Guid of application to modify
Name	Application name
OriginalName	Original application name used by clients
Executable	Name of executable, including .exe
OSRegEx	Regular expression to match operating system version - version must match for application to be managed
VersionRegEx	Regular expression to match application version
ModifiedTime	Current timestamp for concurrency checking. If the record's timestamp does not match, the update is not performed and an exception is thrown.

ApplicationBlacklist_Delete

Remarks

Deletes an application from the global application exclusions list.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationBlacklist_Delete(
Guid ApplicationId,
DateTime ModifiedTime)

Returns

Result - error code 0. Throws an exception on any error.

Parameters

ApplicationId	Guid of application to delete from list
ModifiedTime	Modified time - must match current modified time of row in table for concurrency check

ApplicationBlacklist_Get

Remarks

Return all applications from the application exclusions list

Declaration

profilemanagement.profileservice.datacontracts.BlacklistedApplication[]
ApplicationBlacklist_Get()

Returns

Array of BlaclistedApplication objects

ApplicationBlacklist_Update

Remarks

Updates the database from an array of BlacklistedApplication Data objects where the IUDAction member on each object is set to the relevant action (Insert, Update, and Delete).

Note that Update and Delete operations check for concurrency errors so must be called with the exact modified time to avoid exception.

Declaration

void ApplicationBlacklist_Update(
ref profilemanagement.profileservice.datacontracts.BlacklistedApplication[] ApplicationBlacklist)

Returns

Throws exception on error.

Parameters

ApplicationBlacklist	Array of BlacklistedApplication.

ApplicationGroup_Delete

Remarks

Deletes an application group by its Guid identifier. You must supply the modified time so that the PS can check that it hasn't changed since you read the application group (optimistic concurrency). Throws any error.

Caution

It is possible to delete any application group with this function, even if it is assigned to a user group. The delete will automatically remove the associations from the database. Use ApplicationGroup_GetAllDependencies to check first.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroup_Delete(Guid ApplicationGroupID, DateTime ModifiedTime)

Returns

Result - error code 0. Throws an exception on any error.

Parameters

ApplicationGroupId	GUID identifying the group.
ModifiedTime	Group's current timestamp. If this timestamp doesn't match the database, the delete is not performed and a concurrency exception is thrown.

ApplicationGroup_Get

Remarks

Retrieves an application group by its id (GUID). Populates the ManagedGroupApplications, Registry, Folder and File members of the returned ManagedApplicationGroup object. Note that both registry keys and values are returned to the Registry property - they can be distinguished by the Type field in the Path object.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
ApplicationGroup_Get(Guid ApplicationGroupID)

Returns

Application group object. Throws error if group doesn't exist

Parameters

ApplicationGroupId	GUID identifying the group.

ApplicationGroup_GetAll

Remarks

Retrieves all application groups. **The lists of sub-objects in the returned objects are not populated with this method.** This applies to ManagedGroupApplications and Registry and Folder lists.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup[]
ApplicationGroup_GetAll()

Returns

Array of ManagedApplicationGroup objects.

ApplicationGroup_GetAllDependencies

Remarks

Returns list of user groups which refer to contain the specified application group. Used to check it if it is safe to delete the application group. **The lists of sub-objects in the returned objects are not populated with this method.**

Declaration

profilemanagement.profileservice.datacontracts.UserGroup[]
ApplicationGroup_GetAllDependencies(System.Guid ApplicationGroupId)

Returns

List of user groups using this application group

Parameters

ApplicationGroupId - Guid of application

ApplicationGroup_GetByName

Remarks

Retrieves an application group by its name. Sub-objects (ManagedGroupApplications, Registry and Folder) are populated in this case.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
ApplicationGroup_GetByName(string ApplicationGroupName)

Returns

Application group if exists .

Parameters

ApplicationGroupName	Name of group (not case-sensitive)

ApplicationGroup_GetUnassignedApplicationGroupsByUserGroup

Remarks

Returns list of application groups NOT assigned to the specified user group.

> **Note**
>
> Registry, file and folder exclusions and application lists are not returned by this method.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup[]
ApplicationGroup_ GetUnassignedApplicationGroupsByUserGroup
(Guid UserGroupId)

Returns

List of application groups not assigned to user group UserGroupID

Parameters

User group id (Guid)

ApplicationGroup_Insert

Remarks

Inserts a new application group into the ApplicationGroup table, and returns the created details. To add application-specific folder and registry inclusions or exclusions, use the Path_Insert function. To add applications to the group, use the ApplicationGroupAssignment_Insert function.

Caution

The application group inserted must have a name which does not match any other application, application group, discovered application or reserved application name. This is not checked by this function - it only checks that the name is not already an application group.

Declaration

```
profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
                    ApplicationGroup_Insert (string Name,
                                             string Description)
```

Returns

ApplicationGroup record inserted, including modified time and new Guid. On error an exception is thrown. See warning above about naming.

Parameters

Name	Application group name, as it appear on the console and in the client cache
Description	Free text description

ApplicationGroup_InsertWithOriginalName

Remarks

(Advanced use only). Similar to ApplicationGroup_Insert but sets the OriginalName field. (ApplicationGroup_Insert sets the OriginalName to the same as the Name field, unless there is a name clash, where ApplicationGroup_Insert will generate a new OriginalName.) If the original name specified already exists on another application or application group, the name is modified with a suffix.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
ApplicationGroup_InsertWithOriginalName (string Name,
string OriginalName,
string Description)

Returns

ApplicationGroup record inserted, including modified time and new Guid. On error an exception is thrown. See warning above about naming.

Parameters

Name	Application group name, as it appear on the console
OriginalName	Original name of group used by clients
Description	Free text description

ApplicationGroup_Update

Modifies an existing application group in the ApplicationGroup table, and returns the new details.

Caution

If the application group is renamed the new name must not match any other application, application group, discovered application or reserved application name. This is not checked by this function - it only checks that the new name is not already an application group.

Declaration

profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
ApplicationGroup_Update (Guid ApplicationGroupId,
string Name,
string Description,
DateTime ModifiedTime)

Returns

Application group record as modified, including new modified time. On error an exception is thrown. See warning above about naming.

Parameters

ApplicationGroupId	Guid of application group to modify
Name	Application group name
Description	Free text description
ModifiedTime	Current timestamp for concurrency checking. If the record's timestamp does not match, the update is not performed and an exception is thrown.

ApplicationGroup_UpdateWithOriginalName

(Advanced use only). Similar to ApplicationGroup_Update but sets the OriginalName field explicitly. (ApplicationGroup_Update doesn't change the OriginalName field.) If the original name specified already exists on another application or application group, the name is modified with a suffix.

Declaration

```
profilemanagement.profileservice.datacontracts.ManagedApplicationGroup
                    ApplicationGroup_UpdateWithOriginalName (
                                        Guid ApplicationGroupId,
                                        string Name,
                                        string OriginalName,
                                        string Description,
                                        DateTime ModifiedTime)
```

Returns

Application group record as modified, including new modified time. On error an exception is thrown. See warning above about naming.

Parameters

ApplicationGroupId	Guid of application group to modify
Name	Application group name
OriginalName	Application group name used by clients
Description	Free text description
ModifiedTime	Current timestamp for concurrency checking. If the record's timestamp does not match, the update is not performed and an exception is thrown.

ApplicationGroupAssignment_Delete

Remarks

Deletes an ApplicationGroupAssignment - that is, removes an existing application from an application group (by specifying both Guids).

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroupAssignment_Delete(Guid ApplicationID,
Guid ApplicationGroupID)

Returns

Result - error code 0. Throws an exception on any error.

Parameters

ApplicationId	Guid of application to remove from group
ApplicationGroupId	Guid of group

ApplicationGroupAssignment_Insert

Remarks

Adds an ApplicationGroupAssignment - that is, adds an existing application to an application group.

Declaration

profilemanagement.profileservice.datacontracts.ApplicationGroupAssignment
ApplicationGroupAssignment_Insert(Guid ApplicationID,
Guid ApplicationGroupID)

Returns

Returns resultant assignment record

Parameters

ApplicationId	Guid of application to add to group
ApplicationGroupId	Guid of group

ApplicationGroupAssignment_ManagedFolderInsert

Remarks

Adds a managed folder to an existing application group.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroupAssignment_ManagedFolderInsert(Guid ApplicationGroupID,
String Folder,
String OSRegEx,
Boolean Recursive)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameters

ApplicationGroupId	Guid of application group to add the folder to
Folder	Folder path
OSRegEx	OS regular expression for new folder path
Recursive	Subfolders of the folder are to be managed

ApplicationGroupAssignment_ManagedFolderDelete

Remarks

Delete a managed folder from an existing application group.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroupAssignment_ManagedFolderDelete(Guid ApplicationGroupID,
String Folder,
String OSRegEx)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameters

ApplicationGroupId	Guid of application group to delete the folder from
Folder	Folder path
OSRegEx	OS regular expression for new folder path

ApplicationGroupAssignment_ManagedFolderUpdate

Remarks

Update the OSRegEx and/or Recursive values of a managed folder assigned to an existing application group.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroupAssignment_ManagedFolderUpdate
(Guid ApplicationGroupID,
String Folder,
String OSRegEx
Boolean Recursive)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameters

ApplicationGroupId	Guid of application group the managed folder is assigned to
Folder	Folder path
OSRegEx	OS regular expression for new folder path
Recursive	Subfolders of the folder are to be managed

ApplicationGroupAssignment_ManagedFolderRename

Remarks

Rename a managed folder assigned to an existing application group.

Declaration

profilemanagement.profileservice.datacontracts.Result
ApplicationGroupAssignment_ManagedFolderRename
(Guid ApplicationGroupID,
String oldFolder,
String Folder)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameters

ApplicationGroupId	Guid of application group the managed folder is assigned to
oldFolder	Folder path to be changed from
Folder	Folder path to be changed to

ApplicationPassiveTrace_GetAll

Remarks

Returns application trace data. This data is stored in the database when 'Enable Application Data Collection' is enabled for a UserGroup

Declaration

profilemanagement.profileservice.datacontracts.ApplicationPassiveTrace[]
ApplicationPassiveTrace_GetAll()

Returns

Array of ApplicationPassiveTrace objects. Each object summarizes the trace data for a particular application. Throws exception on error.

ApplicationPassiveTrace_GetAccesses

Remarks

Returns application trace data for a specific application, identified by its application id. This data is stored in the database when 'Enable Application Data Collection' is enabled for a UserGroup

Declaration

profilemanagement.profileservice.datacontracts.ApplicationPassiveTraceAccess[]
ApplicationPassiveTrace_GetAccesses(int ApplicationId)

Returns

Array of ApplicationPassiveTraceAccess objects. Each object contains the trace data for a folder path or registry key for the application. Throws exception on error

Parameters

ApplicationId	Integer id of application in trace data, as returned from ApplicationPassiveTrace_GetAll()

AuthorizedUser_Delete

Remarks

Deletes a user from the AuthorizedUser table. This table contains users authorized to run the Console. On delete any associated roles are also deleted.

Declaration

profilemanagement.profileservice.datacontracts.Result
AuthorizedUser_Delete(Guid AuthorizedUserID, DateTime ModifiedTime)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameter

AuthorizedUserID	Guid of row in AuthorizedUser table
ModifiedTime	Modified time of row for concurrency checking. If this doesn't match, no delete occurs and a concurrency exception is thrown

AuthorizedUser_DeleteRole

Remarks

Deletes a role associated with a user from the AuthorizedUserRole table. Users only belong to a single role in EM 8.2

Declaration

profilemanagement.profileservice.datacontracts.Result
AuthorizedUser_DeleteRole(Guid UserID,int RoleType)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameter

UserID	Guid of row in AuthorizedUser table for owning user
RoleType	Role to remove. 0 = user, 1 = administrator, 2 = webuser, 3 = webadministrator, 4 = SelfService

AuthorizedUser_GetAll

Remarks

Returns all authorized users

Declaration

profilemanagement.profileservice.datacontracts.AuthorizedUser[]
AuthorizedUser_GetAll()

Returns

List of AuthorizedUser records. Throws exception on error

AuthorizedUser_Insert

Remarks

Inserts a new authorized user. This could be a User or an AD Group. The function is only available to administrators. Note that after inserting an authorized user, access is only granted once a role is inserted.

Declaration

```
profilemanagement.profileservice.datacontracts.AuthorizedUser
        AuthorizedUser_Insert(string Name,
                              string Sid,
                              string DistinguishedName,
                              Guid? ADObjectId,
                              bool IsGroup)
```

Returns

Entered AuthorizedUser record for user. Throws exception on error.

Parameters

Name	Authorized user name. (domain\name). Can be AD group
SID	Authorized user SID
DistinguishedName	AD distinguished name of user. This value is not currently used for console login and may be passed as null
Guid?	ADObject id. AD Guid representing the user or group. Not currently used by console login - can be passed as null.
IsGroup	True if name is an AD group, False if a user.

AuthorizedUser_InsertRole

Remarks

Inserts a role for an authorized user. This function is only available to administrators.

Declaration

profilemanagement.profileservice.datacontracts.AuthorizedUserRole AuthorizedUser_InsertRole(Guid AuthorizedUserId, int Role)

Returns

Created AuthorizedUserRole record for user. Throws exception on error.

Parameters

AuthorizedUserId	Guid of authorized user
Role	Role to add: 0 = user, 1 = administrator, 2 = webuser, 3 = webadministrator

AuthorizedUser_Update

Remarks

Modifies details of an existing authorized user

Declaration

```
profilemanagement.profileservice.datacontracts.AuthorizedUser
        AuthorizedUser_Update(Guid AuthorizedUserId,
                              string Name,
                              string Sid,
                              string DistinguishedName,
                              Guid? ADObjectId,
                              bool IsGroup,
                              string ModifiedUser,
                              DateTime ModifiedTime)
```

Returns

Modified AuthorizedUserRole record for user. Throws exception on error.

Parameters

AuthorizedUserId	Guid of authorized user
Name	Authorized user name. (domain\name). Can be AD group
SID	Authorized user SID
DistinguishedName	AD distinguished name of user. This value is not currently used for console login and may be passed as null
Guid?	ADObject id. AD Guid representing the user or group. Not currently used by console login - can be passed as null.
IsGroup	True if name is an AD group, False if a user.
ModifiedUser	User name modifying the record
ModfiiedTime	ModifedTime of existing record for concurrency checking

AuthorizedUser_UpdateRole

Remarks

Change a user's role

Declaration

profilemanagement.profileservice.datacontracts. AuthorizedUserRole
AuthorizedUser_UpdateRole(Guid AuthorizedUserId,
int OldRole,

int NewRole)

Returns

Modified AuthorizedUserRole record for user. Throws exception on error.

Parameters

AuthorizedUserId	Guid of authorized user
OldRole	Old role to change
NewRole	New role (0 = user, 1 = administrator, 2 = webuser, 3 = webadministrator, 4 = SelfService)

AuthorizedUser_CurrentUserIsInRole

Remarks

Verifies the current user is a member of the supplied role.

Declaration

bool CurrentUserIsInRole(AuthorizedUserRoleEnum authorizedUserRole)

Returns

True if the current user is a member of the supplied role.

Parameters

Role	Role to verify

CleanClientApplicationCaches

Remarks

Forces cleanup of client caches (registry and file) to occur for all applications or application groups matching a wild carded pattern. This merely sets cleanup to be pending - the global setting ProfileCleanupDelayDays is still respected. (If this value is set to -1, cleanup is disabled and this function has no effect).

Declaration

profilemanagement.profileservice.datacontracts.Result CleanClientApplicationCaches(
string ApplicationOrApplicationGroupName)

Returns

Result.ErrorCode: -1 disabled >=0, number of applications matched by ApplicationOrApplicationGorupName

Parameters

ApplicationOrApplicationGroupName	Name of application or application group, with SQLServer wildcards % and _ if required.

ClearUserUsageCounts

Remarks

Clears all usage statistics for the specified user.

Declaration

profilemanagement.profileservice.datacontracts.Result
ClearUserUsageCounts(string UserName)

Returns

Result.ErrorCode: 0 = OK. Throws exception on error.

Parameters

UserName	User name (domain\user)

(10.0) Removed UserGroupName

ContactDatabase

Remarks

Checks database availability and returns current DALSchema - the version of the interface between the console and the personalization server. This document is based on DALSchema 8200.

Declaration

profilemanagement.profileservice.datacontracts.ContactDatabase
ContactDatabase(string ConsoleVersion)

Returns

ContactDatabase object. (See classes)

Parameters

string	ConsoleVersion (send - function checks match and returns its version)

DataCheck

Remarks

Checks to see if an application group has any user data.

Declaration

profilemanagement.profileservice.datacontracts.DataCheck
DataCheck(Guid ApplicationGroupID)

Returns

DataCheck object. Throws exception on error.

Parameters

ApplicationGroupID	Guid of ApplicationGroup to check for data

(10.0) Removed UserGroupId

DeleteAllUsersData

Remarks

Deletes all profiles for a single specified user. The data is not physically deleted until the next daily purge job occurs - it simply becomes inaccessible

Declaration

int DeleteAllUsersData(string UserName)

Returns

status: 0 OK, 1 User or UserGroup Not Found. Throws exception on error

Parameters

UserName	User name (domain\user)

(10.0) Removed UserGroupName

DeleteApplicationDataByName

Remarks

Deletes a single file from a user's profile (i.e row in ApplicationData)

Declaration

profilemanagement.profileservice.datacontracts.Result
DeleteApplicationDataByName(string UserName,
string ApplicationGroupName,
string RelativePath,
string Filename
int? SourceSystemId)

Returns

Result.Error code 0 = success, 1 = data not found. Throws exception on error

Parameters

UserName	User name (domain\user)
ApplicationGroupName	Name of application group. For reserved applications a string GUID is required
RelativePath	Path of data to delete
Filename	Filename

SourceSystemId	Specifies source system (where ApplicationGroupName is duplicated on subscriber). 0 or null = local group, >0 published group (currently 1) *(10.1 FR2)*

(10.0) Removed UserGroup

DeleteApplicationDataByPath

Remarks

Deletes all files in a particular profile (identified by Guid) in a specified folder, including any subfolders.

Declaration

profilemanagement.profileservice.datacontracts.Result
DeleteApplicationDataByPath(Guid ApplicationProfilePK, string Path)

Returns

Throws exception on error

Parameters

ApplicationProfilePK	Guid identifying the application profile
Path	Folder to be deleted in “raw” format - i. e. with CSIDL prefix

DeleteApplicationProfileByName

Remarks

Delete an application profile identified by UserGroup, UserName and ApplicationName. This function deletes all the data rows and updates the profile timestamp.

Caution

If the application is running for the user at the time of deletion, the existing data may be restored when the application is closed.

Declaration

int DeleteApplicationProfileByName(string UserGroupName,
string UserName,
string Application,
bool PreserveArchives,
bool? SetAsMigrated,
int? SourceSystemId)

Returns

Status: 0 = OK, 1 = profile doesn't exist. Throws exception on error

Parameters

UserGroupName	User group
UserName	User name
Application	Application Name (GUID for special apps)
PreserveArchives	true - save archives and read history
SetAsMigrated	If non-null and group is in a migration mode, controls whether the profile should be set as migrated after the deletion.
SourceSystemId	Source System id (0/null = local, else published) *(10.1 FR2)*

DeleteArchive

Remarks

Deletes an archive identified by its integer archive id. The archive id can be retrieved from the archive report (see FetchArchiveReport).

Declaration

int DeleteArchive(int ArchiveID)

Returns

Status: 0 = OK, 1 = Archive doesn't exist (may be concurrency failure, i.e. archive deleted by someone else). Other errors cause an exception to be thrown.

Parameters

ArchiveID	Integer archive identity from the relevant ProfileAnalysisArchiveDetail structure returned from a FetchArchiveReport. This identifies the archive globally (user name and application need not be specified)

DeleteWindowsSettingsGroupData

Remarks

Deletes all data for the specified windows settings group in the user's windows settings (formerly Desktop Settings) profile.

Declaration

int DeleteWindowsSettingsGroupData(
string UserName,
string WindowsSettingsGroupDisplayName,
bool? SetAsMigrated,
int? SourceSystemId)

Returns

Status: 0 OK, 1 Windows Settings Group Not Found. Throws exception on error

Parameters

UserName	User name (domain\user)
WindowsSettingsGroupDisplayName	Display name of the windows settings group
SetAsMigrated	Only applies if containing user group is in import mode. If non-null, specifies if the WSG should be set as migrated after deleting the data.
SourceSystemId	Source System id (0/null = local, else published) *(10.1 FR2)*

(10.0) Removed UserGroup, added source system id

FetchArchiveReport

Remarks

Returns an archive report for the specified user

Declaration

profilemanagement.profileservice.datacontracts.ProfileAnalysisArchive[]
FetchArchiveReport(string UserName)

Returns

List of archive report records, one for each application with data. See API Classes section. Throws exception on error

Parameters

UserName	Name of user (domain\name)

(10.0) Removed UserGroupName

FetchReport

Remarks

Returns the data for a profile analysis report (other than an archive report). This function is identical to FetchReport2 (see below) except for the SortOrder parameter.

Declaration

profilemanagement.profileservice.datacontracts.ProfileAnalysisPlotPoint[]
FetchReport(string UserGroupName,
int ReportType,
int ItemType,
string ItemName,
DateTime? StartDate,
DateTime? EndDate,
int UIPageSize,
int? StartRank,
int? NumRows,
out ProfileAnalysisPlotSummary Summary)

FetchReport2

Remarks

Returns the data for a profile analysis report (other than an archive report). . The maximum number of rows returned is controlled by either the UIPageSize parameter (if StartRank is null) or NumRows (if StartRank is specified). Paging through the data is done by noting the MaxRank (in the summary object) and the rank in each plot point, and calling the method again for the next desired block of data. See following notes:

Caution

The first plot point in the returned list has a rank of 2, not 1. This is because rank 1 is used internally to communicate totals between the database and the personalization server. Rank 1 is never presented to the API.

When using PowerShell, passing $null to a string parameter (ItemName in this instance) coerces it to an empty string, which causes the call to fail. Instead call the function via reflection and a parameter list, similar to the following:

$params = @(“Default Users”,1,1,$null,$null,$null,0,20,$null,$null,$null)

$report =

$pmClient.GetType().GetMethod(“FetchReport2”).Invoke($pmClient,$params)

$summary = $params[10]

(10.0) A report for all user groups can be retrieved by specify a user group name of ‘%’.

Declaration

```
profilemanagement.profileservice.datacontracts.ProfileAnalysisPlotPoint[]
        FetchReport2(string UserGroupName,
                                int ReportType,
                                int ItemType,
                                string ItemName,
                                DateTime? StartDate,
                                DateTime? EndDate,
                                int SortOrder,
                                int UIPageSize,
                                int? StartRank,
                                int? NumRows,
                                int? UserGroupSourceSystemId,
                                int? ItemSourceSystemId,
                                out ProfileAnalysisPlotSummary Summary)
```

Returns

Returns an array of ProfileAnalysisPlotPoint structures, each one representing a bar on a profile analysis graph. A summary is also returned as an output parameter.

Parameters

UserGroupName	Name of group. Specifying a percent sign will return a report for all groups
ReportType	0 = usage, 1 = size, 3 = discovered application usage
ItemType	Type of item in ItemName, 0 = application, 1 = user
ItemName	Name of item. If empty, report is for all applications/users according to ItemType
StartDate	Start time/date for a usage graph
EndDate	End time/date for a usage graph
SortOrder	0 = plot value descending, 1 = plot name ascending
UIPageSize	Maximum number of rows to be returned if StartRank is null
StartRank	If non-null, first rank of data to be returned
NumRows	If start rank is not null, maximum number of rows to return
UserGroupSourceSystemId	Source system id of user group name (0/null = local, >0 published) *(10.1 FR2)*
ItemSourceSystemId	Source system id of item, depending on item type (0/null = local, >0 published) *(10.1 FR2)*
Summary	(out) returned summary structure (see API Classes)

ForceReplicationNow

(Removed)

GeoSync_ExecuteGeoSyncCommand

Remarks

On a geosync-enabled publisher system, executes a sync command

Declaration

profilemanagement.profileservice.datacontracts.GeoCommandResponse
GeoSync_ExecuteGeoSyncCommand(
profilemanagement.profileservice.datacontracts.GeoCommand command)

Returns

Command Response object in API classes section.

Parameters

Command	GeoSyncCommand as detailed in API classes section

GeoSync_GetRemotePublisherList

Remarks

Returns remote publishers. Will return details of the single publisher on a subscriber.

Declaration

profilemanagement.profileservice.datacontracts.GeoRemoteServiceList
GeoSync_GetRemotePublisherList()

Returns

GeoRemoteServiceList object containing single publisher

GeoSync_GetRemoteSubscriberList

Remarks

Returns remote subscribers. Will return details of the all subscribers on a publisher.

Declaration

profilemanagement.profileservice.datacontracts.GeoRemoteServiceList
GeoSync_GetRemoteSubscriberList()

Returns

GeoRemoteServiceList object containing subscribers

GeoSync_GetSourceSystemName

Remarks

Gets display name associated with a source system id. This display name identifies the publisher on a subscriber system. The display name returned when SourceSystemId is zero and the system is a subscriber is the name of the current system as seen on the publisher.

Declaration

string GeoSync_GetSourceSystemName(

int SourceSystemId)

Returns

Display name of publisher if SourceSystemId is >0.

Parameters

SourceSystemId	Source system id

GeoSync_GetSyncSchedule

Remarks

Return synchronization schedule for a subscriber service.

Declaration

profilemanagement.profileservice.datacontracts.GeoSyncSchedule

GeoSync_GetSyncSchedule(string serviceName)

Returns

Sync schedule object (see API classes)

Parameters

ServiceName	Full name of subscriber service (ServiceName in Geo.RemoteService table), not the display name.

GeoSync_IsRemotePublisher

Remarks

Returns true if current system is a publisher.

Declaration

bool GeoSync_IsRemotePublisher()

Returns

true if publisher

GeoSync_IsRemoteSubscriber

Remarks

Returns true if current system is a subscriber.

Declaration

bool GeoSync_IsRemoteSubscriber()

Returns

true if subscriber

GeoSync_UpdateSyncSchedule

Remarks

Write sync schedule to a subscriber service

Declaration

void GeoSync_UpdateSyncSchedule(
profilemanagement.profileservice.datacontracts.GeoSyncSchedule Schedule,
bool UpdateLastSyncTime)

Returns

none

Parameters

Schedule	Schedule object (see API classes)
UpdateLastSyncTime	Updates last sync time to current database time if true

GetAllApplicationDataByName

Remarks

Returns all the application data for a specified application for a specified user in a specified application group. Optionally the data blob for each row can be returned.

Declaration

profilemanagement.profileservice.datacontracts.ApplicationData[]
GetAllApplicationDataByName(string UserName,
string ApplicationGroupName,
string WindowsSettingsGroupName,
int? SourceSystemId,
out Guid ApplicationProfilePK)

Returns

List of application data rows. Throws exception on error.

Parameters

UserName	Name of user (domain\name)
ApplicationGroupName	Name of applicationgroup - use string guid (with braces) for special applications
WindowsSettingsGroupName	If not null or empty, gets the windows settings group data identified by the name from a windows settings profile (original name)
SourceSystemId	Source system id qualifying application group / wsg name *(10.1 FR2)*
ApplicationProfilePK	(out) Guid of application profile

(10.0) Removed extraneous parameters and added source system id

GetApplicationData

Remarks

Returns a specific application data row

Declaration

profilemanagement.profileservice.datacontracts.ApplicationData
GetApplicationData(Guid ApplicationProfilePK,
int FileHash)

Returns

ApplicationData object. Throws exception on error.

Parameters

ApplicationProfilePK	Guid of application profile
FileHash	Hash of relative path and filename (ApplicationDataPK)

GetApplicationDataRegistrySettings

Remarks

Returns the registry data blob for the specified application group. The data blob is in the form of a compressed fbr (file-based registry) file or a compressed hive file. When applied to a Windows Settings profile, returns the legacy registry data (used by pre-8.5 clients).

Note

ApplicationGroupName must be passed in Guid format to retrieve Windows Settings data.

Declaration

int GetApplicationDataRegistrySettings(string UserName,
string ApplicationGroupName,
int? SourceSystemId,
out byte[] Data,
out int ApplicationType,
out Guid ApplicationProfilePK,
out int SettingsType)

Returns

Returns the hash of the filename and path from the ApplicationData table's ApplicationDataPK field. If there is no registry data returns 0.

Parameters

UserName	Name of user
ApplicationGroupName	Application group name
SourceSystemId	Source system id qualifying application group name *(10.1 FR2)*
Data (out)	Data blob (compressed fbr). For PowerShell use, variable must be predefined as an array with e.g. $data = @() and then passed by reference, e.g. [ref] $data
ApplicationType (out)	Application type from ApplicationProfile table
ApplicationProfilePK (out)	Application profile GUID
SettingsType (out)	0 = fbr, 1 = hive file

(10.0) UserGroupName removed, compressed bool removed, settings type & source system id added.

GetApplicationDataWindowsSettingsRegistry

Remarks

Returns the registry data blob for the specified Windows Settings group in the user's Windows Settings profile. The data blob is in the form of a compressed fbr (file-based registry) file.

Declaration

```
int GetApplicationDataWindowsSettingsRegistry(
        string UserName,
        string WindowsSettingsGroupName,
        int? SourceSystemId,
        out Byte[] Data,
        out Guid ApplicationProfilePK,
        out int SettingsType)
```

Returns

Returns the hash of the filename and path from the ApplicationData table's ApplicationDataPK field. If there is no registry data returns 0.

Parameters

UserName	Name of user
WindowsSettingsGroupName	Windows setting group display name
SourceSystemId	Source system id quaifying wsg name
Data (out)	Data blob (compressed fbr) For PowerShell use, variable must be predefined as an array with e.g. $data = @() and then passed by reference, e.g. [ref] $data
ApplicationType (out)	Application type from ApplicationProfile table

ApplicationProfilePK (out)	Application profile GUID
SettingsType (out)	0 = fbr file, 1 = hive file

GetApplicationsInUserGroup

Remarks

Gets a list of all application groups associated with a user group (which could appear in a personalization analysis graph) Order is:

1. Application groups in the user group whitelist
2. Special applications (Desktop Settings etc.)

Declaration

```
profilemanagement.profileservice.datacontracts.ProfileAnalysisApplication[]
                GetApplicationsInUserGroup(string UserGroupName,
                                           int? SourceSystemId)
```

Returns

List of applications and groups

Parameters

UserGroupName	Name of user group
SourceSystemId	Source system id qualifying user group id

GetLastAccess

Remarks

Gets the time a profile was last accessed

Declaration

```
DateTime? GetLastAccess(string UserName,
                        string ApplicationGroupName,
                        int? SourceSystemId,
                        out int Status,
                        out GroupMigrationTypeEnum GroupMigrationType,
                        out bool ProfileMigrated)
```

Returns

Date and time the profile was last accessed. Returns null if access time not found (see Status)

Parameters

UserName	Name of user (domain\name)
ApplicationGroup	Application Group Name
SourceSystemId	Source system id qualifying application group name *(10.1 FR2)*
Status	0 = OK, 1 = profile not found (may be concurrency error), 2 = profile found but no history available
GroupMigrationType (out)	Migration type of parent user group: 0 - Disabled 1 - Import 2 - ImportForce 3 - Export
ProfileMigrated (out)	True if application group has already been migrated. Invalid if GroupMigrationType is 'Disabled'

(10.0) Removed UserGroupName

GetLastWindowsSettingsGroupAccess

Remarks

Gets the time a windows setting group was last accessed

Declaration

```
DateTime? GetLastWindowsSettingsGroupAccess(
        string UserName,
        string WindowsSettingsGroupDisplayName,
         int? SourceSystemId,
        out int Status,
        out ProfileService.ServiceContracts.GroupMigrationTypeEnum GroupMigrationType,
        out bool ProfileMigrated)
```

Returns

Date and time the profile was last accessed. Returns null if access time not found (see Status)

Parameters

UserName	Name of user (domain\name)
WindowsSettingsGroupDisplayName	Windows Settings Group name
SourceSystemId	Source system id qualifying wsg name *(10.1 FR2)*

Status (out)	0 = OK, 1 = profile not found (may be concurrency error), 2 = profile found but no history available
GroupMigrationType (out)	Migration type of parent user group: 0 - Disabled 1 - Import 2 - ImportForce 3 - Export
ProfileMigrated (out)	True if group has already been migrated. Invalid if GroupMigrationType is 'Disabled'

(10.0) Removed UserGroupName

GetUsersInGroup

Remarks

Searches for a list of users in a specified user group (personalization group) that match the search criteria.

Note

This function only returns a maximum of 500 users as it is intended as an aid to searching for specific users.

Declaration

```
profilemanagement.profileservice.datacontracts.ProfileAnalysisUser[]
                                    GetUsersInGroup(string UserGroup,
                                                    string LoginNameWildcard,
                                                    string DomainNameWildcard,
                                                    int? SourceSystemId)
```

Returns

List of users found. If this list contains 501 entries, more users exist that match the criteria but were not returned.

Parameters

UserGroup	Name of user group (% returns all)
LoginNameWildcard	SQL-style wildcard expression for user names (% returns all)
DomainNameWildcard	SQL-style wildcard expression for domain names (% returns all)
SourceSystemId	Source system id qualifying UserGroup name *(10.1 FR2)*

GetWindowsSettingsGroupsInArchive

Remarks

Returns details of windows settings groups in an archive of a windows settings profile.

Declaration

```
profilemanagement.profileservice.datacontracts.
ProfileAnalysisWindowsSettingsGroupArchiveDetail[]
                GetWindowsSettingsGroupsInArchive(int ArchiveId, int? SourceSystemId)
```

Returns

List of archive details for the specified archive identity

Parameters

ArchiveId	Integer system-wide archive id
SourceSystemId	If non-null, restricts results to the specified source system. Null returns all Wsgs. This is to cater for the possibility of a profile with both local and published wsgs *(10.1 FR2)*

GetWindowsSettingsGroupsInProfile

Remarks

Returns details of windows settings groups in a windows settings profile. No size information is returned

Declaration

profilemanagement.profileservice.datacontracts.
ProfileAnalysisWindowsSettingsGroupDetail[] GetWindowsSettingsGroupsInProfile(
string UserName)

Returns

List of windows setting group details for the user's windows settings profile.

Parameters

UserName	User name (domain\user)

(*10.0)* Removed UserGroupName

Global_Get

Remarks

Returns the single-row global data

Declaration

profilemanagement.profileservice.datacontracts.Global Global_Get()

Returns

Global object. Throws on error.

GlobalProperty_DeleteByName

Remarks

Deletes a value from the GlobalProperty table. This table is accessible via the console “Advanced Settings” function.

Caution

Deleting any Global Property may have an unexpected effects including system failure. Any value deleted by this function can be restored with the GlobalProperty_RestoreAll function. However this also restores all values to their factory settings.

Declaration

profilemanagement.profileservice.datacontracts.Result
GlobalProperty_DeleteByName(string Name, DateTime ModifiedTime)

Returns

Result.ErrorCode = 0. Throws exception on error.

Parameters

Name	Name of property
ModifiedTime	Time for concurrency check

GlobalProperty_GetAll

Remarks

Retrieve all value from the GlobalProperty table. This table is accessible via the console “Advanced Settings” function.

Declaration

profilemanagement.profileservice.datacontracts.GlobalProperty[]
GlobalProperty_GetAll()

Returns

List of GlobalProperty objects. Throws exception on error

GlobalProperty_GetByName

Remarks

Retrieve a value from the GlobalProperty table. This table is accessible via the console “Advanced Settings” function.

Declaration

profilemanagement.profileservice.datacontracts.GlobalProperty
GlobalProperty_GetByName(string Name)

Returns

GlobalProperty object. Throws exception on error

Parameters

Name	Name of property

GlobalProperty_Insert

Remarks

Add a (non-system) setting to the GlobalProperty table.

Declaration

profilemanagement.profileservice.datacontracts.GlobalProperty
GlobalProperty_Insert(string Name,
string Value,
string Description)

Returns

Resultant GlobalProperty object. Throws exception on error

Parameters

Name	Name of property
Value	Value of property
Description	Text description

GlobalProperty_RestoreAll

Remarks

Restore all properties to factory settings. This function will remove any non-system settings, undelete any deleted system settings and set all those settings to their default values.

Declaration

profilemanagement.profileservice.datacontracts.GlobalProperty[]
GlobalProperty_RestoreAll()

Returns

List of GlobalProperty objects after the restore. Throws exception on error

GlobalProperty_Update

Remarks

Modifies an existing GlobalProperty setting.

Declaration

profilemanagement.profileservice.datacontracts.GlobalProperty
GlobalProperty_Update(string Name,
string Value,
string Description,
DateTime ModifiedTime)

Returns

Modified GlobalProperty object

Parameters

Name	Name of global property to modify
Value	New value
Description	New description
ModifiedTime	Existing modified time for concurrency checking

GroupBasedDesktopSettingsStates_GetStateByUserGroup *(OBSOLETE)*

Remarks

This function still exists but will only return information relevant to legacy clients. Refer to earlier versions of this document for a description.

GroupBasedDesktopSettingsStates_UpdateState *(OBSOLETE)*

Remarks

This function still exists but will only return information relevant to legacy clients. Refer to earlier versions of this document for a description.

InsertApplicationData

Remarks

Inserts an ApplicationData object into the database. Calculates and adds the file hash (ApplicationDataPK). This function can be used to add file data to an existing profile.

The profile is located by the ApplicationProfilePK in the ApplicationData object. The binary data should be compressed if large (although this is not mandatory). Compression is via the gzip algorithm.

This function will update the profile timestamp so that clients will receive the new data on the next sync.

Declaration

int InsertApplicationData(
profilemanagement.profileservice.datacontracts.ApplicationData,
bool OverwriteIfExists,
out int FileHash)

Returns

Integer status: 0 OK, 1 = data exists and OverwriteIfExists is false

Parameters

ApplicationData	Data to be inserted
OverwriteIfExists	If true, overwrite existing
FileHash	Filehash generated for the new row

MakeArchiveNow

Remarks

Creates an archive for the specified user and application

Declaration

profilemanagement.profileservice.datacontracts.Result
MakeArchiveNow(string UserName,
string ApplicationGroupName,
int? SourceSystemId,
bool IsProtected,
string Description)

Returns

Result.ErrorCode: 0 - OK 1 - profile doesn't exist

Parameters

UserGroupName	User group
UserName	User Name
ApplicationName	Application Group name (GUID for Windows Settings)
SourceSystemId	Source system qualifying application group name *(10.1 FR2)*
IsProtected	Protect the archive against automatic deletion. Only one archive per user can be protected.
Description	Archive description

Membership_Get

Remarks

Returns membership record identified by membership id (Guid)

Declaration

profilemanagement.profileservice.datacontracts. Membership
Membership_Get(Guid MembershipPK)

Returns

Membership record. Throws exception on error.

Note

A non-existent GUID produces a null object exception.

Parameters

MembershipPK	Guid of membership in Membership table

MoveDataToGroup

Remarks

Moves a user's data from one User Group to another

Declaration

profilemanagement.profileservice.datacontracts.Result
MoveDataToGroup (string UserName,
string SourceGroupName,
string DestinationGroupName
bool IncludeDiscovered,
bool Overwrite)

Returns

Result.Error code 0 = data exists in destination group but override not set (error), 1 = OK, worked, 2 = worked OK but new group's whitelists don't match old (warning), -1 user or group does not exist

Parameters

UserName	User name (domain\user)
SourceGroupName	User group of source data
DestinationGroupName	Destination user group
IncludeDiscovered	if true, include discovered applications
Overwrite	if true, overwrite user's existing data in destination group

Passive_Reset

Remarks

Remove a Path (registry or folder) from an inclusion or exclusion list. The parent guid is either an application id, application group id or the global parent id (`{10732D21-EA06-4588-8144-D79EBD0EB2FF}`) for global lists.

Declaration

profilemanagement.profileservice.datacontracts.Result
Path_Delete(Guid PathID, Guid ParentId, DateTime ModifiedTime)

Parameters

PathID	Guid of path in Path table

ParentId	Guid of parent (extra check)
ModifiedTime	Modified Time of Path

Path_Delete

Remarks

Remove a Path (registry or folder) from an inclusion or exclusion list. The parent guid is either an application id, application group id or the global parent id (`{10732D21-EA06-4588-8144-D79EBD0EB2FF}`) for global lists.

Declaration

profilemanagement.profileservice.datacontracts.Result
Path_Delete(Guid PathID, Guid ParentId, DateTime ModifiedTime)

Parameters

PathID	Guid of path in Path table
ParentId	Guid of parent (extra check)
ModifiedTime	Modified Time of Path

Path_GetByParentAndPath

Declaration

profilemanagement.profileservice.datacontracts.Path
Path_GetByParentAndPath(Guid ParentId,
Guid PathId)

Parameters

ParentId	Guid of parent (extra check)
PathId	Guid of path in Path table

Path_Insert

Remarks

Add a new Path (registry or folder) to an inclusion or exclusion list. The list is identified by a parent id (Guid), which may be an application (for application-specific paths), an application group (for application group-specific paths) or the global parent id for the global lists. The global parent id is `{10732D21-EA06-4588-8144-D79EBD0EB2FF}`.

Caution

The parent guid must be either the global guid or an application or application group guid. Adding an unknown guid produces an orphaned entry in the Path table. Deleting the parent does not remove the entry from the Path table, which again will become orphaned.

Declaration

profilemanagement.profileservice.datacontracts.Path
Path_Insert (string Value,
bool Include,
int Type,
Guid ParentId)

Returns

Path record inserted, including modified time and new Guid. On error exception is thrown.

Parameters

Value	Registry key or folder name. Folder names may be start with a CSIDL variable (in braces) or environment variable (surrounded by % characters)
Include	True = inclusion, false = exclusion
Type	Type of path, 0 = registry key, 1 = folder
ParentId	Application or ApplicationGroup Guid, or global parent id {10732D21-EA06-4588-8144-D79EBD0EB2FF}

Path_Update

Remarks

Modify an existing path record.

Declaration

profilemanagement.profileservice.datacontracts.Path Path_Update(
Guid PathId,
Guid ParentId,
string Value,
bool Include,
int Type,
DateTime ModifiedTime)

Returns

Path record modified.

Parameters

PathId	Identity of record to update
ParentId	Path of record to update (extra check)
Value	New value
Include	true if include type
Type	0 = registry, 1 = folder
ModifiedTime	Current time on record for concurrency check

Passive_Reset

Remarks

Removes all data collection data from the database. This is the passive trace data collected by endpoints when ‘Enable Data Collection’ is checked on the User Group.

Declaration

void Passive_Reset()

Caution

This function deletes all passive trace data and cannot be reversed.

RestoreArchive

Remarks

Restores an archive, identified by its integer archive id, to the current profile. The archive id can be retrieved from the archive report (see FetchArchiveReport).

Caution

Restoring an archive overwrites the current data for the user and application.

Declaration

int RestoreArchive(int ArchiveID)

Returns

Status: 0 = OK, 1 = Archive doesn't exist (may be concurrency failure, i.e. archive deleted by someone else). Other errors cause an exception to be thrown.

Parameters

ArchiveID	Integer archive identity from the relevant ProfileAnalysisArchiveDetail structure returned from a FetchArchiveReport. This identifies the archive globally (user name and application need not be specified)

ResetAllUsersData

Remarks

Deletes all personalization data for all profiles for a specified user.

Declaration

int ResetAllUsersData(
string UserName,
bool SetAsMigrated)

Returns

Status: 0 = OK, 1 = User or UserGroup no longer exists

Parameters

UserName	Name of user (domain\user)
SetAsMigrated	Whether empty profiles should be set as migrated if a migration mode is active on the user group. Ignored if migration not active

(*10.0)* Removed UserGroupName

RestoreWindowsSettingsGroupFromArchive

Remarks

Restores a WindowsSettingsGroup from an archive. The archive is identified by its integer archive id. The archive id can be retrieved from the archive report (see FetchArchiveReport). The group name is the original name of the windows settings group, not the display name.

Caution

Restoring an archive overwrites the current data for the user's Windows Settings Group.

Declaration

int RestoreWindowsSettingsGroupFromArchive(int ArchiveId,
string WindowsSettingsGroupName)

Returns

Status: 0 = OK, 1 = Archive doesn't exist (may be concurrency failure, i.e. archive deleted by someone else). Other errors cause an exception to be thrown.

Parameters

ArchiveID	Integer archive identity from the relevant ProfileAnalysisArchiveDetail structure returned from a FetchArchiveReport. This identifies the archive globally (user name and application need not be specified). Archive must be for a Windows Settings Group
WindowsSettingsGroupName	Original name of the Windows Settings Group. The original name for a known display name can be found using the function WindowsSettingsGroup_GetGroupByDisplayName

Server_Delete

Remarks

Deletes a record in the Server table

Declaration

profilemanagement.profileservice.datacontracts.Result
Server_Delete(Guid ServerId, DateTime? ModifiedTime)

Returns

Result.ErrorCode = 0, throws exception on error

Parameters

ServerId	Id of row to delete
ModifiedTime	Current modified time for concurrency check

Server_Get

Remarks

Returns a record from the Server table by Guid

Declaration

profilemanagement.profileservice.datacontracts.Server
Server_Get(Guid ServerId)

Returns

Server object.

Parameters

ServerId	Id of row to retrieve

Server_GetAll

Remarks

Returns all Servers

Declaration

profilemanagement.profileservice.datacontracts.Server[]
Server_GetAll()

Returns

List of all Servers

Server_GetAllBySite

Remarks

Returns all Servers defined on a site

Declaration

profilemanagement.profileservice.datacontracts.Server[]
Server_GetAllBySite(Guid SiteId)

Returns

List of all Servers on the site

Parameters

SiteId	Guid of owning site. Default site has the well-known Guid `{00000000-0000-0000-3333-000000000000}`

Server_Insert

Remarks

Insert a new server into the Server table

Declaration

```
profilemanagement.profileservice.datacontracts.Server
        Server_Insert(string Name,
                      string Url
                      int Port,
                      string Description,
                      string DistinguishedName,
                      Guid? ADObjectId,
                      Guid SiteId,
                      string FQDN,
                      string DNSSuffix,
                      ServerTypeEnum ServerType,
                      int? VirtualHostRetries,
                      int? VirtualHostTimeOut)
```

Returns

Created server record

Parameters

Fields of server object required (see Server class). SiteId must already exist.

Server_Update

Remarks

Modify an existing server record

Declaration

profilemanagement.profileservice.datacontracts.Server
Server_Update(Guid ServerId,
string Name,
string Url,
int Port,
string Description,
string DistinguishedName,
Guid? ADObjectId,
Guid SiteId,
DateTime? ModifiedTime,
string FQDN,
string DNSSuffix,
int? VirtualHostRetries,
int? VirtualHostTimeOut)

Returns

Modified server record

Parameters

ServerId	Guid of server to modify
Name	new name
Url	new url
Port	new port
Description	new description
DistinguishedName	new AD name
ADObjectId	new AD object Guid
SiteId	new SiteId (must exist in Site table)
ModifiedTime	Time for concurrency check
FQDN	Fully qualified domain name
DNSSuffix	DNS suffix of FQDN
VirtualHostRetries	Number of retries the server will perform on a failed virtual host before moving to the next one in the list.

VirtualHostTimeOut	Timeout between retries the server will perform on a failed virtual host.

Server_UpdateOrder

Remarks

Modify the precedence of a server or host within a site

Declaration

profilemanagement.profileservice.datacontracts.Server
Server_UpdateOrder(Guid ServerId,

int OrderNo,

DateTime? ModifiedTime

Guid SiteId)

Returns

Modified server record

Parameters

ServerId	Guid of server to modify
OrderNo	Order precedence (low number = higher priority)
ModifiedTime	Time for concurrency check
SiteId	Guid of the site the server is assigned to

SessionData_GetByUserGroup

Remarks

Return all session data assigned globally or to a personalization group. If ‘Enable global session data’ is set on the personalization group this must be called twice once to get the personalization group session data and once for the inherited global session data

Declaration

profilemanagement.profileservice.datacontracts.SessionData[]

SessionData_GetByUserGroup(Guid UserGroup)

Returns

Collection of SessionData Objects.

Parameters

UserGroup	Guid of the personalization group or GlobalID for global session data.

SessionData_Update

Remarks

Update a collection of Session Data objects where the IUDAction set to the relevant action (Insert, Update, and Delete).

Note that Update and Delete operations check for concurrency errors so must be called with the exact session data modified time to avoid exception

Declaration

void SessionData_Update(

ref profilemanagement.profileservice.datacontracts.SessionData[] SessionData)

Returns

Throws exception on error.

Parameters

SessionData[]	Array of SessionData.

Site_Delete

Remarks

Deletes a record in the Site table

Declaration

profilemanagement.profileservice.datacontracts.Result
Site_Delete(Guid SiteId, DateTime? ModifiedTime)

Returns

Result.ErrorCode = 0, throws exception on error

Parameters

SiteId	Id of row to delete
ModifiedTime	Current modified time for concurrency check

Site_Get

Remarks

Returns a record from the Site table by Guid

Declaration

profilemanagement.profileservice.datacontracts.Server Site_Get(Guid SiteId)

Returns

Site object.

Parameters

SiteId	Id of row to retrieve

Site_GetAll

Remarks

Returns all Sites

Declaration

profilemanagement.profileservice.datacontracts.Site[] Site_GetAll()

Returns

List of all Sites

Site_GetWithMemberships

Remarks

Returns a row from the Site table by SiteId, together with associated memberships and servers.

Declaration

```
profilemanagement.profileservice.datacontracts.Site
        Site_GetWithMemberships(Guid SiteId,
                                out Membership[] Memberships,
                                out Server[] Servers)
```

Returns

Site object.

Parameters

SiteId	Guid of site to retrieve
Memberships	Associated memberships (out) For PowerShell use, variable must be predefined as an array with e.g. $memberships = @() and then passed by reference, e.g. [ref] $memberships
Servers	Associated servers (out)

SystemInfo_Get

Remarks

Get system information

Declaration

```
profilemanagement.profileservice.datacontracts.SystemInfo SystemInfo_Get()
```

Returns

SystemInfo record

UpdateApplicationDataBlob

Remarks

Writes the data back to an ApplicationData row in the database for a specified application profile and file. Data is normally written back compressed (gzip format), but uncompressed data will also work on endpoints provided the data is not an fbr (registry) file.

Declaration

```
profilemanagement.profileservice.datacontracts.Result[]
                    UpdateApplicationDataBlob(Guid ApplicationProfilePK,
                                              int ApplicationDataPK,
                                              byte[] Data,
                                              int UncompressedSize)
```

Returns

Result, error code = 0. Any errors cause an exception to be thrown

Parameters

ApplicationProfilePK	Guid of application profile
ApplicationDataPK	File/folder has identifying the row of ApplicationData
Data	Binary data to be inserted
UncompressedSize	Size in bytes of data when uncompressed. (Matches size of data if uncompressed)

UpdateArchive

Remarks

Update an archive description and IsProtected flag. Only one archive per user can be configured as protected. If an archive is protected, any archive previously protected will be changed to unprotected. The archive id can be retrieved from the archive report (see FetchArchiveReport).

Declaration

int UpdateArchive(int ArchiveID,

bool IsProtected,

string Description)

Returns

Result, error code = 0. Any errors cause an exception to be thrown

Parameters

ArchiveID	Integer archive identity from the relevant ProfileAnalysisArchiveDetail structure returned from a FetchArchiveReport. This identifies the archive globally (user name and application need not be specified)
IsProtected	Protect this archive against automatic deletion
Description	Description text (max 20 char)

User_Delete

Remarks

Deletes a user from the database by GUID, including all profile data, archives and usage history in all user groups (it is possible for users to have profiles in separate groups).

Caution

If the user has a large amount of data, deletion may not be possible within the PS query timeout period. If this happens, an exception is thrown and the deletion does not occur - the transaction is rolled back.

Declaration

profilemanagement.profileservice.datacontracts.Result
User_Delete(Guid UserId, DateTime ModifiedTime)

Returns

Result.ErrorCode = 0. Error cause an exception to be thrown

Parameters

UserId	Guid identifying user
ModifiedTime	Timestamp of user for concurrency check - if the modified time doesn't match the user is not deleted.

User_Get

Remarks

Returns user details by UserId.

Declaration

profilemanagement.profileservice.datacontracts.User
User_Get(Guid UserId)

Returns

User record

Parameters

UserId	Guid identifying user

User_GetAll

Remarks

Returns all user details. It may not be possible to call this function on large databases because of message size limitations.

Declaration

profilemanagement.profileservice.datacontracts.User[] User_GetAll()

Returns

List of User records

UserGroup_Delete

Remarks

Delete UserGroup (Personalization Group) by user group id. All associated records are deleted or marked for delete**, including users in the group and all their data**. Marked for delete items are removed by the daily cleanup job.

Declaration

profilemanagement.profileservice.datacontracts.Result
UserGroup_Delete(Guid GroupId, DateTime ModifiedTime)

Returns

Result.ErrorCode = 0. Throws exception on error

Parameters

UserGroupID	User group guid
ModifiedTime	Concurrency check - if record modified time doesn't match, no deletion occurs and exception is thrown.

UserGroup_Get

Remarks

Get UserGroup details by user group id

Declaration

profilemanagement.profileservice.datacontracts.UserGroup
UserGroup_Get(Guid GroupId,
out Membership[] Memberships,
out Membership[] OfflineRules,
out Membership[] EndpointSelfServiceRules
)

Returns

User group details. Throws exception on error.

Parameters

UserGroupID	User group guid
Memberships	Returns list of associated memberships (from membership table). Memberships are conditions which must be satisfied for a user to be a member of the user group. All memberships associated with the group are returned (first and second level). For PowerShell use, variable must be predefined as an array with e.g. $memberships = @() and then passed by reference, e.g. [ref] $memberships
OfflineRules	Returns list of associated offline rules. These are conditions which must be satisfied for offline mode to be enabled
EndpontSelfServiceRules	Returns list of associated Endpoint Self Service rules. These are conditions which must be satisfied for the endpoint self-service tool to be enabled at the client

UserGroup_GetAll

Remarks

Returns list of all user groups

Declaration

profilemanagement.profileservice.datacontracts.UserGroup[] UserGroup_GetAll()

Returns

List of user groups. Any errors cause an exception to be thrown

UserGroup_GetGeoSyncList

Remarks

Gets GeoSync list for user group

Declaration

profilemanagement.profileservice.datacontracts.UserGroupGeoSyncList
UserGroup_GetGeoSyncList(Guid userGroupId)

Returns

Geosync list object

Parameters

UserGroupId	Guid of User group

UserGroup_UpdateGeoSyncList

Remarks

Updates a group's geosync list

Declaration

profilemanagement.profileservice.datacontracts.Result
UserGroup_UpdateGeoSyncList(
profilemanagement.profileservice.datacontracts.UserGroupGeoSyncList GeoSyncList)

Returns

Resutl object

Parameters

GeoSyncList	GeoSyncList object. User group id is in this object. 'AvailableRemoteServices' is ignored on update

UserGroupApplication_Add

Remarks

Adds an application group to a user group

Declaration

profilemanagement.profileservice.datacontracts.Result UserGroupApplication_Add(
Guid ApplicationId,
Guid UserGroupId)

Returns

Throws exception on error.

Parameters

ApplicationId	Guid of application group
UserGroupId	Guid of User group

UserGroupApplication_Remove

Remarks

Removes an application group from a user group

Declaration

profilemanagement.profileservice.datacontracts.Result
UserGroupApplication_Remove (
Guid ApplicationId, Guid UserGroupId)

Returns

Throws exception on error.

Parameters

ApplicationId	Guid of application group
UserGroupId	Guid of User group

WindowsSettingsCondition_Create

Remarks

Creates a new Windows Settings Condition to be associated with a Windows Settings Group. After creation the condition must be associated with a Windows Settings Group or it will become inaccessible.

Documentation of the xml formats required for UEMConditions and EMXConditions is not provided here. It is recommended that the console is used to create conditions

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsCondition
WindowsSettingsCondition_Create(
string Name,
string Description,
string UEMConditions,
string EMXConditions,
bool AvailableForReuse)

Returns

Object representing the created condition

Parameters

Name	Name of condition. May be left blank for non-reusable conditions
Description	Text description of condition
UEMConditions	Xml description of condition tree in UEM (Console) format
EMXConditions	Xml description of condition tree in EMX (Client) format
AvailableForReuse	Condition can be used in multiple Windows Settings Groups

WindowsSettingsCondition_DeleteById

Remarks

Deletes a windows settings condition by its Guid identity. The current modified time of the condition must be supplied as a concurrency check - if the modified time differs from that in the database, the delete fails.

Declaration

void WindowsSettingsCondition_DeleteById(
Guid ConditionId,
DateTime ModifiedTime)

Parameters

ConditionId	Guid id of condition to be deleted
ModifiedTime	Modified time of condition to be deleted for concurrency check

WindowsSettingsCondition_GetAll

Remarks

Returns array of all defined conditions

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsCondition[]
WindowsSettingsCondition_GetAll()

Returns

Array of all defined conditions.

WindowsSettingsCondition_GetGroups

Remarks

Retrieves list of WindowsSettingsGroups using a specified condition

Declaration

**profilemanagement.profileservice.datacontracts.
WindowsSettingsConditionGroupAssociation[] WindowsSettingsCondition_GetGroups(
Guid ConditionID)**

Returns

Array of WindowsSettingsConditionGroupAssociation objects

Parameters

ConditionId	Identity of condition

WindowsSettingsCondition_Update

Remarks

Modifies a windows settings condition.

Declaration

**DateTime WindowsSettingsCondition_Update(
profilemanagement.profileservice.datacontracts.WindowsSettingsCondition
Condition)**

Returns

New modified time of the condition

Parameters

Condition	Condition to be modified. Modified time in object must match database or a concurrency error occurs

WindowsSettingsGroup_AssignCondition

Remarks

Assigns a condition to a windows settings group

Declaration

DateTime WindowsSettingsGroup_AssignCondition(
Guid WindowsSettingsGroupId,
Guid ConditionId,
DateTime WindowsSettingsGroupModifiedTime)

Returns

New modified time of the windows settings group

Parameters

WindowsSettingsGroupId	Identity of windows settings group
ConditionId	Identity of condition
WindowsSettingsGroupModifiedTime	Modified time for windows settings group for concurrency checking

WindowsSettingsGroup_AssignToUserGroup

Remarks

Assigns one or more Windows Settings Groups to a single user group.

Declaration

DateTime WindowsSettingsGroup_AssignToUserGroup(
Guid UserGroupId,
Guid[] WindowsSettingsGroupIds,
DateTime UserGroupModifiedTime)

Returns

New modified time of the user group

Parameters

UserGroupId	Identity of user group
WindowsSettingsGroupIds	Array of windows settings group ids to assign
UserGroupModifiedTime	Modified time for user group for concurrency checking

WindowsSettingsGroup_CloneGroup

Remarks

Clones a windows settings group and optionally its associated condition

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup
WindowsSettingsGroup_CloneGroup(
string Name,
string Description,
Guid OriginalGroupId,
Guid? OriginalConditionId)

Returns

Object representing the new windows settings group.

Parameters

Name	Name for new group
Description	Description for new group
OriginalGroupId	Id of group to be cloned
OriginalConditionId	If not null, id of condition to be cloned and assigned to new group

WindowsSettingsGroup_Create

Remarks

Creates a new empty windows settings group.

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup
WindowsSettingsGroup_Create(
string Name,
string Description)

Returns

Newly-created windows settings group

Parameters

Name	Name of new group
Description	Description of new group

WindowsSettingsGroup_CreateCustomSetting

Remarks

Create a custom setting from a WindowsSettingsGroupSetting object. Creates the setting based on the information in the SettingName, SettingDescription and OperatingSystemInformation fields. On exit the id, times and user fields are filled in.

Note

The SettingData field is ignored on input - this is the database representation of the Operating System info which is created by the function.

Declaration

void WindowsSettingsGroup_CreateCustomSetting(
ref profilemanagement.profileservice.datacontracts.WindowsSettingsGroupSetting Setting)

Returns

Parameter is updated with new id

Parameters

Setting	in/out - on input specifies the OSInfo, setting name and description. On output times and setting id are filled in

WindowsSettingsGroup_DeleteCustomSettingById

Remarks

Deletes one or more custom settings by setting id

Declaration

void WindowsSettingsGroup_DeleteCustomSettingById(
profilemanagement.profileservice.datacontracts.IdAndModifiedTimePair[] GroupInfo)

Parameters

GroupInfo	Array of custom settings ids and corresponding modified times for concurrency checking

WindowsSettingsGroup_DeleteGroupsById

Remarks

Deletes one or more windows settings groups by group id

Declaration

void WindowsSettingsGroup_DeleteGroupsById(
profilemanagement.profileservice.datacontracts.IdAndModifiedTimePair[]
GroupInfo)

Parameters

GroupInfo	Array of windows settings group ids and corresponding modified times for concurrency checking

WindowsSettingsGroup_GetAll

Remarks

Returns all windows settings groups

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup[]
WindowsSettingsGroup_GetAll()

Returns

Array of WindowsSettings Groups.

WindowsSettingsGroup_GetAllSettings

Remarks

Returns all defined settings, (custom and fixed)

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroupSetting[]
WindowsSettingsGroup_GetAllSettings()

Returns

Array of WindowsSettingsGroupSetting objects

WindowsSettingsGroup_GetCondition

Remarks

Returns the condition associated with a windows settings group

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsCondition WindowsSettingsGroup_GetCondition(
Guid GroupId)

Returns

Condition object, or null if the group has no assigned condition

Parameters

GroupId	Identity of windows settings group

WindowsSettingsGroup_GetForUserGroup

Remarks

Returns list of windows settings groups assigned to a user group

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup[] WindowsSettingsGroup_GetForUserGroup(
Guid UserGroupId)

Returns

Array of windows settings groups

Parameters

UserGroupId	Identify of user group

WindowsSettingsGroup_GetGroupByDisplayName

Remarks

Returns windows setting group object by display name (i.e. name visible on console).

Note

If a windows settings group is renamed, only the display name changes - the original name stays the same so clients work the same way.

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup
WindowsSettingsGroup_GetGroupByDisplayName(
string DisplayName, int? SourceSystemId)

Returns

Windows Settings group

Parameters

DisplayName	Windows Settings group display name
SourceSystemId	Source system qualifying display name *(10.1 FR2)*

WindowsSettingsGroup_GetGroupById

Remarks

Return windows settings group by its group id

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroup
WindowsSettingsGroup_GetGroupById(
Guid GroupId)

Returns

Windows settings group

Parameters

GroupId	Windows Settings group identity

WindowsSettingsGroup_GetGroupsForSettings

Remarks

Returns a list of windows setting group names associated with one or more windows settings

Declaration

profilemanagement.profileservice.datacontracts.WindowsSettingsGroupsUsingSetting[]
WindowsSettingsGroup_GetGroupsForSettings(
Guid[] Ids)

Returns

Array of WindowsSettingsGroupsUsingSetting objects, one for each windows settingid passed in

Parameters

Ids	Array of windows settings ids

WindowsSettingsGroup_GetUserGroupsForWindowsSettingsGroups

Remarks

Returns a list of user group names associated with one or more windows settings groups.

Declaration

profilemanagement.profileservice.datacontracts.
UserGroupsUsingWindowsSettingsGroup[]
WindowsSettingsGroup_GetUserGroupsForWindowsSettingsGroups(
Guid[] Ids)

Returns

Array of UserGroupsUsingWindowsSettingsGroup objects, one for each windows setting group id passed in

Parameters

Ids	Array of windows setting group ids

WindowsSettingsGroup_RemoveConditionFromGroup

Remarks

Removes the condition associated with a windows settings group.

Declaration

DateTime WindowsSettingsGroup_RemoveConditionFromGroup(
Guid GroupId,
DateTime ModifiedTime,
out bool DeletedCondition)

Returns

New modified time of windows settings group after deletion

Parameters

GroupId	Identity of windows setting group
ModifiedTime	Current modified time of group for concurrency check
DeletedCondition	Flag indicating whether a condition was deleted (false if condition was reusable and had other references)

WindowsSettingsGroup_RemoveFromUserGroup

Remarks

Removes one or more windows settings groups from the specified user group

Declaration

DateTime WindowsSettingsGroup_RemoveFromUserGroup(
Guid UserGroupId,
Guid[] WindowsSettingsGroupIds,
DateTime UserGroupModifiedTime)

Returns

New modified time of user group

Parameters

UserGroupId	Identity of user group
WindowsSettingsGroupIds	Array of windows settings group ids to remove
UserGroupModifiedTime	Current user group modified time for concurrency checking

WindowsSettingsGroup_UpdateCustomSetting

Remarks

Update custom setting from setting object. Note that SettingData is ignored on input - the details of the setting are in the OperatingSystemInformation field.

Declaration

DateTime WindowsSettingsGroup_UpdateCustomSetting(
profilemanagement.profileservice.datacontracts.WindowsSettingsGroupSetting
Setting)

Returns

New modified time of setting.

Parameters

Setting	Setting object. The setting to update is identified by the SettingId field, and the ModifiedTime must match the current value in the database for concurrency checking.

WindowsSettingsGroup_UpdateNameAndDescription

Remarks

Updates the display name and description for a windows settings group

Declaration

void WindowsSettingsGroup_UpdateNameAndDescription(
Guid GroupId,
string Name,
string Description,
ref DateTime ModifiedTime)

Parameters

GroupId	Identity of windows settings group
Name	New name for group
Description	New description for group
ModifiedTime	On input, current modified time of group. On output, new modified time

WindowsSettingsGroup_UpdateSettingsForGroup

Remarks

Assigns all the settings for a windows setting group, removing any existing once

Declaration

void WindowsSettingsGroup_UpdateSettingsForGroup(
Guid GroupId,
Guid[] SettingIds,
ref DateTime ModifiedTime)

Parameters

GroupId	Identity of windows settings group
SettingIds	Array of windows settings ids to be assigned to the group
ModifiedTime	On input, current modified time of group. On output, new modified time

EMPImportExport PowerShell Module

Introduction

The EMPImportExport module is shipped with version 8.5 and later of the Personalization Server. This contains the two PowerShell cmdlets **Import-EMPSDatabase** and **Export-EMPSDatabase**, which can be used to import and export the configuration data (not the user data) from personalization server databases. This is the same functionality as provided by the console import and export functions, and the file formats are exactly the same.

Loading

The module is installed to the personalization server directory *C:\Program Files\AppSense\Environment Manager\Personalization Server\API\Modules* in a standard installation. To load the module from PowerShell enter:

```
PS> Import-Module "C:\Program Files\AppSense\Environment Manager\Personalization Server\API\Modules\EMPImportExport"
```

The module directory is self-contained and may be transported to another computer with PowerShell installed, and will function identically.

Standard command line help is provided, accessible as normal by entering

```
get-help <cmdlet name>
```

Importing the EMPImportExport module automatically loads PSProxy4.dll, giving access to the Connect function as detailed above.

Export-EMPSDatabase Cmdlet

`Export-EMPSDatabase has the following parameters:`

-Host	Name of Personalization Server. Specify the computer name or an IP Address. If the port is non-standard, this can be specified in the host name (e.g. server12:5000), but this is only for compatibility with the PSProxy Connect function. The preferred way to specify the port is with the -TcpPort parameter. Either -Host or -Proxy must be specified to identify the Personalization Server.
-Proxy	Specifies a proxy object created by the PSProxy Connect function to be used to connect to the Personalization Server. Either -Host or -Proxy must be specified.
-Https	Switch parameter - if present cmdlet will use https to connect to the server.
-TcpPort	Port number for the server if non-standard. (Standard values are 80 for http and 443 for https).
-Credential	Specify a PSCredential object identifying a user authorized to use the Personalization Server (this must be an authorized console user, not a web console user). If a username is entered PowerShell will prompt for a password.
-Authentication	Specify type of authentication used by server. Values are Detect, Windows or Anonymous. Detect is the default, whereby the cmdlet tries Windows first, then Anonymous.

-ExportFile	Path to xml file to be created. If file exists it is overwritten (if -Confirm is specified, user is prompted to overwrite). If -ExportFile is omitted, the cmdlet writes the configuration to the PowerShell pipeline as a System.Xml. Linq.XDocument object.
-Applications	Specify one or more applications to be exported to the output. If omitted, all applications are exported. If “none” is specified, no applications are exported. For example: -Applications “Microsoft Word”, “Microsoft Outlook“. Wildcards * are accepted
-ApplicationGroups	Specify one or more application groups to be exported to the output. Syntax is similar to Applications above. If an application group is selected which requires applications not selected above, the referenced applications are automatically re-added and a warning message is displayed.
-PersonalizationGroups	Specify one or more personalization groups to be exported - syntax as for applications.
-Sites	Specify one or more sites to be exported - syntax as for applications.
-CustomWindowsSettings	Specify one or more custom windows settings (as created by the custom settings editor) to be exported. Syntax as for applications
-WindowsSettingsConditions	Specify one or more reusable conditions to be exported. Non-reusable conditions assigned to windows setting groups are automatically exported if the group is exported. Syntax as for applications
-WindowsSettingsGroups	Specify windows setting groups to be exported. Syntax as for applications.
-ExcludedApplications	This controls whether the <GlobalBlacklist> section of the PS configuration is exported. Setting this to “none” excluded this section from the xml.
-Settings	Specify one or more setting types to be exported. Supported types are “Global Settings”, "Global Application Inclusions and Exclusions","none” or “*”.
-PassThru	If specified, the cmdlet writes a copy of its file output to the pipeline as a System.Xml.Linq.XDocument object even if -Path is specified.
-Verbose	Verbose output is produced, confirming which parts of the database are being exported.
-Confirm	Causes the cmdlet to prompt for confirmation if overwriting an existing file.
-WhatIf	Displays a message if a file is to be overwritten but doesn’t overwrite it.
-Debug	Generates verbose output and asks for confirmation after each line of verbose information.

Import-EMPSDatabase Cmdlet

Import-EMPSDatabase has the following parameters:

-Host	As for Export-EMPSDatabase
-Proxy	As for Export-EMPSDatabase
-Https	As for Export-EMPSDatabase
-TcpPort	As for Export-EMPSDatabase
-Credential	As for Export-EMPSDatabase
-Authentication	As for Export-EMPSDatabase
-ImportFile	Path of Xml file to be imported. If omitted, the cmdlet expects to read a System.Xml.Linq.XDocument object from the pipeline
-Applications	As for Export-EMPSDatabase - selects part of the configuration to be imported.
-ApplicationGroups	As for Export-EMPSDatabase
-PersonalizationGroups	As for Export-EMPSDatabase
-CustomWindowsSettings	As for Export-EMPSDatabase
-WindowsSettingsConditions	As for Export-EMPSDatabase
-WindowsSettingsGroups	As for Export-EMPSDatabase
-ExcludedApplications	As for Export-EMPSDatabase
-Sites	As for Export-EMPSDatabase
-Settings	As for Export-EMPSDatabase
-ConflictAction	Specifies action to take if the object being imported already exists in the target database. Possible values are Replace or Merge. This works in a similar way to the console import tool, but can only be applied to the configuration as a whole.
-Purge	If specified, causes any objects in the target database not in the xml configuration to be removed. Existing objects are overwritten so that any user data in the target database is preserved (database GUIDS are maintained). Any existing applications specifying the same executable and version of an application in the imported configuration is renamed if necessary to match the imported configuration. Not compatible with -ConflictAction Merge (an error is displayed)
-Force	Overrides the checksum validation of the xml file. Configuration files are written with a checksum when exported from the console or the Export-EMPSDatabase cmdlet. This detects if a file has been modified in transit. If the file has been legitimately altered this parameter can be used to override the check.
-AllowDefaultOverwrite	Allows an imported configuration to modify default category applications. This might be necessary when importing a configuration from an older version of the database where blacklisted applications added manually are now shipped as defaults. However

	it is recommended that instead the imported configuration should be manually edited to match the target database.
-PassThru	If specified, the cmdlet writes a copy of its input to the output pipeline as a System.Xml.Linq.XDocument object
-Verbose	Verbose output is produced, confirming which parts of the database are being imported.
-Confirm	Causes the cmdlet to prompt for confirmation before writing to the database
-WhatIf	Displays a message when about to write to the database, but doesn't write. Can be used with -Verbose to check action of the cmdlet.
-Debug	Generates verbose output and asks for confirmation after each line of verbose information.

Examples

Refer to the PowerShell help for examples.

Appendix I - Sample Script

The following script deletes ApplicationData rows for all files ending in ".log" in the user group "TestGroup". It uses profile analysis functions to step through each user then each application in the group. For a small database it would be possible to get all users in one go with User_GetAll. Also if only one application was involved the function GetAllApplicationDataByName might be useful.

The function MyFetchReport together with the preceding constants could be useful in other scripts.

The first two lines set the group to search and the extension of the files to be deleted.

```
$GroupToSearch     = "TestGroup" # User group to search
$ExtensionToDelete = ".log"      # File extension to delete


#=======================================
#      Constants for FetchReport2
#=======================================

$reportTypeUsage        = 0
$reportTypeSize         = 1
$reportTypeDiscovered   = 2

$itemTypeApplication    = 0
$itemTypeUser           = 1

$sortOrderDescending    = 0
$sortOrderAscending     = 1

#============================================================
#      MyFetchReport
#   Provides a wrapper around FetchReport2
#   which does the invoke for the caller
#   Returns an array of plot points
#   and a summary to the $summary parameter
#
#   Start by passing $null to $startRank and at least
#   2 to $numRowsToReturn, or rows can be missed in the
#   special case where there is only one user in the
#   group
#============================================================
function MyFetchReport($client,
                       $userGroupName,
                       $reportType,
                       $itemType,
                       $itemName,
                       $startDate = $null,
                       $endDate = $null,
                       $sortOrder,
                       [int] $numRowsToReturn,
                       $startRank = $null,
                       [ref] $summary)
{
    # Build parameter list

    $params = @($userGroupName, $reportType, $itemType, $itemName,
                $startDate, $endDate ,$sortOrder, $numRowsToReturn,
                $null, $null, $null)

    if ($startRank -eq $null)
    {
        $params[8] = $null
```

```
        $params[9] = $null
    }
    else
    {
        $params[8] = [int] $startRank
        $params[9] = $numRowsToReturn
    }

    $report = $client.GetType().GetMethod("FetchReport2").Invoke($client,$Params)

    $summary.Value = $params[10]  # get the 'out' param from the array
    #
    #  Special case where there is only one row:
    #  the rank can come out at "1" when it
    #  should be "2"
    #
    if (($report.Count -eq 1) -and ($params[11].MaxRank -eq 2))
    {
        $report[0].Rank = 2
    }

    $report   # return the report

}

#============================================
#      ProcessAllUsers
#============================================

function ProcessAllUsers($userGroupName)
{
    "Searching group $userGroupName"

    #=====================================================================
    #  Use MyFetchReport to page through the users in the
    #  database.  On a small database User_GetAll might do the trick,
    #  but we assume not in this case
    #=====================================================================

    # Number of users read in a block
    $userBlockSize = 10

    $summary = $null
    $report = @(MyFetchReport -client $pmClient `
                              -userGroupName $userGroupName `
                              -reportType $reportTypeSize `
                              -itemType $itemTypeUser `
                              -itemName $null `
                              -sortOrder $sortOrderDescending `
                              -numRowsToReturn $userBlockSize `
                              -summary ([ref] $summary))

    # Retrieve max rank from summary
    $maxRank = $summary.MaxRank
    $currentUserRank = 2

    while ($true)
    {
        foreach ($item in $report)
        {
            ProcessUser $userGroupName $item.Name
        }

        if ($report.Count -gt 0)
        {
            $currentUserRank = $report[$report.Count-1].Rank + 1
        }
```

```
        if ($currentUserRank -gt $maxRank)
        {
            break
        }
        $report = @(MyFetchReport -client $pmClient `
                        -userGroupName $userGroupName `
                        -reportType $reportTypeSize `
                        -itemType $itemTypeUser `
                        -itemName $null `
                        -sortOrder $sortOrderDescending `
                        -numRowsToReturn $userBlockSize `
                        -startRank $currentUserRank `
                        -summary ([ref] $summary))
    }

}

#=================================================================
#      ProcessUser
#=================================================================
function ProcessUser($userGroupName, $userName)
{
    "    User $userName"

    #=============================================================
    #  Use MyFetchReport again to get all apps for the user
    #=============================================================

    $applicationBlockSize = 10

    $summary = $null
    $report = @(MyFetchReport -client $pmClient `
                              -userGroupName $userGroupName `
                              -reportType $reportTypeSize `
                              -itemType $itemTypeUser `
                              -itemName $userName `
                              -sortOrder $sortOrderDescending `
                              -numRowsToReturn $userBlockSize `
                              -summary ([ref] $summary))

    $maxRank = $summary.MaxRank
    $currentApplicationRank = 2

    while ($true)
    {
        foreach ($item in $report)
        {
            ProcessApplication $userGroupName $userName $item.Name
        }

        if ($report.Count -gt 0)
        {
            $currentUserRank = $report[$report.Count-1].Rank + 1
        }
        if ($currentUserRank -gt $maxRank)
        {
            break
        }
        $report = @(MyFetchReport -client $pmClient `
                                  -userGroupName $userGroupName `
                                  -reportType $reportTypeSize `
                                  -itemType $itemTypeUser `
                                  -itemName $userName `
                                  -sortOrder $sortOrderDescending `
                                  -numRowsToReturn $userBlockSize `
                                  -startRank  $currentUserRank `
                                  -summary ([ref] $summary))
```

```
    }

}

#=========================================================
#  ProcessApplication
#=========================================================
function ProcessApplication($userGroupName, $userName, $applicationName)
{
    "       Application $applicationName"

    $ApplicationProfilePK = new-object System.Guid;

    $appData = $pmClient.GetAllApplicationDataByName($userName,
                                                     $applicationName,
                                                     $null,
                                                     0,
                                                     [ref] $ApplicationProfilePK)
    if ($appData -ne $null)
    {
        foreach ($file in $appData)
        {
            if ($file.Filename.EndsWith($ExtensionToDelete))
            {
                $fn = $file.Filename
                "           File to delete: $fn"

                $result =
                    $pmClient.DeleteApplicationDataByname($userName,
                                                          $applicationName,
                                                          $file.RelativePath,
                                                          $file.Filename,
                                                          0)
                if ($result.ErrorCode -eq 0)
                {
                    $fullfile = $file.RelativePath + "\" + $file.Filename
         "Successfully deleted $fullfile from $userName/$applicationName"
                }
            }

        }
    }
}

#=====================================================================
#  Main program
#=====================================================================

$ErrorActionPreference = "stop"

#   Load proxy for net 4 (powershell 3)
import-module "$env:ProgramFiles\AppSense\Environment Manager\Personalization
Server\API\PSProxy4.dll"

#   Connect to local host
$pmClient = [PSProxy]::Connect("localhost")

ProcessAllUsers $GroupToSearch

$pmClient.Close()
```