AVEVA™

Application Server

2023 R2 SP1

aveva.com

Contents

GRAccess Toolkit

The AVEVA™ GRAccess object model provides a way to automate configuration of local and remote Galaxies. The Galaxy configuration is exposed to .NET and COM clients through a programmable object mode.

How to use the GRAccess Toolkit

You can use the AVEVA™ GRAccess object model to write programs that automate configuration of local and remote Galaxies. The Galaxy configuration is exposed to .NET and COM clients through a programmable object model called GRAccess.

Target clients

C# is the recommended programming language for GRAccess. GRAccess also works with other COM or .NET programming languages such as VB6, C++, and VB.NET.

GRAccess provides a .NET primary interop assembly for use in Microsoft .NET languages, and a COM type library that can be used in VB6 or other COM clients.

Microsoft Visual Studio

Microsoft Visual Studio is required. The version you need depends on the applications you will be creating:

- Microsoft Visual Studio 2012 Update 1 or later is required for .NET applications
- Microsoft Visual Studio 2019 is required for C++ applications.

You must run with administrative privileges in order to build the projects using Visual Studio or run an application (.exe) built using the GRAccess Toolkit.

Configuration files

When you create a GRAccess application, a configuration file for application should be created in the same location as the GRAccess application (EXE). The configuration file name should be in the format:
<ApplicationName>.exe.config

Add the following xml into the file, under:
configuration > runtime > assemblyBinding

```
<configuration>
<startup useLegacyV2RuntimeActivationPolicy="true">
<requiredRuntime version="v2.0.50727" safemode="true"/>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.1"/>
</startup>
<runtime>
<assemblyBinding>
```

```
<dependentAssembly>
<assemblyIdentity name="ArchestrA.GRAccess"
publicKeyToken="23106a86e706d0ae"
culture=""/>
<bindingRedirect oldVersion="1.0.0.0-1.9.0.0" newVersion="2.0.0.0"/>
</dependentAssembly>
</assemblyBinding>
</runtime>
</configuration>
```

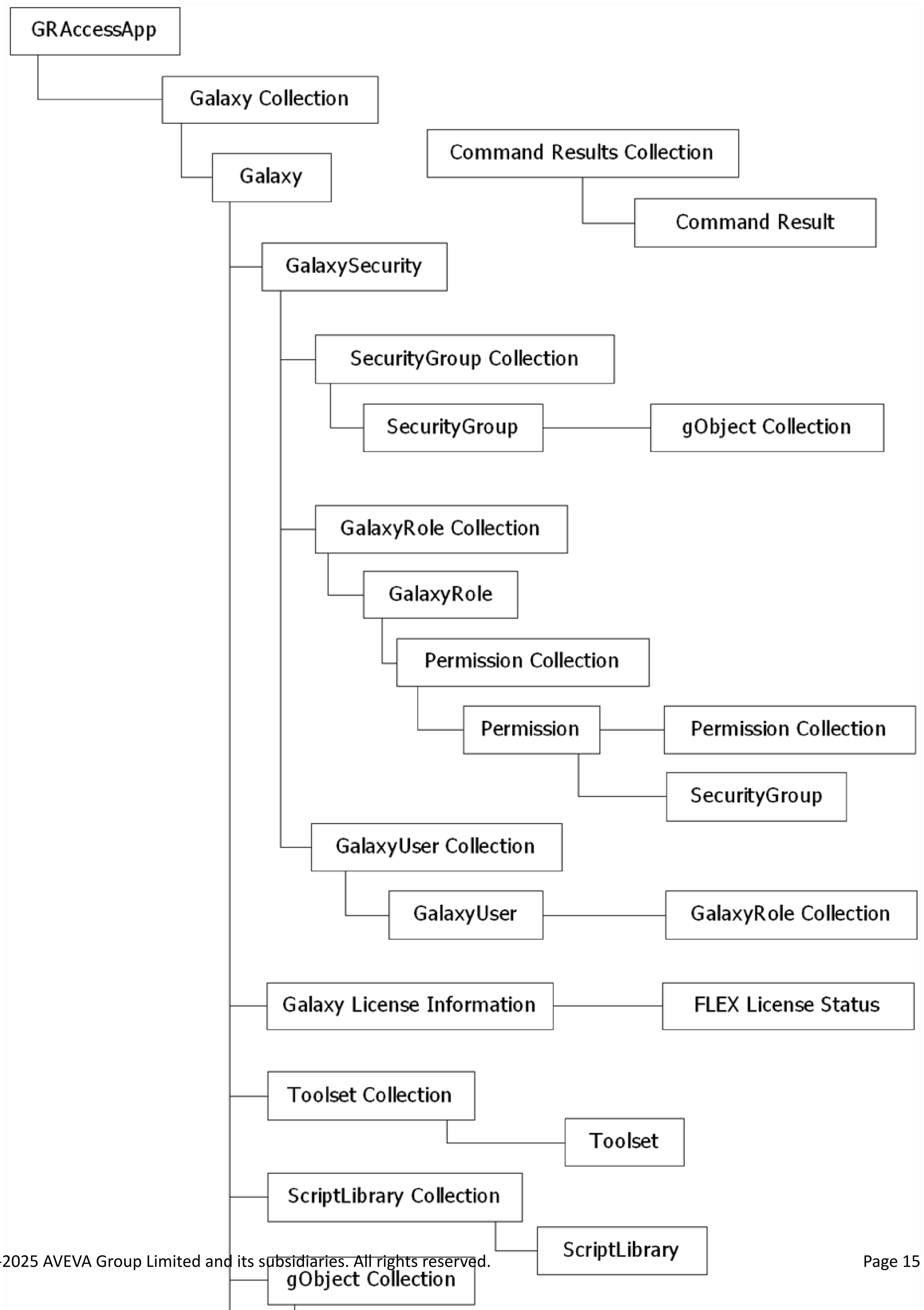
For more information about assembly binding, refer to:

https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/redirect-assembly-versions#specifying-assembly-binding-in-configuration-files

GRAccess object model

GRAccess provides a hierarchical object model. At the root of this object model is the GRAccessAppClass. The GRAccessAppClass provides access to other ArchestrA objects including Galaxies, gObject Templates and Instances.

Typically, only the GRAccessAppClass and the MxValue objects are created directly. Other objects are obtained by navigating the GRAccessAppClass object object model.


GRAccessApp
Galaxy Collection
Galaxy
Command Results Collection
Command Result
GalaxySecurity
SecurityGroup Collection
SecurityGroup
gObject Collection
GalaxyRole Collection
GalaxyRole
Permission Collection
Permission
Permission Collection
SecurityGroup
GalaxyUser Collection
GalaxyUser
GalaxyRole Collection
Galaxy License Information
FLEX License Status
Toolset Collection
Toolset
ScriptLibrary Collection
ScriptLibrary
gObject Collection

GRAccess methods to set license mode

Galaxies can be configured to use one of two license modes: either Flex, which requires an annual subscription, or perpetual, which is a traditional one-time purchase of a software license.

Flex licenses are applied to platform objects. The Flex licenses are single-engine or multi-engine, meaning that a Flex-configured platform object is allowed to have either a single AppEngine assigned to it, or if a multi-engine is applied, the licensing does not constrain the number of AppEngines.

GRAccess contains two methods to support Flex licenses programmatically.

- **PlatformLicenseType:** This is a IgObject property that can be used to retrieve or update (get/set) the Flex license type of a platform.
- **FlexLicensesInformation:** This is an IGalaxy property that can be used to obtain the status, including type and number, of Flex licenses that are available to the Galaxy Repository. Status is returned as an array and includes:
  - License type: single engine or multi-engine
  - Number of licenses available to the GR
  - Number of licenses deployed by the GR
  - Number of licenses configured for the GR
  - Number of licenses available for deployment by the GR

API reference

You can use the GRAccess API to write programs that manipulate the configurations of local and remote Galaxies.

COM classes

This section describes the COM classes that are defined and exposed through GRAccess. These include:

- GRAccess
- GRAccessApp
- MxValue

GRAccess

GRAccess is similar to GRAccessApp but is cocreated in-process to the client. Restrictions include:

- EXE program hosting the component must reside in the AppServer Bin folder.
- Single-stepping over GRAccess method calls with the Visual Basic 6.0 debugger may cause execution to run-away and not stop at the next line.

Interfaces implemented

IgObject

GRAccessApp

Clients use the GRAccess Component to gain access to the Galaxy Repository.

Interfaces implemented

IgObject

MxValue

The MxValue provides storage and retrieval of System Platform data types.

MxValues are passed as parameters in the AutomationObject Get and Set methods.

Interfaces implemented

IMxValue

Interfaces

This section describes the COM interfaces that are defined and exposed through GRAccess. These interfaces include:

Objects	Collections of
IAssociatedFileAccess	
IAttribute	IAttributes
ICommandResult	ICommandResults
ICondition	IConditions
IGalaxy	IGalaxies
IGalaxyRole	IGalaxyRoles
IGalaxySecurity	
IGalaxyUser	IGalaxyUsers
IGraphicAccess	
IInstance	

IMxValue	
IgObject	IgObjects
IPermission	IPermissions
IScriptLibrary	IScriptLibraries
ISecurityGroup	ISecurityGroups
ISettings	
ITemplate	
IToolset	IToolsets

IAssociatedFileAccess

Internal use only.

IAttribute

Represents a single attribute of an object.

IAttribute class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult Object, which has the last method call's result.

RtSethandler property

Returns the RtSethandler property of the Attribute.

CfgSethandler property

Returns the CfgSethandler property of the Attribute.

SecurityClassification property

Returns the SecurityClassification property of the Attribute.

Name property

Returns the Name property of the Attribute.

DataType property

Returns the Type property of the Attribute.

SetLocked method

Sets the Locked property of the Attribute.

UpperBoundDim1 property

Returns the UpperBoundDim1 property of the Attribute.

SetSecurityClassification method

Sets the SecurityClassification property of the Attribute.

Locked property

Returns the Locked property of the Attribute.

SetValue method

Sets the Value of the Attribute.

Value property

Returns the Value of the Attribute.

AttributeCategory property

Returns the Category of the Attribute.

HasBuffer property

Returns the HasBuffer property of the Attribute.

SetHasBuffer method

Sets the HasBuffer property of the Attribute.

CommandResult property

Returns the CommandResult object, which has the last method call’s result.

Class

IAttribute

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

RtSethandler property

Returns the RtSethandler property of the attribute.

Class

IAttribute

Syntax

[C#]

```
bool RtSethandler { get; };
```

[Visual C++]

```
HRESULT RtSethandler(
   [out, retval] VARIANT_BOOL* hasRtSethandler
);
```

Parameters

hasRtSethandler

Returns the RtSethandler property of the attribute.

CfgSethandler property

Returns the CfgSethandler property of the attribute.

Class

IAttribute

Syntax

[C#]

```
bool CfgSethandler { get; };
```

[Visual C++]

```
HRESULT CfgSethandler(
   [out, retval] VARIANT_BOOL* hasCfgSethandler
);
```

Parameters

hasCfgSethandler

Returns the CfgSethandler property of the attribute.

SecurityClassification property

Returns the SecurityClassification property of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxSecurityClassification SecurityClassification { get; };
```

[Visual C++]

```
HRESULT SecurityClassification(
   [out, retval] MxSecurityClassification* securityClassification
);
```

Parameters

securityClassification

Returns the SecurityClassification property of the attribute.

Name property

Returns the name property of the attribute.

Class

IAttribute

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* name
);
```

Parameters

name

Returns the name of the attribute.

DataType property

Returns the type property of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxDataType DataType { get; };
```

[Visual C++]

```
HRESULT DataType(
   [out, retval] MxDataType* dataType
);
```

Parameters

dataType

Returns the type property of the attribute.

SetLocked method

Sets the locked property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetLocked(
MxPropertyLockedEnum NewMxPropertyLockedEnum
);
```

[Visual C++]

```
HRESULT SetLocked(
   [in] MxPropertyLockedEnum
);
```

Parameters

MxPropertyLockedEnum

The new locked property of the attribute.

UpperBoundDim1 property

Returns the UpperBoundDim1 property of the attribute.

Class

IAttribute

Syntax

[C#]

```
short UpperBoundDim1 { get; };
```

[Visual C++]

```
HRESULT UpperBoundDim1(
   [out, retval] short* upperBoundDim1
);
```

Parameters

upperBoundDim1

Returns the UpperBoundDim1 property of the attribute.

SetSecurityClassification method

Sets the SecurityClassification property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetSecurityClassification(
MxSecurityClassification NewMxSecurityClassification
);
```

[Visual C++]

```
HRESULT SetSecurityClassification(
   [in] MxSecurityClassification
);
```

Parameters

MxSecurityClassification

The new SecurityClassification property of the attribute.

Locked property

Returns the Locked property of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxPropertyLockedEnum Locked { get; };
```

[Visual C++]

```
HRESULT Locked(
   [out, retval] MxPropertyLockedEnum*
);
```

Parameters

MxPropertyLockedEnum

Returns the Locked property of the attribute.

SetValue method

Sets the value of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetValue(
MxValue NewValue
);
```

[Visual C++]

```
HRESULT SetValue(
   [in] IMxValue* NewValue
);
```

Parameters

NewValue

The new value of the attribute.

Value property

Returns the value of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxValue value { get; };
```

[Visual C++]

```
HRESULT Value(
   [out, retval] IMxValue** value
);
```

Parameters

value

Returns the value of the attribute.

AttributeCategory property

Returns the category of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxAttributeCategory AttributeCategory { get; };
```

[Visual C++]

```
HRESULT AttributeCategory(
   [out, retval] MxAttributeCategory* attributeCategory
);
```

Parameters

attributeCategory

Returns the category of the attribute.

HasBuffer property

Returns the HasBuffer property of the attribute.

Class

IAttribute

Syntax

[C#]

```
bool HasBuffer { get; };
```

[Visual C++]

```
HRESULT HasBuffer(
   [out, retval] VARIANT_BOOL* hasBuffer
);
```

Parameters

hasBuffer

Returns the HasBuffer property of the attribute.

SetHasBuffer method

Sets the HasBuffer property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetHasBuffer(bool hasBuffer);
```

[Visual C++]

```
HRESULT SetHasBuffer(
   [in] VARIANT_BOOL hasBuffer
);
```

Parameters

hasBuffer

The new value of the attribute.

IAttributes

Represents a collection of attribute objects.

IAttributes class members

This class has the following members.

Operations

Item property

Returns the Attribute object by name or index.

Count property

Returns the number of attributes in the collection.

ShortDescription property

Returns the short description for this object.

ExecutionOrder property

Returns the name of the execution order of this object.

SecurityGroup property

Returns the securitygroup for this object.

ExecutionRelatedObject property

Returns the name of the execution related object of this object.

Item property

Return the attribute object by name or index.

Class

IAttributes

Syntax

[C#]

```
IAttribute this[object attributeIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT attributeIdentifier,
   [out, retval] IAttribute**
);
```

Parameters

attributeIdentifier

The name of the attribute object to return if a string is specified. If a number is specified, it is interpreted as the index into this collection and is in the range from 1 to count.

IAttribute

The returned attribute object or nothing if attribute doesn't exist.

Count property

Returns the number of attributes in the collection.

Class

IAttributes

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

The number attributes in the collection.

ShortDescription property

Returns the short description of this object.

Class

IAttributes

Syntax

[C#]

```
string ShortDescription { set; get; };
```

[Visual C++]

```
[propget]
```

```
HRESULT ShortDescription(
   [out, retval] BSTR*
);
[propput]
HRESULT ShortDescription(
   [in] BSTR
);
```

ExecutionOrder property

Returns the name of the execution order of this object.

Class

IAttributes

Syntax

[C#]

```
EExecutionOrder ExecutionOrder { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ExecutionOrder(
   [out, retval] EExecutionOrder*
);
[propput]
HRESULT ExecutionOrder(
   [in] EExecutionOrder
);
```

SecurityGroup property

Retrieves the security group for this object.

Class

IAttributes

Syntax

[C#]

```
string SecurityGroup SecurityGroup { set; get; };
```

[Visual C++]

```
[propget]
HRESULT SecurityGroup(
   [out,retval] BSTR*
);
[propget]
```

```
HRESULT SecurityGroup(
   [in] BSTR*
);
```

ExecutionRelatedObject property

Returns the name of the execution-related object of this object.

Class

IAttributes

Syntax

[C#]

```
string ExecutionRelatedObject { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ExecutionRelatedObject(
   [out, retval] BSTR*
);
[propput]
HRESULT ExecutionRelatedObject(
   [in] BSTR newExecutionRelatedObject
);
```

Parameters

newExecutionRelatedObject

The name of the execution-related object.

ICommandResult

Indicates the result of a given operation (for instance, success, fail and why).

ICommandResult Class Members

This class has the following members.

Operations

Successful property

Indicates whether the operation was successful.

Text property

Textual description corresponding to the result code.

ID property

Reason code for the result.

CustomMessage property

Custom message associated with the result.

Successful property

Indicates whether the operation was successful.

Class

ICommandResult

Syntax

[C#]

```
bool Successful { get; };
```

[Visual C++]

```
HRESULT Successful( [out, retval] VARIANT_BOOL* );
```

Text property

Textual description corresponding to the result code.

Class

ICommandResult

Syntax

[C#]

```
string Text { get; }
```

[Visual C++]

```
HRESULT Text( [out, retval] BSTR* );
```

ID property

Reason code for the result.

Class

ICommandResult

Syntax

[C#]

```
EGRCommandResult ID { get; };
```

[Visual C++]

```
HRESULT ID( [out, retval] EGRCommandResult* );
```

CustomMessage property

Custom message associated with the result.

Class

ICommandResult

Syntax

[C#]

```
string CustomMessage { get; };
```

[Visual C++]

```
HRESULT CustomMessage( [out, retval] BSTR* );
```

ICommandResults

Represents a collection of CommandResult objects.

ICommandResults class members

This class has the following members.

Operations

Count property

Returns the number of CommandResults in the collection.

Item method

Returns the CommandResult object by index.

CompletelySuccessful property

Indicates that all CommandResult objects in the collection are successful.

Count property

Returns the number of CommandResult objects in the collection.

Class

ICommandResults

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Item method

Returns the CommandResult object by index.

Class

ICommandResults

Syntax

[C#]

```
ICommandResult get_Item(
int index
);
```

[Visual C++]

```
HRESULT Item(
   [in] LONG index,
   [out, retval] ICommandResult** customResult
);
```

Parameters

index

The Index of the CommandObject. This is a numeric value from 1 to count.

customResult

The returned CommandResult object.

CompletelySuccessful property

Indicates that all CommandResult objects in the collection are successful.

Class

ICommandResults

Syntax

[C#]

```
bool CompletelySuccessful { get; };
```

[Visual C++]

```
HRESULT CompletelySuccessful(
   [out, retval] VARIANT_BOOL* completelySuccessful
);
```

Parameters

completelySuccessful

Indicates that all CommandResult objects in the collection are successful.

ICondition

Indicates a condition used when querying for a list of objects.

ICondition class members

This class has the following members.

Operations

Kind property

Indicates the type of condition to be used by the query.

Value property

The value to be searched for. Meaning is based on the *Kind* parameter.

Negation property

Negation. Specify False to "NOT" the expression.

Kind property

Indicates the type of condition to be used by the query.

Class

ICondition

Syntax

[C#]

```
EConditionType Kind { get; };
```

[Visual C++]

```
HRESULT Kind( [out, retval] EConditionType* Kind );
```

Value property

The value to be searched for. Meaning is based on the Kind parameter.

Class

ICondition

Syntax

[C#]

```
object value { get; };
```

[Visual C++]

```
HRESULT Value( [out, retval] VARIANT* Value );
```

Negation property

Negation. Specify False to "NOT" the expression.

Class

ICondition

Syntax

[C#]

```
EConditionType Negation { get; };
```

[Visual C++]

```
HRESULT Negation([out, retval] VARIANT_BOOL* Value );
```

IConditions

Represents a collection of condition objects.

IConditions class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult Object, which has the last method call's result.

Remove method

Removes a condition object from the collection.

Add method

Adds a condition object to the collection.

Count property

Returns the number of condition objects in the collection.

Item property

Returns the condition object by index.

Join method

Joins two condition objects.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IConditions

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Remove method

Remove a condition object from the collection.

Class

IConditions

Syntax

[C#]

```
void Remove(
   int index
);
```

[Visual C++]

```
HRESULT Remove(
   [in] LONG index
);
```

Parameters

index

The index of the condition object. This is a numeric value from 1 to count.

Add method

Add a condition object to the collection.

Class

IConditions

Syntax

[C#]

```
void Add(
   EConditionType Kind,
   object value,
   bool Negation
);
```

[Visual C++]

```
HRESULT Add(
   [in] EConditionType Kind,
   [in] VARIANT Value,
   [in, optional, defaultvalue( -1 )]
   VARIANT_BOOL negation
);
```

Parameters

Kind

Indicates the type of condition to be used by the query.

Value

The value to be searched for. Meaning is based on the *Kind* parameter.

negation

Negation. To NOT the expression, pass False for negation.

Count property

Returns the number of condition objects in the collection.

Class

IConditions

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Item property

Returns the condition object by index.

Class

IConditions

Syntax

[C#]

```
ICondition this[int index] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] LONG index,
   [out, retval] ICondition**
);
```

Parameters

index

The index of the condition object. This is a numeric value from 1 to Count.

ICondition

The returned condition object.

Join method

Join two conditions.

Class

IConditions

Syntax

[C#]

```
void Join(
   IConditions conditions
);
```

[Visual C++]

```
HRESULT Join(
   [in] IConditions* conditions
);
```

Parameters

conditions

The returned condition object.

IGalaxy

Represents a Galaxy.

IGalaxy class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult object, which has the last method call's result.

QueryObjectsMultiCondition method

Retrieves a gObjects collection using multiple conditions.

QueryScriptLibraries method

Query Script Libraries.

Name property

Retrieves the name of the Galaxy.

Backup method

Backs-up the given Galaxy to a backup file.

VersionString property

Retrieves the version string of the Galaxy.

Login method

Logs in to an ArchestrA Galaxy and performs a forced synchronization of the client with the Galaxy Repository. This method must be called before any other Galaxy method.

LoginEx method

Logs in to an ArchestrA Galaxy and offers the option to perform synchronization at a later point in time and cancel synchronization. This method must be called before any other Galaxy method.

LoginWithAccessMode method

Logs into a Galaxy. This method must be called before any other Galaxy method. If GRAccess is already logged into a Galaxy, the session is logged off and restablished with the specified Galaxy.

SynchronizeClient method

Synchronizes the client with the server.

GetTimeMasterSettings method

Retrieves TimeMaster settings.

CreategObjectCollection method

Creates an empty gObjects collection object

ImportObjects method

Imports objects into the Galaxy.

ImportObjectsEx method

Imports objects into the Galaxy.

ImportObjectsEx2 method

Imports one or more of the objects contained in an aaPDF or aaPKG file into the Galaxy Repository.

ImportArchestrAApp method

Imports AVEVA OMI apps from a specified folder.

QueryToolsets method

Retrieves the list of available Toolsets.

QueryObjectsByName method

Retrieves a gObjects collection based on a set of tagnames.

GetUserDefaults method

Get the logged in user's defaults from the Galaxy repository.

VersionNumber property

Retrieves the version number of the Galaxy.

SetUserDefaults method

Set the logged in user's defaults.

Restore method

Restores a Galaxy from a backup file.

Logout method

Logs out from an ArchestrA Galaxy.

UpgradeRequired property

Retrieves the Boolean indicated if the Galaxy needs to be upgraded.

MigrateGalaxy method

Upgrade a Galaxy.

CdiVersionString property

Retrieves the version string of the Cdi.

GetSecuritySettings method

Retrieves the Galaxy's security settings.

CreateConditionsObject method

Creates a condition object used with QueryObjectsMultiCondition.

QueryObjects method

Retrieves a gObjects collection based on the given criteria.

ImportScriptLib method

Imports this script library.

ImportScriptLibEx method

Imports a script library (aaSLib file).

GetLocaleSettings method

Retrieves the Galaxy's local settings.

ExportAll method

Exports all objects in the Galaxy.

GetReadOnlySecurity method

Accesses the Galaxy security in read-only mode.

CommandResults property

Returns a collection of CommandResult objects.

GRLoad method

Loads the Galaxy from a CSV file.

GetFlexLicensesInformation method

Returns a collection of Flex license information for single and multi-engine platforms.

IGalaxyQueryToolsetsEx method

Retrieves the list of available Toolsets.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IGalaxy

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

QueryObjectsMultiCondition method

Retrieves a gObjects collection using multiple conditions.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjectsMultiCondition(
   EgObjectIsTemplateOrInstance templateOrInstance, IConditions conditions
);
```

[Visual C++]

```
HRESULT QueryObjectsMultiCondition(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in] IConditions* Conditions,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Conditions

A condition collection object containing a list of conditions to be ANDed together.

IgObjects

Returns a gObjects collection.

QueryScriptLibraries method

Query Script Libraries.

Class

IGalaxy

Syntax

[C#]

```
IScriptLibraries QueryScriptLibraries();
```

[Visual C++]

```
HRESULT QueryScriptLibraries(
   [out, retval] IScriptLibraries** scriptLibraries
);
```

Parameters

scriptLibraries

The list of script libraries.

Name property

Retrieves the name of the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* Galaxy
);
```

Parameters

Galaxy

Returns the name of the Galaxy.

Backup method

Backs up the given Galaxy to a backup file.

Class

IGalaxy

Syntax

[C#]

```
void Backup(
   int lProcessId,
   string backupFilename,
   string GRNodeName,
   string Galaxy
);
```

[Visual C++]

```
HRESULT Backup
(
   [in]LONG lProcessId,
   [in] BSTR backupFilename,
   [in, optional] BSTR GRNodeName,
   [in, optional] BSTR Galaxy
);
```

Parameters

lProcessId

Proccess ID of the calling process.

backupFilename

Name of the backup file. File extension .cab is appended if not specified. If a folder name is specified, the contents of the folder is deleted and the backup files are placed directly in the folder.

GRNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

Galaxy

The name of the Galaxy (application database).

VersionString property

Retrieves the version string of the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
string VersionString { get; };
```

[Visual C++]

```
HRESULT VersionString(
   [out, retval] BSTR* VersionString
);
```

Parameters

VersionString

Returns the version string of the Galaxy.

Login method

Logs in to an ArchestrA Galaxy and performs a forced synchronization of the client with the Galaxy Repository. This method must be called before any other Galaxy method.

Class

IGalaxy

Syntax

[C#]

```
void Login(
   string UserName, string Password
);
```

[Visual C++]

```
HRESULT Login(
   [in, optional] BSTR UserName,
   [in, optional] BSTR Password
);
```

Parameters

UserName

The name of the user.

- When **Galaxy** Authentication mode is configured, blank or omitted *UserName* means log in as "admin".
- When **OS User based** or **OS Group based** Authentication mode is configured, *UserName* is ignored and the current logged in user or current user group membership is used to authenticate the user.

Password

The user's password.

- When **Galaxy** Authentication mode is configured, blank or omitted *Password* means use a blank password.
- When **OS User based** or **OS Group based** Authentication mode is configured, *Password* is ignored and the current logged in user or current user group membership is used to authentication the user.

Remarks

If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy.

LoginEx method

Logs in to an ArchestrA Galaxy and offers the option to perform synchronization at a later point in time and cancel synchronization. This method must be called before any other Galaxy method.

Class

IGalaxy

Syntax

[C#]

```
void LoginEx(
   string UserName, string Password, bool bForceSynchronization
);
```

[Visual C++]

```
HRESULT LoginEx(
   [in, optional] BSTR UserName,
   [in, optional] BSTR Password,
   [in, optional] VARIANT_BOOL bForceSynchronization
);
```

Parameters

UserName

The name of the user.

- When **Galaxy** Authentication mode is configured, blank or omitted means log in as "admin".
- When **OS User based** or **OS Group based** Authentication mode is configured, *UserName* is ignored and the current logged in user or current user group membership is used to authenticate the user.

Password

The user's password.

- When **Galaxy** Authentication mode is configured, blank or omitted means use a blank password.
- When **OS User based** or **OS Group based** Authentication mode is configured, *Password* is ignored and the current logged in user or current user group membership is used to authenticate the user.

bForceSynchronization

Set to true to synchronize the client with the server if out of sync is detected.

Remarks

If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy.

LoginWithAccessMode method

Logs into a Galaxy. This method must be called before any other Galaxy method. If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy. This API does the following:

- Passes the Package Operation Mode while logging in.
- Avoids the need for GRAccess to log in to the Galaxy while configuring Security through the System Platform IDE. This method calls the IGalaxyBase::Login method.

Class

IGalaxy

Syntax

[C#]

```
void LoginWithAccessMode(string UserName], string Password, int bModeReadWrite = 0);
```

[Visual C++]

```
HRESULT LoginWithAccessMode
   (
      [in, optional] BSTR Username,
      [in, optional] BSTR Password,
      [in, defaultvalue(FALSE)] BOOL bModeReadWrite
   );
```

Parameters

Username

Name of the user. If Username is blank/omitted, log in as “admin.”

Password

Password associated with the Username. If password is blank/omitted, password is none.

bModeReadWrite

The Package Operation Mode (read/write mode) to be applied to the user.

0: Indicates Read/Write mode (default).

1: Indicates Read-Only mode.

Blank/omitted: Read-Only mode.

SynchronizeClient method

Synchronizes the client with the server.

Class

IGalaxy

Syntax

[C#]

```
void SynchronizeClient()
```

[Visual C++]

```
HRESULT SynchronizeClient();
```

Remarks

User should be logged in.

GetTimeMasterSettings method

Retrieves TimeMaster settings.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetTimeMasterSettings( );
```

[Visual C++]

```
HRESULT GetTimeMasterSettings(
   [out, retval] ISettings**
);
```

Parameters

ISettings

Returns a gObjects collection object.

Remarks

Automatically check-out the instance for the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

CreategObjectCollection method

Creates an empty gObjects collection object.

Class

IGalaxy

Syntax

[C#]

```
IgObjects CreategObjectCollection( );
```

[Visual C++]

```
HRESULT CreategObjectCollection(
   [out, retval] IgObjects** gObjects
);
```

Parameters

gObjects

The newly created gObjects object.

ImportObjects method

Imports objects into the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void ImportObjects(
   string inputFile,
   bool OverwritesAllowed
);
```

[Visual C++]

```
HRESULT ImportObjects(
   [in] BSTR inputFile,
   [in, optional, defaultvalue( 0 )]
   VARIANT_BOOL OverwritesAllowed
);
```

Parameters

inputFile

The name of the input file -- can be aaPDF or aaPKG file. aaPDF is base Template file format and aaPKG is exported objects file format.

OverwritesAllowed

Indicates whether existing objects can be overwritten.

Remarks

User should be logged in.

ImportObjectsEx method

Class

IGalaxy

Syntax

[C#]

```
void ImportObjectsEx(
   string inputFile,
   E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   string appendToObjectName
);
```

[Visual C++]

```
HRESULT ImportObjectsEx(
   // The name of the input file -- can be a PDF or //.csv file.
   [in] BSTR inputFile,
   [in] E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   [in] E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   [in] BSTR appendToObjectName
);
```

ImportObjectsEx2 method

Imports one or more of the objects contained in an aaPDF or aaPKG file into the Galaxy Repository.

Class

IGalaxy

Syntax

[C#]

```
void ImportObjectsEx2
(
   string inputFile,
   ArchestrA.GRAccess.E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   ArchestrA.GRAccess.E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   string appendToObjectName, ArchestrA.GRAccess.E_RESOLVE_PROTECTION_CONFLICT_ACTION
   eProtectionConflict
);
```

[Visual C++]

```
HRESULT ImportObjectsEx2
   (
      [in] BSTR inputFile,
      [in] E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
      [in] E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
      [in] BSTR appendToObjectName,
      [in] E_RESOLVE_PROTECTION_CONFLICT_ACTION eProtectionConflict
   );
```

Parameters

inputFile

Name of the file to be imported. The file can be PDF or .csv format.

Returns:

S_OK: Import successful

E_UNEXPECTED: The IImportExport file content is not formatted as aaPDF or aaPKG content. See Notes, below.

E_FAIL: Unable to add objects to the Galaxy Repository

Notes

E_RESOLVE_PROTECTION_CONFLICT_ACTION eProtectionConflict is supported for handling protection conflicts.

The System Platform IDE gets a reference to IImportExport by calling the GetIImportExport() method on IGalaxyConfiguration.

ImportArchestrAApp method

Imports AVEVA OMI apps from a specified folder.

Class

IGalaxy

Syntax

[C#]

```
void ImportArchestrAApp(string appPath);
```

[Visual C++]

```
HRESULT ImportArchestrAApp
   (
      [in] BSTR appPath
   );
```

Parameters

appPath

String that sets the path to the folder containing the AVEVA OMI app(s).

QueryToolsets method

Retrieves a collection object containing the available Toolsets.

Class

IGalaxy

Syntax

[C#]

```
IToolsets QueryToolsets( );
```

[Visual C++]

```
HRESULT QueryToolsets(
   [out, retval] IToolsets** toolsets
);
```

Parameters

toolsets

The Toolsets collection object.

Remarks

User should be logged in.

QueryObjectsByName method

Retrieves a gObjects collection based on a set of tagnames.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjectsByName(
   EgObjectIsTemplateOrInstance templateOrInstance, ref string[] tagnames
);
```

[Visual C++]

```
HRESULT QueryObjectsByName(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in, out] SAFEARRAY(BSTR)* Tagnames,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Tagnames

An array of tagnames. Any tagname found in the Galaxy will be included in the returned collection.

IgObjects

Returns a gObjects collection object.

Remarks

User should be logged in.

Note: If a tagname is not found, it is simply not included in the returned collection object.

GetUserDefaults method

Get the logged in user's defaults from the Galaxy repository.

Class

IGalaxy

Syntax

[C#]

```
string GetUserDefaults(
   EUserDefault userDefault
);
```

[Visual C++]

```
HRESULT GetUserDefaults(
   [in] EUserDefault userDefault,
   [out, retval] BSTR* defaultValue
);
```

Parameters

userDefault

The default.

defaultValue

The value of the default.

VersionNumber property

Retrieves the version number of the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
int VersionNumber { get; };
```

[Visual C++]

```
HRESULT VersionNumber(
   [out, retval] LONG* VersionNumber
);
```

Parameters

VersionNumber

Returns the version number of Galaxy.

SetUserDefaults method

Set the logged in user's defaults.

Class

IGalaxy

Syntax

[C#]

```
void SetUserDefaults(
   EUserDefault userDefault,
   string defaultValue
);
```

[Visual C++]

```
HRESULT SetUserDefaults(
   [in] EUserDefault userDefault,
   [in] BSTR defaultValue
);
```

Parameters

userDefault

The default.

defaultValue

The value of the default.

Restore method

Restores a Galaxy from a backup file.

Class

IGalaxy

Syntax

[C#]

```
void Restore(
   int lProcessId,
   string backupFilename,
   string GRNodeName,
   string Galaxy,
   bool bRestoreOlderVersion
);
```

[Visual C++]

```
HRESULT Restore(
   [in]LONG lProcessId,
   [in] BSTR backupFilename,
   [in, optional] BSTR GRNodeName,
   [in, optional] BSTR Galaxy,
   [in, optional] VARIANT_BOOL bRestoreOlderVersion
);
```

Parameters

lProcessId

Proccess ID of the calling process.

backupFilename

Name of the backup file. File extension cab is appended if not specified. If a folder name is specified, the Galaxy is restored from the contents of the folder.

GRNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

Galaxy

The name of an exiting Galaxy (application database) into which the backed up Galaxy is to be restored into.

bRestoreOlderVersion

Indicates whether to restore the database even if backup is an older version. If omitted, this field is set to false indicating an older database that should not be restored.

Remarks

User should not be logged in.

Logout method

Logs out from an ArchestrA Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void Logout( );
```

[Visual C++]

```
HRESULT Logout( );
```

Remarks

Logs out from an ArchestrA Galaxy.

UpgradeRequired property

Retrieves a Boolean that indicates whether the Galaxy needs to be upgraded.

Class

IGalaxy

Syntax

[C#]

```
bool UpgradeRequired { get; };
```

[Visual C++]

```
HRESULT UpgradeRequired(
   [out, retval] VARIANT_BOOL* vtUpgradeRequired
);
```

Parameters

vtUpgradeRequired

Returns a Boolean indicating whether the Galaxy needs to be upgraded.

MigrateGalaxy method

Upgrade a Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void MigrateGalaxy(
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT MigrateGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName
);
```

Parameters

galaxyName

The name of the Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

CdiVersionString property

Retrieves the version string of the Cdi.

Class

IGalaxy

Syntax

[C#]

```
string CdiVersionString { get; }
```

[Visual C++]

```
HRESULT CdiVersionString(
   [out, retval] BSTR* VersionString
);
```

Parameters

VersionString

Returns the version string of Cdi.

GetSecuritySettings method

Retrieves the security settings.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetSecuritySettings( );
```

[Visual C++]

```
HRESULT GetSecuritySettings(
   [out, retval] ISettings** settings
);
```

Parameters

settings

Returns a gObjects collection.

Remarks

Automatically checks-out the instance of the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

CreateConditionsObject method

Creates a conditions object used with QueryObjectsMultiCondition.

Class

IGalaxy

Syntax

[C#]

```
IConditions CreateConditionsObject( );
```

[Visual C++]

```
HRESULT CreateConditionsObject(
   [out, retval] IConditions** conditions
);
```

Parameters

conditions

The newly created conditions object.

QueryObjects method

Retrieves gObjects collection based on the given criteria.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjects(
   EgObjectIsTemplateOrInstance templateOrInstance,
   EConditionType Kind,
   object value,
   EMatch MatchCondition
);
```

[Visual C++]

```
HRESULT QueryObjects(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in] EConditionType Kind,
   [in] VARIANT Value,
   [in, optional, defaultvalue( MatchCondition )] EMatch MatchCondition,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Kind

Indicates the type of condition used by the query.

Value

The value to be searched for. Meaning is based on the Kind parameter.

MatchCondition

To negate the condition, pass False.

IgObjects

Returns a gObjects collection.

Remarks

User should be logged in.

ImportScriptLib method

Import a script library.

Class

IGalaxy

Syntax

[C#]

```
void ImportScriptLib(
   string path
);
```

[Visual C++]

```
HRESULT ImportScriptLib(
   [in] BSTR path
);
```

Parameters

path

Path to the .aaslib file.

ImportScriptLibEx method

Imports a Script Library (aaSLib file).

Class

IGalaxy

Syntax

[C#]

```
void ImportScriptLibEx(string path, ArchestrA.GRAccess.EImportDLLOptions options);
```

[Visual C++]

```
HRESULT ImportScriptLibEx
   (
      [in] BSTR path,
      [in] EImportDLLOptions options
   );
```

Parameters

path

Filepath to the Script Library file (.aaSLib).

EImportDLLOptions

Override options. The overrides will block the import of a DLL if it is deployed to a run-time node.

- **BlockIfDLLIsDeployed = 0:** Block if dependencies exist; allow unconditionally if there are none.
- **AllowImportIfDLLIsDeployed = 1:** Allow import unconditionally.
- **AllowImportIfDLLIsDeployedAndHasLowerVersionInGalaxy = 2:** Allow import even if dependencies exist,

provided that the version of the DLL in the Galaxy is less than that which is being imported.

- **BlockImportIfImportVersionNotHigherThanVersionInGalaxy = 3:** Block import if dependencies exist, OR if Import DLL version is less than or equal to Galaxy DLL version.

GetLocaleSettings method

Retrieves locale settings.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetLocaleSettings( );
```

[Visual C++]

```
HRESULT GetLocaleSettings(
   [out, retval] ISettings**
);
```

Parameters

ISettings

Returns a gObjects collection.

Remarks

Automatically check-out the instance for the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

ExportAll method

Exports all objects in the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void ExportAll(
   EExportType exportType,
   string outputFile
);
```

[Visual C++]

```
HRESULT ExportAll(
   [in] EExportType exportType,
   [in] BSTR outputFile
);
```

Parameters

exportType

Indicates whether to export to aaPKG file.

outputFile

The name of the output file. Appends extension if one is not specified.

Remarks

User should be logged in.

GetReadOnlySecurity method

Accesses the Galaxy security in read-only mode. The user can not modify the Galaxy security.

Class

IGalaxy

Syntax

[C#]

```
IGalaxySecurity GetReadOnlySecurity( );
```

[Visual C++]

```
HRESULT GetReadOnlySecurity(
   [in] IGalaxySecurity** galaxySecurity
);
```

Parameters

galaxySecurity

IGalaxySecurity provides an API interface to read the Galaxy security information configured using the security dialog box in aaIDE.

CommandResults property

Returns the CommandResults object.

Class

IGalaxy

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

GRLoad method

Class

IGalaxy

Syntax

[C#]

```
void GRLoad(
   string CSVFileNameWithPath,
   GRLoadMode GRLoadMode
);
```

[Visual C++]

```
HRESULT GRLoad(
   [in, string] LPCOLESTR CSVFileNameWithPath,
   [in] GRLoadMode grLoadMode
);
```

GetFlexLicensesInformation method

Returns an array that contains FLEX licenses available to the Galaxy Repository. The array includes the license type, the number of available licenses, number of deployed licenses, number of configured licenses, and number available for deployment (undeployed but available licenses).

One array is returned for each license type (single-engine platform licenses and multiple-engine platform licenses).

Class

IGalaxy

Syntax

[C#]

```
FLEXLICENSESINFORMATION[] GetFlexLicensesInformation();
```

[Visual C++]

```
HRESULT GetFlexLicensesInformation
(
[out, retval]SAFEARRAY(struct FLEXLICENSESINFORMATION)* response
);
struct FLEXLICENSESINFORMATION
{
int licenseType;
int availableToGR;
int deployed;
int configured;
int availableForDeploy;
}FLEXLICENSESINFORMATION;
```

Parameters

licenseType

Integer that defines the FLEX License Type:

- 1 = single engine platform
- 2 = multiple-engine platform

Each license type outputs an array that lists the number of licenses in each the following categories(number available to GR, number deployed, number configured, number available). There will be one array for single engine platform licenses, and one array for multiple engine licenses.

availableToGR

Total number of licenses (single or multiple) available for the Galaxy Repository to use.

deployed

Number of licenses (single or multiple) deployed from the Galaxy Repository.

configured

Number of licenses (single or multiple) that have been configured in the AVEVA™ License Manager.

availableForDeploy

Number of licenses (single or multiple) that are available to be deployed (total available minus number already deployed).

Example

```
// create GRAccessAppClass object
GRAccessApp grAccess = new GRAccessAppClass();
// Query for galaxies
IGalaxies gals = grAccess.QueryGalaxies(nodeName);
if (gals == null || grAccess.CommandResult.Successful == false)
{
```

```
Console.WriteLine(grAccess.CommandResult.CustomMessage + grAccess.CommandResult.Text);
return;
}
IGalaxy galaxy = gals[galaxyName];
FLEXLICENSESINFORMATION[] flex = galaxy.GetFlexLicensesInformation();
if (flex != null && flex.Length > 0)
{
Console.WriteLine(
"LicenceType:\t\t" + ((flex[0].licenceType == 1) ? "Single AppEngine":"Unlimited
AppEngine") + "\n" +
"AvailableToGR:\t\t" + flex[0].availableToGR.ToString() + " \n" +
"Deployed:\t\t" + flex[0].deployed.ToString() + "\n" +
"Configured:\t\t" + flex[0].configured.ToString() + “\n" +
"AvailableForDeploy:\t" + flex[0].availableForDeploy.ToString() + ");
}
for(int i= 0; i < flex.Length; i++)
{
Console.WriteLine(
"LicenceType:\t\t" + ((flex[i].licenceType == 1) ? "Single AppEngine" : "Unlimited
AppEngine") + "\n" +
"AvailableToGR:\t\t" + flex[i].availableToGR.ToString() + " \n" +
"Deployed:\t\t" + flex[i].deployed.ToString() + "\n" +
"Configured:\t\t" + flex[i].configured.ToString() + "\n" +
"AvailableForDeploy:\t" + flex[i].availableForDeploy.ToString());
Console.WriteLine("============================================================");
}
```

IGalaxyQueryToolsetsEx method

Class

IGalaxy

Syntax

[C#]

```
IToolsets QueryToolsetsEx(
   FolderType eFolderType
);
```

[Visual C++]

```
HRESULT QueryToolsetsEx
(
   // The list of toolsets.
   [in] enum FolderType eFolderType,
   [out, retval] IToolsets** toolsets
);
```

IGalaxies

Represents a collection of Galaxies.

IGalaxies class members

This class has the following members.

Operations

Item property

Return a Galaxy

Count property

Returns the number of Galaxy objects in the collection.

Item property

Return a toolset.

Class

IGalaxies

Syntax

[C#]

```
IGalaxy this[object GalaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT GalaxyIdentifier,
   [out, retval] IGalaxy**
);
```

Parameters

GalaxyIdentifier

The index of a Galaxy. This is a numeric value from 1 to count, or the name of the Galaxy.

IGalaxy

The returned Galaxy object.

Count property

Returns the number of Galaxy objects in the collection.

Class

IGalaxies

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

IGalaxyRole

Accesses the role object and its properties, including role name, access level, list of general permissions, and operational permissions.

IGalaxyRole class members

This class has the following members.

Operations

RoleName property

Returns the name of the Galaxy role.

AccessLevel property

Returns the access level of the Galaxy role.

Permissions property

Returns the general permissions of the Galaxy role.

OperationalPermissions property

Returns the operational permissions of the Galaxy role.

RoleName property

Returns the name of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
string RoleName { get; };
```

[Visual C++]

```
HRESULT RoleName(
   [out,retval]BSTR *roleName
);
```

AccessLevel property

Returns the access level of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
int AccessLevel { get; };
```

[Visual C++]

```
HRESULT AccessLevel(
   [out,retval]LONG *accessLevel
);
```

Permissions property

Returns the general permissions of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
IPermissions Permissions { get; };
```

[Visual C++]

```
HRESULT Permissions(
   [out,retval]IPermissions **generalPermissions
);
```

OperationalPermissions property

Returns the operational permissions of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
IPermissions OperationalPermissions { get; };
```

[Visual C++]

```
HRESULT OperationalPermissions(
   [out,retval]IPermissions **operationalPermissions
);
```

IGalaxyRoles

Returns information about the roles in the Galaxy.

IGalaxyRoles class members

This class has the following members.

Operations

Item property

Returns a Galaxy role.

Count property

Returns the number of roles in the collection.

Item property

Return a Galaxy role.

Class

IGalaxyRoles

Syntax

[C#]

```
IGalaxyRole this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IGalaxyRole** GalaxyRole
);
```

Parameters

galaxyIdentifier

The index of a role. This is a numeric value from 1 to count, or the name of the role.

GalaxyRole

The returned Galaxy role object.

Count property

Returns the number of rolesin the collection.

Class

IGalaxyRoles

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

Returns the number of roles in the collection.

IGalaxySecurity

Accesses top-level security information and Galaxy security objects, including security groups, roles, and users configured for the Galaxy.

IGalaxySecurity class members

This class has the following members.

Operations

AuthenticationMode property

Retrieves the authentication mode of the Galaxy.

SecurityGroupsAvailable property

Retrieves the available security groups of the Galaxy.

RolesAvailable property

Retrieves the collection of roles configured for this Galaxy.

UsersAvailable property

Retrieves the collection of users configured for this Galaxy.

LoginTime property

Retrieves the configured login time.

RoleUpdateInterval property

Retrieves role update interval time.

AuthenticationMode property

Retrieves the authentication mode of the Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
EAUTHMODE AuthenticationMode { get; };
```

[Visual C++]

```
HRESULT AuthenticationMode(
   [out,retval]EAUTHMODE *authMode
);
```

Parameters

AuthenticationMode

Indicates whether Galaxy security mode is none, Galaxy, OS User, or OS Group.

SecurityGroupsAvailable property

Retrieves the available security groups of the Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
ISecurityGroups SecurityGroupsAvailable { get; };
```

[Visual C++]

```
HRESULT SecurityGroupsAvailable(
   [out,retval]ISecurityGroups** securityGroups
);
```

Parameters

SecurityGroupsAvailable

The collection of security groups configured for this Galaxy.

RolesAvailable property

Retrieves the collection of roles configured for this Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
IGalaxyRoles RolesAvailable { get; };
```

[Visual C++]

```
HRESULT RolesAvailable(
   [out,retval]IGalaxyRoles** roles
);
```

Parameters

roles

The collection of roles configured for this Galaxy.

UsersAvailable property

Retrieves the collection of users configured for this Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
IGalaxyUsers UsersAvailable { get; };
```

[Visual C++]

```
HRESULT UsersAvailable(
   [out,retval]IGalaxyUsers** users
);
```

Parameters

users

The collection of users configured for this Galaxy.

LoginTime property

Retrieves the configured login time.

Class

IGalaxySecurity

Syntax

[C#]

```
string LoginTime { get; };
```

[Visual C++]

```
HRESULT LoginTime(
   [out,retval]BSTR* loginTime
);
```

Parameters

LoginTime

The timeout period during which the system validates the user’s membership against the OS groups selected as ArchestrA roles. Minimum value is 0 (zero), maximum is 9,999,999. Default value is 0 (zero), which turns off this feature so the operation does not time out. Specify a value based on the speed of your network and the number of groups configured in ArchestrA roles. The slower the network or the larger the number of groups, the greater the value.

RoleUpdateInterval property

Retrieves role update interval time.

Class

IGalaxySecurity

Syntax

[C#]

```
string RoleUpdateInterval { get; };
```

[Visual C++]

```
HRESULT RoleUpdateInterval(
   [out,retval]BSTR* roleUpdateInterval
);
```

Parameters

RoleUpdateInterval

The time between each validation attempt per OS group for the user's membership when a logon is attempted. The user membership update is done one role per Role Update interval to minimize network usage. The minimum allowed value is 0 (zero) and the maximum is 9,999,999. The default value is 0 (zero), which turns off this feature so the operation does not pause between validating user membership and groups. This option operates independently of the Login Time option. Even if Login Time times out, the role update operation continues in the background and eventually updates user-to-role relationships for this user in the local cache.

IGalaxyUser

Provides user information: user name, full name, and associated role list.

If you are using OS group authentication mode, you can not configure Galaxy users. During login from client utilities, user information is updated in the Galaxy, which can then be obtained using the IGalaxyUser interface.

The FullName property is only valid when a user is an OS (domain or local) user.

IGalaxyUser class members

This class has the following members.

Operations

UserName property

Returns the user name along with the domain name in the form *<domain_name>**<user_name>*.

FullName property

Returns the full name of the user retrieved from the domain controller in the form *<first_name> <last_name>*.

AssociatedRoles property

Returns collection of roles associated with this user.

UserName property

Returns the user name along with the domain name in the form *<domain_name>**<user_name>*.

Class

IGalaxyUser

Syntax

[C#]

```
string UserName { get; };
```

[Visual C++]

```
HRESULT UserName(
   [out,retval]BSTR* *userName
);
```

FullName property

Returns the full name of the user retrieved from the domain controller in the form *<first_name> <last_name>*.

Class

IGalaxyUser

Syntax

[C#]

```
string FullName { get; };
```

[Visual C++]

```
HRESULT FullName(
   [out,retval]BSTR* *fullName
);
```

AssociatedRoles property

Returns collection of roles associated with this user.

Class

IGalaxyUser

Syntax

[C#]

```
IGalaxyRoles AssociatedRoles { get; };
```

[Visual C++]

```
HRESULT AssociatedRoles(
   [out,retval]IGalaxyRoles **associatedRoles
);
```

IGalaxyUsers

Represents a collection of Galaxy users.

IGalaxyUsers class members

This class has the following members.

Operations

Item property

Returns a Galaxy user.

Count property

Returns the number of roles in the collection.

Item property

Return a Galaxy user.

Class

IGalaxyUsers

Syntax

[C#]

```
IGalaxyUser this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IGalaxyUser** galaxyUser
);
```

Parameters

galaxyIdentifier

The index of a user. This is a numeric value from 1 to count, or the name of the user.

galaxyUser

The returned Galaxy user.

Count property

Returns the number of users in the collection.

Class

IGalaxyUsers

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* UserCount
);
```

Parameters

UserCount

Returns the number of users in the collection.

IGraphicAccess

The set of GraphicAccess application programming interfaces (API) allow users to export graphics to an XML file programmatically. Use the same set of APIs to create an Industrial Graphic in another galaxy or to overwrite an existing graphic(s) by importing the graphic definition from an XML file. The APIs can also be used to export the configured references or GPI for a graphic and importing multiple XML files from a folder location.

Before using the set of GraphicAccess APIs, you must use the GRAccess Toolkit to log in and connect to an Galaxy repository.

The APIs are available through the following interfaces:

- IGraphicAccess – Export and import an Industrial Graphic
- IGraphicAccess2 – Export the substitutable strings and configured references for an Industrial Graphic
- IGraphicAccess3 – Export the GPI of an Industrial Graphic
- IGraphicAccess4 – Import Industrial Graphics from a folder location

About Programmatic Graphic Export and Import

The GraphicAccess programmatic API exports or imports an extensive set of properties of an Industrial graphic. For standard Industrial Graphics, an exported or imported graphic can contain the following:

- Custom properties
- Configured References

- Graphic elements
- Connector lines
- Graphic groups
- Graphic animations
- Element styles
- Named scripts
- Predefined scripts
- Overridden text strings
- Numeric format styles
- DataStatus elements
- Trend Pen
- Multi Pens Trend
- Alarm Client
- Trend Client

In addition to the properties of a standard Industrial graphic, an exported or imported Symbol Wizard can contain the following:

- Wizard Options
- Choice groups
- Choices
- Layers
- Rules

The IGraphicAccess API

The GraphicAccess API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

IGraphicAccess is used to export an Industrial Graphic to an XML file, or to import an XML file into a Galaxy as an Industrial Graphic. IGalaxy points to the Galaxy where the import/export operation is performed.

```
public interface IGraphicAccess
{
/// <summary>Export an Industrial Graphic to XML file</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath);
/// <summary>Import the XML file and generate an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <param name="bOverWrite">Flag which indicates if existing graphic should be
overwritten</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ImportGraphicFromXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bOverWrite);
void ValidateSymboltoExport(IGalaxy galaxy, string graphicName);
}
public interface IGraphicAccessResult : ICommandResult
{
XMLOperations.StatusCode Status { get; }
}
```

If the Galaxy has security enforced, you must first log in to the Galaxy Repository with proper credentials. Also, the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions must be set active before attempting an import or export operation.

You can set the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions from the IDE **Configure Security** dialog box. For more information about setting Galaxy permissions, see the Application Server Help.

GraphicAccess interface methods

The IGraphicAccess interface includes separate graphic export and import methods.

ExportGraphicToXml method

Exports an Industrial Graphic to an XML file. Galaxy, graphic name, and the XML file path are passed as parameters to the ExportGraphicToXml function.

Syntax

```
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy containing the graphic to export.

graphicName

Name of the graphic to export.

xmlFilePath

Directory folder to place the XML file containing the exported graphic.

ImportGraphicFromXml method

Imports a graphic from an existing XML file. Galaxy, graphic name, the XML file path, and an overwrite flag are passed as parameters to the ImportGraphicfromXml function.

Syntax

```
IGraphicAccessResult ImportGraphicFromXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bOverWrite);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy where the graphic will be imported.

graphicName

Name of the graphic to import.

xmlFilePath

Directory folder location of the XML file.

bOverWrite

Boolean flag that indicates if an existing graphic can be overwritten by an imported graphic with the same name.

ValidateSymboltoExport method

Validates the graphic before exporting it to an XML file. For more information, see *Validate Objects in Application Server Help*.

Syntax

```
void ValidateSymboltoExport(IGalaxy galaxy, string graphicName);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy that contains the graphic that requires validation.

graphicName

Name of the graphic to validate.

After an import or export operation is complete, the results are set to IGraphicAccessResult which is derived from ICommandResult. A message appears and indicates if the operation succeeded or failed. For detailed information about the ICommandResult interface, see the *GRAccess Toolkit API User's Guide*.

A succeeded message only means the import or export operation finished successfully. It does not indicate the quality of the exported XML file or the imported graphic. Check the SMC log file for any warning or error messages after each export or import operation.

About programmatic string and reference export and import

The GraphicAccess2 API exports all of a graphic's substitutable strings to a specified XML file, regardless of whether some of the strings have been overridden. All graphic strings that appear from a **Ctrl+L** or **Substitute Strings** operation from within the Graphic Editor will be exported, allowing for bulk editing of the graphic strings.

The GraphicAccess2 API can also be used to export a list of configured references for a selected object. The list is exported to a XML string. The API will export the following configured references:

- Mx references
- InTouch tags, including those configured from the argument of attribute

  Example: attribute("InTouch:zhmi-abc")
- Custom properties configured with animation links
- Client scripts
- Custom property overrides
- Client controls
- All applicable references configured within an embedded graphic, regardless of the number of times it is embedded

The IGraphicAccess2 API

The GraphicAccess2 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

IGraphicAccess2 is used to export all of a graphic's substitutable strings to a specified XML file, regardless of whether some of the strings have been overridden.

The GraphicAccess2 API can also be used to export a list of configured references for a selected graphic to an XML string.

IGraphicAccess2 passes galaxy, graphicName, the XML file path, and substitute strings as parameters.

```
public interface IGraphicAccess2 : IGraphicAccess
{
/// <summary>Export an Industrial Graphic to XML file</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <param name="bExportSubstituteStrings">Option to export SubstituteStrings</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bExportSubstituteStrings);
/// <summary>Get references configured on an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="sConfiguredReferencesXML">Returns all the references in XML format</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult GetConfiguredReferences(IGalaxy galaxy, string graphicName, out string
sConfiguredReferencesXML);
}
```

If the Galaxy has security enforced, you must first log in to the Galaxy Repository with proper credentials. Also, the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions must be set active before attempting an import or export operation.

You can set the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions from the IDE **Configure Security** dialog box. For more information about setting Galaxy permissions, see the Application Server Help.

GraphicAccess2 interface methods

The GraphicAccess2 interface uses separate methods to export substitutable strings and configured references for an Industrial Graphic.

ExportGraphicToXml method

Exports all substitutable strings in an Industrial Graphic to an XML file. Galaxy, graphic name, the XML file path and Export Substitute Strings option are the parameters for the ExportGraphictoXML function. This API will not export empty strings and strings used in invisible graphics.

Syntax

```
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bExportSubstituteStrings);
```

Parameters

galaxy

IGalaxy obtained from the GRAccess and returns the galaxy.

graphicName

Name of the graphic to export.

xmlFilePath

Directory folder to place the XML file containing the exported graphic.

bExportSubstituteStrings

When set 'True' will include the substitute string from graphic in the export XML file.

GetConfiguredReferences method

Exports all of the configured references within an Industrial Graphic to an XML file.

Syntax

```
IGraphicAccessResult GetConfiguredReferences(IGalaxy galaxy, string graphicName, out string
sConfiguredReferencesXML);
```

Parameters

galaxy

IGalaxy obtained from the GRAccess and returns galaxy containing the graphic object.

graphicName

Name of the graphic with references to export.

sConfiguredReferencesXML

The configured references for the graphic will be returned to this parameter in a string.

About Programmatic GPI Export

The IGraphicAccess3 API exports the Graphics Performance Index (GPI) rating for a graphic and related details to a specified XML file. The details are as follows:

Item	**Relevance**	**Example**
Description	Component type	Number of Trend Pens
Count	Number of items comprising the component type	3
Impact Score	Call up time in milliseconds for the component type	2.9

The IGraphicAccess3 API

The IGraphicAccess3 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll, located in: \Program Files (x86)\ArchestrA\Framework\Bin

The IGraphicAccess3 API is used to export the Graphics Performance Index (GPI) of the Industrial Graphic.

IGraphicAccess3 passes galaxy, graphicName and the XML file path as parameters.

```
public interface IGraphicAccess3 : IGraphicAccess2
{
/// <summary>Get GPI XML of an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
///<param name="graphicName">The graphic name</param>
///<param name="sGPIXML">Returns GPI details in XML format</param>
/// <returns>Result of the method</returns
ICommandResult GetGPI (IGalaxy galaxy, string graphicName, out string sGPIXML);
}
```

GraphicAccess3 interface methods

GetGPI Method

Exports the GPI of a graphic to an XML file.

Syntax

```
ICommandResult GetGPI (IGalaxy galaxy, string graphicName, out string sGPIXML);
```

Parameters

galaxy

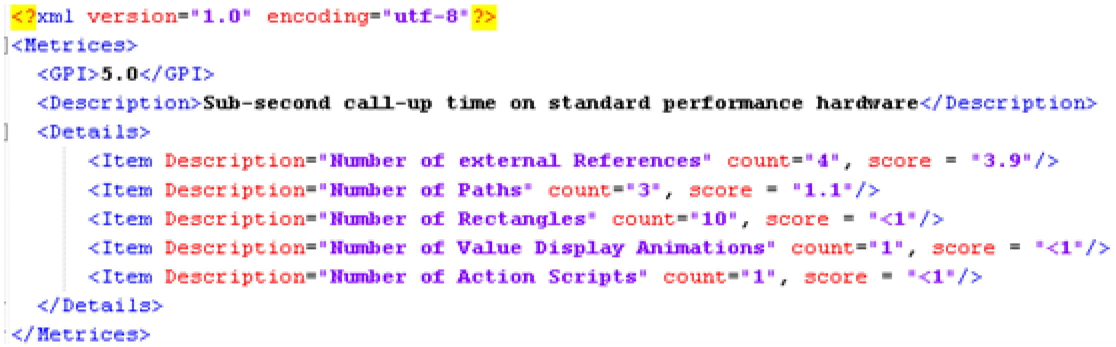
IGalaxy obtained from the GRAccess and returns the galaxy.

graphicName

Name of the graphic for which the GPI is needed.

sGPIXML

The GPI information (XML) for the graphic will be returned to this parameter in a string.



```
<?xml version="1.0" encoding="utf-8"?>
<Metrices>
  <GPI>5.0</GPI>
  <Description>Sub-second call-up time on standard performance hardware</Description>
  <Details>
      <Item Description="Number of external References" count="4", score = "3.9"/>
      <Item Description="Number of Paths" count="3", score = "1.1"/>
      <Item Description="Number of Rectangles" count="10", score = "<1"/>
      <Item Description="Number of Value Display Animations" count="1", score = "<1"/>
      <Item Description="Number of Action Scripts" count="1", score = "<1"/>
  </Details>
</Metrices>
```

About Programmatic Import from a Folder Location

The GraphicAccess4 programmatic API and can import Industrial Graphics from multiple XML files available in a folder.

Implementing the IGraphicAccess4 API

The GraphicAccess4 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

IGraphicAccess4 is used to import graphic XML files. IGalaxy points to a Galaxy where the import operation is performed.

```
public interface IGraphicAccess4 : IGraphicAccess3
{
/// <summary>Import the XML files from the folder and generate Industrial
Graphics</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="strFolderPath">The folder/directory path where graphic XML files are
present</param>
/// <param name="bOverWrite">Flag which indicates if existing graphic should be
overwritten</param>
/// <param name="opNotifier">Returns the status of the import operation to the calling
function (client)</param>
/// <param name="cancelToken">Sets the token which indicates if the operation needs to be
canceled during import</param>
/// <returns>Result of the method</returns>
Task<IGraphicAccessResult> ImportGraphics(IGalaxy galaxy, string strFolderPath, bool
bOverWrite, IGraphicOperationStatus opNotifier = null, CancellationToken cancelToken =
default);
}
```

The IGraphicAccess4 uses the IGraphicOperationStatus callback interface to determine if the operation is in progress, cancelled or completed.

```
namespace ArchestrA.Visualization.GraphicAccess
{
public interface IGraphicOperationStatus
{
void OnOperationProgressMessage(string message);
void OnOperationComplete();
void OnOperationCancel();
}
}
```

GraphicAccess4 Interface Methods

ImportGraphics Method

Imports graphics from existing XML files available in a folder. Galaxy, folder path, an overwrite flag, progress of the operation and a token to determine if the operation is cancelled are passed as parameters to the ImportGraphics function.

Syntax

```
Task<IGraphicAccessResult> ImportGraphics(IGalaxy galaxy, string strFolderPath, bool
bOverWrite, IGraphicOperationStatus opNotifier = null, CancellationToken cancelToken =
default);
```

Parameters

galaxy

Points to a galaxy to which graphics will be imported or where the API will be run.

strFolderPath

Directory folder location of the XML files.

bOverWrite

Boolean flag that indicates if an existing graphic can be overwritten by an imported graphic with the same name.

opNotifier

Returns the status of the import operation to the calling function (client) *cancelToken*

Calling function (Client) can set this token to cancel the import graphics operation. For more information on CancellationToken, see the MSDN help.

After the import operation is complete, the results are set to Task<IGraphicAccessResult> which is derived from ICommandResult. A message appears and indicates if the operation succeeded or failed. For detailed information about the ICommandResult interface, see the G*RAccess Toolkit API Help*.

A succeeded message only means the import operation finished successfully. It does not indicate the quality of the exported XML file or the imported graphic. Check the SMC log file for any warning or error messages after each import operation.

IgObject

Represents a gObject that could be either a template or an instance.

IgObject class members

This class has the following members.

Operations

Tagname property

Returns or sets the tagname of this object.

SortOrder property

Returns or sets the sort order of this object.

ContainedName property

Returns or sets the contained name of this object (for example, Inlet).

HierarchicalName Property

Returns the hierarchical name of this object (for example, Reactor.Tank.Inlet).

Alias property

Returns or sets the alias of this object.

Attributes property

The collection of attributes of this object.

ConfigurableAttributes property

The collection of attributes of this object that can be configured.

ValidationStatus property

The validation status of this object based on the last time the object was validated and saved.

Errors property

The list of errors that was generated the last time the object was validated and saved.

Warnings property

The list of warnings that was generated the last time the object was validated and saved.

CheckedOutBy property

The name of the user that this object is checked out by.

CheckoutStatus property

The check-out status of the object.

ConfigVersion Property

The configuration version of the object.

Category property

The category of the object.

DerivedFrom property

Returns the name of the template that this object was created from.

BasedOn property

The name of the base template (root ancestor) of this object.

Container property

Returns or sets the name of the container object, or blank if object is not contained.

Area property

Returns or sets name of the area object, or blank if the object does not have an area.

Host property

Returns or sets the name of the host for this object, or blank if unassigned.

GetObjectHelpURL method

Returns the URL where the object’s help is stored in the Galaxy Repository.

Save method

Saves the object after it is configured.

CheckIn method

Checks in an object.

CheckOut method

Checks out an object.

UndoCheckOut method

Undoes check-out of this object.

Unload method

Unloads the gObject cache.

EditStatus property

The edit status of the object.

AddExtensionPrimitive method

Add an extension primitive.

DeleteExtensionPrimitive method

Delete an extension primitive.

RenameExtensionPrimitive method

Rename an object extension primitive.

AddUDA method

Add a UDA.

DeleteUDA method

Delete a UDA by its name.

RenameUDA method

Renames a UDA.

UpdateUDA method

Update a UDA.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

CategoryGUID property

Returns the Category GUID of the object.

GetExtendedAttributes method

Returns the Extended Attributes in the hierarchy.

PlatformLicenseType property

Returns or sets the FLEX licensing state.

Tagname property

Returns or sets the tagname of this object.

Class

IgObject

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Tagname(
   [out, retval] BSTR* theTagname
);
[propput]
HRESULT Tagname(
   [in] BSTR newTagName
);
```

Parameters

theTagname

The current tagname of this object.

newTagName

The new tagname of this object. The object does not need to be checked out. The operation is immediate.

ContainedName property

Sets or returns the contained name of this object (for example, Inlet).

Class

IgObject

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ContainedName(
   [out, retval] BSTR* theContainedName
);
[propput]
HRESULT ContainedName(
   [in] BSTR newContainedName
);
```

Parameters

theContainedName

The current contained name of this object.

newContainedName

The new contained name of this object.

HierarchicalName Property

Returns the hierarchical name of this object (for example, Reactor.Tank.Inlet).

Class

IgObject

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
HRESULT HierarchicalName(
   [out, retval] BSTR* theHierarchicalName
);
```

Parameters

theHierarchicalName

The current hierarchical name of this object.

Attributes property

Returns the collection of attributes of this object.

Class

IgObject

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
HRESULT Attributes(
   [out, retval] IAttributes** theAttributes
);
```

Parameters

theAttributes

The collection of attributes of this object.

ConfigurableAttributes property

The collection of this object's attributes that can be configured. Configuring attributes requires that the object be checked out.

Class

IgObject

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
HRESULT ConfigurableAttributes(
   [out, retval] IAttributes** theConfigurableAttributes
);
```

Parameters

theConfigurableAttributes

The collection of attributes that can be configured.

ValidationStatus property

The validation status of this object based on the last time the object was validated and saved.

Class

IgObject

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Parameters

theValidationStatus

The validation status of the object.

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

IgObject

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
HRESULT Errors(
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

Parameters

theErrors

The list of errors.

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

IgObject

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
HRESULT Warnings(
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

Parameters

theWarnings

The list of warnings.

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

IgObject

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
HRESULT CheckedOutBy(
   [out, retval] BSTR* CheckedOutByUserName
);
```

Parameters

CheckedOutByUserName

The user’s name.

CheckoutStatus property

Returns the check out status of the object.

Class

IgObject

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

Parameters

theCheckoutStatus

The check-out status.

ConfigVersion Property

Returns the configuration version number of the object . The configuration version is incremented each time the object is checked in.

Class

IgObject

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

```
HRESULT ConfigVersion(
   [out, retval] LONG* theConfigVersion
);
```

Parameters

theConfigVersion

The configuration version.

Category property

Returns the category of the object.

Class

IgObject

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
HRESULT Category(
   [out, retval] ECATEGORY* theCategory
);
```

Parameters

theCategory

The category.

DerivedFrom property

Returns the name of the template that this object was created from.

Class

IgObject

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
HRESULT DerivedFrom(
   [out, retval] BSTR* theParentTemplate
);
```

Parameters

theParentTemplate

The name of the object's parent template.

BasedOn property

Returns the name of the base template (root ancestor) of this object.

Class

IgObject

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
HRESULT BasedOn(
   [out, retval] BSTR* theBaseTemplate
);
```

Parameters

theBaseTemplate

The name of the object's base template.

Container property

Returns or sets the name of the container object, or blank if object is not contained.

Class

IgObject

Syntax

[C#]

```
string Container { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Container(
   [out, retval] BSTR* theContainer
[propput]
HRESULT Container(
   [in] BSTR NewContainer
);
```

Parameters

theContainer

The name of the current container object, or blank if object is not contained.

NewContainer

The new name of the container, or blank if uncontained. If this object is an ApplicationObject, its area and host are also updated. The object does not need to be checked out. The operation is immediate.

Area property

Returns or sets the name of the Area object, or blank if the object does not have an area.

Class

IgObject

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
[propput]
HRESULT Area(
   [in] BSTR NewArea
);
```

Parameters

theArea

The name of the current Area object, or blank if the object doesn't have an area.

NewArea

The new name of the area, or blank to unassign the object from its existing area.

Host property

Returns or sets the name of the host for this object, or blank if unassigned.

Class

IgObject

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Host(
   [out, retval] BSTR* theHost
);
[propput]
HRESULT Host(
   [in] BSTR NewHost
);
```

Parameters

theHost

The current name of the host for this object.

NewHost

The new name of the host, or blank to unassign the object from its existing host.

GetObjectHelpURL method

Returns the URL where the object's help is stored on the Galaxy repository. The client can display the URL in Internet Explorer or use web publishing to change the data.

Class

IgObject

Syntax

[C#]

```
string GetObjectHelpURL();
```

[Visual C++]

```
HRESULT GetObjectHelpURL(
   [out, retval] BSTR*
);
```

Save method

Save the object after it is configured. Object must be checked out.

Class

IgObject

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
HRESULT Save();
```

CheckIn method

Checks in an object. Invoked for a checked-out object after it is configured and saved.

Class

IgObject

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

Parameters

CheckInComment

Check in an object.

CheckOut method

Checks out an object.

Class

IgObject

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
HRESULT CheckOut();
```

UndoCheckOut method

Reverses the check out operation without making changes.

Class

IgObject

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
HRESULT UndoCheckOut();
```

Unload method

Unloads the gObject cache. This method does not affect functionality. It is used to release internal resources obtained by the object during operations. Unload() can be called to free those resources. After this method is called, the object is still usable and will reload the internal resources as needed.

Class

IgObject

Syntax

[C#]

```
void Unload();
```

[Visual C++]

```
HRESULT Unload();
```

EditStatus property

Edit the status of the object.

Class

IgObject

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
HRESULT EditStatus(
   [out, retval] enum EEditStatus* theEditStatus
);
```

Parameters

theEditStatus

The edit status of the object.

AddExtensionPrimitive method

Adds an extension primitive to the object.

Class

IgObject

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   [in] BSTR ExtensionType,
   [in] BSTR ExtensionPrimitiveName,
   [in, optional, defaultvalue( -1 )]
   VARIANT_BOOL IsObjectExtension
);
```

Parameters

ExtensionType

Type of extension.

ExtensionPrimitiveName

Name of extension primitive to be added. For an attribute extension primitive, this parameter is the fully qualified name of the attribute being extended. For an object extension primitive, it is the new name specified by user for object extension primitive.

IsObjectExtension

If ExtensionPrimitiveName is the new name specified by the user, this field is set to true (for example, when add script pass is true).

DeleteExtensionPrimitive method

Deletes an extension primitive.

Class

IgObject

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
HRESULT DeleteExtensionPrimitive(
   [in] BSTR ExtensionType,
   [in] BSTR ExtensionPrimitiveName
);
```

Parameters

ExtensionType

Type of extension.

ExtensionPrimitiveName

Name of extension primitive to be deleted.

RenameExtensionPrimitive method

Renames an object extension primitive.

Class

IgObject

Syntax

[C#]

```
void RenameExtensionPrimitive(
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
HRESULT RenameExtensionPrimitive(
   [in] BSTR OldPrimitiveName,
   [in] BSTR NewPrimitiveName
);
```

Parameters

OldPrimitiveName

Old name of primitive extension.

NewPrimitiveName

New name specified by user.

AddUDA method

Adds a UDA.

Class

IgObject

Syntax

[C#]

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
HRESULT AddUDA
(
   [in] BSTR UDAName,
   [in, optional, defaultvalue( MxBoolean )] MxDataType Datatype,
   [in, optional, defaultvalue( MxCategoryWriteable_UC_Lockable )] MxAttributeCategory
   Category,
   [in, optional, defaultvalue( MxSecurityOperate )] MxSecurityClassification Security,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL IsArray,
   [in, optional] VARIANT ArrayElementCount
);
```

Parameters

UDAName

The new UDA's name.

Datatype

The new UDA's datatype.

Category

The new UDA's category.

Security

The new UDA's security classification.

IsArray

The new UDA is an array.

ArrayElementCount

The new UDA's array element count.

DeleteUDA method

Deletes a UDA by its name.

Class

IgObject

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

```
HRESULT DeleteUDA(
   [in] BSTR UDAName
);
```

Parameters

UDAName

Delete a UDA by its name.

RenameUDA method

Changes the name of a UDA.

Class

IgObject

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
HRESULT RenameUDA(
   [in] BSTR OldUDAName,
   [in] BSTR NewUDAName
);
```

Parameters

OldUDAName

The old external name of the primitive to be renamed.

NewUDAName

The new external name of the primitive.

UpdateUDA method

Updates a UDA with new data that overwrites the old data.

Class

IgObject

Syntax

[C#]

```
void UpdateUDA(
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
HRESULT UpdateUDA
(
   [in] BSTR wszUDAInfo,
   [in, optional, defaultvalue( MxBoolean )] MxDataType Datatype,
   [in, optional, defaultvalue( MxCategoryWriteable_UC_Lockable )] MxAttributeCategory
   Category,
   [in, optional, defaultvalue( MxSecurityOperate )] MxSecurityClassification Security,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL IsArray,
   [in, optional] VARIANT ArrayElementCount);
);
```

Parameters

wszUDAInfo

The updated UDA data.

Datatype

The updated UDA's datatype.

Category

The updated UDA's category.

Security

The updated UDA's security classification.

IsArray

The updated UDA is an array.

ArrayElementCount

The updated UDA's array element count.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IgObject

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

CategoryGUID property

Returns the category GUID of the object.

Class

IgObject

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
HRESULT CategoryGUID
(
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

GetExtendedAttributes method

Returns the extended attributes in the hierarchy.

Class

IgObject

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

```
HRESULT GetExtendedAttributes(
   [in] BSTR AttributeName,
   [in] int upto_level,
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   [out,retval]IAttributes** theAttributes
);
```

Parameters

AttributeName

Attribute full name which is extended

upto_level

Up to how many level user needs extended primitives -1 will return all levels

ReturnOnlyTheseCategories

Return attributes which are only in these categories. If safe array is empty, method will return all attributes.

theAttributes

Returns the attributes

PlatformLicenseType property

Sets or returns the license type of the Platform object.

Class

IgObject

Syntax

[C#]

```
int PlatformLicenseType { get; set; }
```

[Visual C++]

To get the FLEX License type of the Platform (gObject):

```
[propget]
HRESULT PlatformLicenseType
   (
      [out, retval] int* platformLicenseType
   );
```

To set the FLEX License type of the Platform (gObject):

```
[propput]
HRESULT PlatformLicenseType
   (
      [in] int platformLicenseType
   );
```

Parameters

PlatformLicenseType

Integer that defines the Platform License Type (single or multi-engine)

If PlatformLicenseType is not provided (blank), the existing Platform License Type is cleared.

- 1 = Single Engine
- 2 = Multi-Engine

IgObjects

Represents a collection of gObject objects.

IgObjects class members

This class has the following members.

Operations

CommandResults property
Returns the CommandResults Object, which has the last method call's result.

Area property
Sets the area for the given set of gObjects.

Host property
Sets the host for the given set of gObjects.

AddFromCollection method
Adds objects to the list from another list.

Upload method
Uploads AutomationObject configuration changes made in the runtime.

CheckIn method
Checks in the gObjects in the collection.

Add method
Adds an object to the list.

Container property
Sets the container for the given set of gObjects.

ExportObjects method
Exports object from the Galaxy.

ExportObjectsAsProtected method
Exports objects from the Galaxy as protected.

DeleteAllObjects method
Deletes all objects in the collection.

CheckOut method
Checks out the gObjects in the collection.

SecurityGroup property

Sets the security group for the given set of gObjects.

Undeploy method

Undeploys the instances in the given collection of objects.

Deploy Method

Deploys the instances in the given collection of objects.

Item property

Returns the gObject object by index.

Count property

Returns the number of gObjects in the collection.

UndoCheckOut method

Undoes check-out on the gObjects in the collection.

DeployEx method

Deploy the instances in the given collection of objects.

UndeployEx method

Undeploy the instances in the given collection of objects.

UploadEx method

Uploads AutomationObject configuration changes made in the runtime.

CommandResults property

Returns the CommandResults object, which has the last method call's result.

Class

IgObjects

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

Parameters

CommandResults

Returns the CommandResults object.

Area property

Set the area for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Area { set; };
```

[Visual C++]

```
HRESULT Area(
   in, retval] BSTR* NewArea
);
```

Parameters

NewArea

The name of the Area object, or blank or unspecified to unassign from existing area.

Remarks

The area of an AppObject gObject cannot be set directly. Set the container or host. The gObjects do not need to be checked out; the operation is immediate.

Host property

Sets the host for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Host { set; };
```

[Visual C++]

```
HRESULT Host(
   [in] BSTR NewHost
);
```

Parameters

NewHost

The name of the host object, or blank or unspecified to unassign from existing area.

Remarks

If the gObject is an ApplicationObject, its area and container are also updated. The gObjects do not need to be checked out; the operation is immediate.

AddFromCollection method

Add objects to the list from another list.

Class

IgObjects

Syntax

[C#]

```
void AddFromCollection(
   IgObjects gObjects
);
```

[Visual C++]

```
HRESULT AddFromCollection(
   [in] IgObjects* gObjects
);
```

Parameters

gObjects

Collection representing templates or instances.

Upload method

Uploads AutomationObject configuration changes made at run time.

Class

IgObjects

Syntax

[C#]

```
void Upload(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

[Visual C++]

```
HRESULT Upload(
   [in, optional, defaultvalue( doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

Parameters

automaticallyUndocheckout

Automatically undo the check out.

skipOtherUsersCheckedOutObjects

Skip other users' checked out objects.

skipObjectsWithPendingUpdates

Skip objects with pending updates.

CheckIn method

Check in the gObjects in the collection.

Class

IgObjects

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
HRESULT CheckIn(
   [in, optional] BSTR CheckInComment
);
```

Parameters

CheckInComment

The check-in comment.

Add method

Add an object to the list.

Class

IgObjects

Syntax

[C#]

```
void Add(
   igObject gObject
);
```

[Visual C++]

```
HRESULT Add(
   [in] igObject* gObject
);
```

Parameters

gObject

A gObject to be added to the list.

Container property

Sets the container for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Container { set; };
```

[Visual C++]

```
HRESULT Container(
   [in] BSTR NewContainer
);
```

Parameters

NewContainer

The name of the container, or blank to uncontain.

Remarks

If the gObject is an ApplicationObject, its area and host are also updated. The gObjects do not need to be checked out; the operation is immediate.

ExportObjects method

Exports an object from the Galaxy.

Class

IgObjects

Syntax

[C#]

```
void ExportObjects(
   EExportType exportType, string outputFile
);
```

[Visual C++]

```
HRESULT ExportObjects(
   [in] EExportType exportType,
   [in] BSTR outputFile
);
```

Parameters

exportType

Indicates whether to export to aaPKG file.

outputFile

The name of the output file. Appends extension if one is not specified.

ExportObjectsAsProtected method

Exports object templates and graphics from the Galaxy as protected.

Class

IgObjects

Syntax

[C#]

```
void ExportObjectsAsProtected(
   string outputFile
);
```

[Visual C++]

```
HRESULT ExportObjectsAsProtected(
   [in] BSTR outputFile
);
```

Parameters

outputFile

The name of the output file. Appends extension if one is not specified.

Remarks

There is no exportType parameter as only objects and templates, not instances, can be exported as protected. The default exportType is "exportAsPDF".

DeleteAllObjects method

Delete all objects in the collection.

Class

IgObjects

Syntax

[C#]

```
void DeleteAllObjects();
```

[Visual C++]

```
HRESULT DeleteAllObjects();
```

CheckOut method

Check out the gObjects in the collection.

Class

IgObjects

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
HRESULT CheckOut();
```

SecurityGroup property

Sets the security group for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string SecurityGroup { set; };
```

[Visual C++]

```
HRESULT SecurityGroup(
   [in] BSTR rha
);
```

Parameters

rhs

The name of the security group.

Undeploy method

Undeploy the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

```
HRESULT Undeploy(
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure
);
```

Parameters

UndeployRule

Indicates whether to force an object off-scan if it is running on-scan prior to undeployment.

markAsUndeployedOnFailure

On failure, set deploy status as undeployed.

Deploy Method

Deploy the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Deploy(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   bool markAsDeployedOnStatusMismatch
);
```

[Visual C++]

```
HRESULT Deploy
(
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch
);
```

Parameters

actionForDeployedObjects

Action for currently deployed object.

skipIfCurrentlyUndeployed

Action for currently undeployed object.

deployOnScan

Deploy on scan.

forceOffScan

Force off scan.

markAsDeployedOnStatusMismatch

Deploy status mismatch.

Item property

Returns the gObject object by index.

Class

IgObjects

Syntax

[C#]

```
IgObject this[object gObjectIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT gObjectIdentifier,
   [out, retval] IgObject**
);
```

Parameters

gObjectIdentifier

The index of the gObject. This is a numeric value from 1 to Count, or the name of the gObject.

IgObject

The returned gObject object.

Count property

Returns the number of gObjects in the collection.

Class

IgObjects

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

UndoCheckOut method

Reverses the check out operation the gObjects in the collection without making changes.

Class

IgObjects

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
HRESULT UndoCheckOut();
```

DeployEx method

Deploys the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void DeployEx(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   bool markAsDeployedOnStatusMismatch,
   string strCustomData
);
```

[Visual C++]

```
HRESULT DeployEx(
   // Action for currently deployed object
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   // Action for currently undeployed object
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   // Deploy on scan
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   // Force off scan
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   // Deploy status mismatch
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UndeployEx method

Undeploys the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   bool markAsUndeployedOnFailure
   string strCustomData
);
```

[Visual C++]

```
HRESULT UndeployEx(
   // Indicates whether to force an object off-scan
   // if it's running on-scan prior to undeployment.
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   // On Failure, set deploy status as undeployed
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UploadEx method

Uploads AutomationObject configuration changes made at run time.

Class

IgObjects

Syntax

[C#]

```
void UploadEx(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   string strCustomData
);
```

[Visual C++]

```
HRESULT UploadEx(
   // Automatically Undocheckout
   [in, optional, defaultvalue(doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   // Skip other users CheckedOuts
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   // Skip objects with pending updates
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

IGRAccess

Main interface implemented by GRAccess used by clients to gain access to the Galaxy Repository.

IGRAccess class members

This class has the following members.

Operations

CommandResult property

Returns the *CommandResult* Object, which has the last method call's result.

CreateGalaxy method

Create a new Galaxy on the GR node.

DeleteGalaxy method

Delete a Galaxy.

QueryGalaxies method

Query available Galaxies on a given GR node.

QueryGalaxiesEx method

Queries a given Galaxy Repository (GR) node to list all available Galaxies, including the Modern InTouch ViewApp Galaxies. You do not need to be logged in to run this query.

CreateGalaxyFromTemplate method

Creates a new Galaxy from the template.

CommandResult property

Returns the *CommandResult* object, which has the last method call's result.

Class

IGRAccess

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the *CommandResult* object.

CreateGalaxy method

Creates a new Galaxy on the GR node.

Class

IGRAccess

Syntax

[C#]

```
void CreateGalaxy(
   string galaxyName,
   string GRNodeName,
   bool enableSecurity,
   EAuthenticationMode AuthenticationMode,
   string osUserName
);
```

[Visual C++]

```
HRESULT CreateGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName,
   [in, optional, defaultvalue( -1 )]
   VARIANT_BOOL enableSecurity,
   [in, optional, defaultvalue ( osAuthenticationMode )] EAuthenticationMode
   authenticationMode,
   [in, optional] BSTR osUserName
);
```

Parameters

galaxyName

The name to give the new Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

enableSecurity

Indicates whether security should be enabled.(valid only for Galaxy authentication).

authenticationMode

Indicates whether OS or Galaxy authentication should be used.

osUserName

The name of the OS user if osAuthenticationMode is specified. Ignored otherwise. If not specified in OS authentication mode, the user currently logged in to the OS is used.

DeleteGalaxy method

Deletes a Galaxy.

Class

IGRAccess

Syntax

[C#]

```
void DeleteGalaxy(
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT DeleteGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName
);
```

Parameters

galaxyName

The name of the Galaxy to be deleted.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

QueryGalaxies method

Queries available Galaxies on a given GR node.

Class

IGRAccess

Syntax

[C#]

```
IGalaxies QueryGalaxies(
   string GRNodeName
);
```

[Visual C++]

```
HRESULT QueryGalaxies(
   [in, optional] BSTR grNodeName,
   [out, retval] IGalaxies** galaxies
);
```

Parameters

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

galaxies

The list of galaxies.

QueryGalaxiesEx method

Queries a given Galaxy Repository (GR) node to list all available Galaxies, including the Modern InTouch ViewApp Galaxies. You do not need to be logged in to run this query.

Class

GRAccess

Syntax

[C#]

```
IGalaxies QueryGalaxiesEx(string GRNodeName);
```

[Visual C++]

```
HRESULT QueryGalaxiesEx
   (
      [in, optional] BSTR grNodeName,
      [out, retval] IGalaxies** galaxies
   );
```

Parameters

grNodeName

Name of the Galaxy Repository to be queried.

CreateGalaxyFromTemplate method

Creates a new Galaxy from the template.

Class

IGRAccess

Syntax

[C#]

```
   void CreateGalaxyFromTemplate(
   string createGalaxyTemplateName,
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT CreateGalaxyFromTemplate(
   [in] BSTR CreateGalaxyTemplateName,
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName,
[out, retval] IGalaxies** galaxies
);
```

Parameters

createGalaxyTemplateName

The template name to create the Galaxy.

galaxyName

The name to give the new Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

IInstance

Represents an instance object.

IInstance class members

This class has the following members.

Operations

AddExtensionPrimitive method

Add an extension primitive.

AddUDA Method

Add a UDA.

Area property

Get or set the area of an object.

Attributes property

Gets the collection of attributes for this object.

BasedOn property

Returns the name of the base template for this object.

Category property

Returns the category of this object.

CategoryGUID property

Returns the category GUID for this object.

CheckedOutBy property

Returns the name of the user who has checked out this object.

CheckIn method

Checks in an object.

CheckOut method

Checks out an object.

CheckoutStatus property

Returns the check out status of the object.

CommandResult property

Returns the CommandResult object, which has the results from the object’s last method call.

ConfigurableAttributes property

Returns a collection of this object’s attributes that can be modified.

ConfigVersion property

Returns the this object’s configuration version number.

ContainedName property

Sets or returns the object’s contained name.

Container property

Sets or returns the object’s container.

DeleteExtensionPrimitive method

Delete an extension primitive.

DeleteInstance method

Delete this instance.

DeleteUDA method

Delete a UDA.

Deploy method

Deploy this instance.

DeployedVersion property

The configuration version of this instance at the time it was last deployed.

DeployEx method

Deploy this instance.

DeploymentStatus property

Indicates whether the instance is currently deployed and if so, if it has pending changes.

DerivedFrom property

Returns the name of the object’s template.

EditStatus property

Edit the status of this object.

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

GetExtendedAttributes method

Get the extended attributes in the hierarchy.

GetObjectHelpURL method

Returns the URL to where the object help is stored on the Galaxy repository.

HierarchicalName property

Returns the hierarchical name of this object.

Host property

Returns the name of this object’s host.

RenameExtensionPrimitive method

Rename the object extension primitive.

RenameUDA method

Change the name of the UDA.

Save method

Save the object.

Tagname property

Returns or sets the object’s tagname.

Undeploy method

Undeploy this instance.

UndeployEx method

Undeploy this instance.

UndoCheckOut method

Reverse the check out operation without saving changes.

Unload method

Unload the object cache.

UpdateUDA method

Save the UDA information.

Upload method

Uploads AutomationObject configuration changes made in the runtime.

UploadEx method

Uploads the AutomationObject configuration changes.

ValidationStatus property

Returns this object’s validation status.

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

AddExtensionPrimitive method

Adds an extension primitive.

Class

IInstance

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of extension primitive to be added -
   // For attribute extension primitive - Fully
   // qualified name of attribute being extended
   // For object extension primitive - New name
   // specified by user for object extension primitive
   [in] BSTR ExtensionPrimitiveName,
   // If wszName is the new name specified by user,
   // this field will be set to true. for example
   //when add script pass true
   [in, optional, defaultvalue( -1 )]VARIANT_BOOL IsObjectExtension
);
```

AddUDA Method

Adds a UDA.

Class

IInstance

Syntax

[C#]

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Add a UDA
//
// Returns:
// S_OK - Successfully add a UDA
HRESULT AddUDA(
   //Same XML format. But only one is allowed.
   [in] BSTR UDAName,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Area property

Gets or sets the area of this object. An AppObject area must be set using the SetContainer or SetHost methods. The object does not need to be checked out, and the operation is immediate.

Class

IInstance

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
// Returns name of the Area object, or blank if the
// object doesn't have an Area.
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
// Set the Area of this object.
[propput]
HRESULT Area(
   // The name of the Area, or blank to unassign
   // from existing Area.
   [in] BSTR NewArea
);
```

Attributes property

Returns the collection of attributes of this object.

Class

IInstance

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object.
// Configuring attributes requires that the object
// be checked out.
HRESULT Attributes(
   // The collection of Attributes of this object.
   [out, retval] IAttributes** theAttributes
);
```

BasedOn property

Returns the name of the base template for this object.

Class

IInstance

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
// The name of the base template (root ancestor) of this object.
HRESULT BasedOn(
   // The name of the base template (root ancestor)
   // of this object.
   [out, retval] BSTR* theBaseTemplate
);
```

Category property

Returns the category of this object.

Class

IInstance

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
// The Category of the object.
HRESULT Category(
   // The Category of the object.
   [out, retval] enum ECATEGORY* theCategory
);
```

CategoryGUID property

Gets the category GUID of this object.

Class

IInstance

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
// The Category GUID of the object.
HRESULT CategoryGUID(
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

IInstance

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
// The name of the user that checked out this object.
HRESULT CheckedOutBy(
   // The name of the user that checked out this object.
   [out, retval] BSTR* CheckedOutByUserName
);
```

CheckIn method

Checks in this object.

Class

IInstance

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
// Checks in an object.
// Called on a checked-out object after it is
// configured and saved.
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

CheckOut method

Checks out an object.

Class

IInstance

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
// Checks out an object.
HRESULT CheckOut();
```

CheckoutStatus property

Returns the check out status of the object.

Class

IInstance

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
// Get the check-out status of the object.
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

CommandResult property

Returns the CommandResult object, which has the results from the last method call.

Class

IInstance

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

ConfigurableAttributes property

Returns a collection of this object’s attributes that can be configured.

Class

IInstance

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object that
// can be configured.
// Configuring attributes requires that the object
// be checked out.
HRESULT ConfigurableAttributes(
   [out, retval] IAttributes** theConfigurableAttributes
);
```

ConfigVersion property

Returns this object’s configuration version number. The configuration version is incremented each time the object is checked in.

Class

IInstance

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

```
// The configuration version of the object.
HRESULT ConfigVersion(
   [out, retval] LONG* theConfigVersion
);
```

ContainedName property

Sets or returns this object's contained name. When setting the contained name, the object does not need to be checked out, and the operation is immediate.

Class

IInstance

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
//
// Returns Contained name of this object (eg. Inlet)
[propget]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [out, retval] BSTR* theContainedName
);
// Sets the contained name of this object (eg. Inlet)
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [in] BSTR newContainedName
);
```

Container property

Returns or sets the container for this object. When setting the container, the object does not need to be checked out, and the change is immediate. For an ApplicationObject, the ApplicationObject's Area and Host area are also updated.

Class

IInstance

Syntax

[C#]

```
string Container { set; get; };
```

[Visual C++]

```
// Returns name of the Container object, or blank if
//object is not contained.
[propget]
HRESULT Container(
   // The name of the Container object, or blank if
   // the object is not contained.
   [out, retval] BSTR* theContainer
);
// Set the Container for this object.
[propput]
HRESULT Container(
   // The name of the container, or blank to uncontain.
   [in] BSTR NewContainer
);
```

DeleteExtensionPrimitive method

Deletes an extension.

Class

IInstance

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
// Delete an extension primitive
// Returns S_OK
HRESULT DeleteExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of Extension primitive to be deleted.
   [in] BSTR ExtensionPrimitiveName
);
```

DeleteInstance method

Delete an instance.

Class

IInstance

Syntax

[C#]

```
void DeleteInstance(
   EForceDeleteInstanceOption ForceDeleteOption
);
```

[Visual C++]

```
HRESULT DeleteInstance(
   [in, optional, defaultvalue( dontForceInstanceDelete )] EForceDeleteInstanceOption
   ForceDeleteOption
);
```

Parameters

ForceDeleteOption

Various options to indicate whether deletion of the object should be forced.

DeleteUDA method

Deletes a UDA.

Class

IInstance

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

```
// Delete a UDA
// Returns:
// S_OK - Successfully delete a UDA
HRESULT DeleteUDA(
   // Delete a UDA by UDA name.
   [in] BSTR UDAName
);
```

Deploy method

Deploys the instance.

Class

IInstance

Syntax

[C#]

```
void Deploy(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   ECascade cascade,
   bool markAsDeployedOnStatusMismatch
);
```

[Visual C++]

```
HRESULT Deploy(
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch
);
```

Parameters

actionForDeployedObjects

Action for currently deployed object.

skipIfCurrentlyUndeployed

Action for currently undeployed object.

deployOnScan

Deploy on scan.

forceOffScan

Force off scan.

cascade

Cascade.

markAsDeployedOnStatusMismatch

Deploy status mismatch.

DeployedVersion property

The configuration version of this instance at the time it was last deployed.

Class

IInstance

Syntax

[C#]

```
int DeployedVersion { get; };
```

[Visual C++]

```
HRESULT DeployedVersion(
   [out, retval] LONG*
);
```

DeployEx method

Deploy this instance.

Class

IInstance

Syntax

[C#]

```
void DeployEx(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   ECascade cascade,
   bool markAsDeployedOnStatusMismatch,
   string strCustomData
);
```

[Visual C++]

```
// Deploy this Instance.
HRESULT DeployEx(
   // Action for currently deployed object
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   // Action for currently undeployed object
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
```

```
    skipIfCurrentlyUndeployed,
    // Deploy on scan
    [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
    // Force off scan
    [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
    // Cascade
    [in, optional, defaultvalue( doCascade )] ECascade cascade,
    // Deploy status mismatch
    [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch,
    // Data used by custom category package
    [in, optional, defaultvalue("")] BSTR strCustomData
);
```

DeploymentStatus property

Indicates whether the instance is currently deployed and if so, if it has pending changes.

Class

IInstance

Syntax

[C#]

```
EDeploymentStatus DeploymentStatus { get; };
```

[Visual C++]

```
HRESULT DeploymentStatus(
    [out, retval] EDeploymentStatus* theDeploymentStatus
);
```

Parameters

theDeploymentStatus

Indicates whether the instance is currently deployed and if so, if it has pending changes.

DerivedFrom property

Returns the name of the object’s template.

Class

IInstance

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
// Returns the name of the template that this object
// was created from.
[propget]
HRESULT DerivedFrom(
   // The name of the template that this object was
   // created from.
   [out, retval] BSTR* theParentTemplate
);
```

EditStatus property

Edits the status of the object.

Class

IInstance

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
// The edit status of the object.
HRESULT EditStatus(
   // The edit status of the object.
   [out, retval] enum EEditStatus* theEditStatus
);
```

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

IInstance

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
// The list of errors generated the last
// time the object was validated and saved.
[propget]
HRESULT Errors(
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

GetExtendedAttributes method

Retrieves the extended attributes in the hierarchy.

Class

IInstance

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

```
//Get Extended Attributes in Hierarchy
HRESULT GetExtendedAttributes(
   // Attribute full name which is extended
   [in] BSTR AttributeName,
   // Up to how many level user needs extended
   // primitives
   // -1 will return all levels
   [in] int upto_level,
   // Return attributes which are only these
   // categories. If safearray is empty, returns all
   // attributes.
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   // Return attributes
   [out,retval]IAttributes** theAttributes
);
```

GetObjectHelpURL method

Returns the URL where the object help is stored on the Galaxy repository.

Class

IInstance

Syntax

[C#]

```
string GetObjectHelpURL();
```

[Visual C++]

```
// Returns the URL to where the Object Help is
// stored on the Galaxy Repository.
// The client can display the URL in
// InternetExplorer or use web publishing
// to change the data.
HRESULT GetObjectHelpURL(
   // Returns the URL for the Object Help
   // location on the GR.
   [out, retval] BSTR*
);
```

HierarchicalName property

Returns the hierarchical name of this object.

Class

IInstance

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
// The Hierarchical name of this object. (eg.
// Reactor.Tank.Inlet)
HRESULT HierarchicalName(
   [out, retval] BSTR* theHierarchicalName
);
```

Host property

Returns the name of this object’s host.

Class

IInstance

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
// Retrieve the name of the Host for this object, or
// blank if unassigned.
HRESULT Host(
   // The name of the Host for this object.
   [out, retval] BSTR* theHost
);
```

RenameExtensionPrimitive method

Renames the object extension primitive.

Class

IInstance

Syntax

[C#]

```
void RenameExtensionPrimitive(
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
// Rename an object extension primitive
// Returns S_OK
HRESULT RenameExtensionPrimitive(
   // Old name of primitive extension
   [in] BSTR OldPrimitiveName,
   // New name specified by user
   [in] BSTR NewPrimitiveName
);
```

RenameUDA method

Changes the name of a UDA.

Class

IInstance

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
// Rename a UDA
// Returns:
// S_OK - Successfully rename a UDA
HRESULT RenameUDA(
   // The old external name of the primitive
   [in] BSTR OldUDAName,
   // The new external name of the primitive
   [in] BSTR NewUDAName
);
```

Save method

Saves the object.

Class

IInstance

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
// Saves the object after it's configured.
// Object must be checked out.
HRESULT Save();
```

Tagname property

Returns or sets this object’s tagname.

Class

IInstance

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
// Returns the Tagname of this object.
[propget]
HRESULT Tagname(
   // The Tagname of this object.
   [out, retval] BSTR* theTagname
);
// Sets the Tagname of this object.
// The object does not need to be checked out;
// the operation is immediate.
[propput]
HRESULT Tagname(
   // The Tagname of this object.
   [in] BSTR newTagName
);
```

Undeploy method

Undeploy this instance.

Class

IInstance

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   ECascade cascade,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

```
HRESULT Undeploy(
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure
);
```

Parameters

UndeployRule

Indicates whether to force an object off-scan if it is running on-scan prior to undeployment.

cascade

Cascade.

markAsUndeployedOnFailure

On failure, set deploy status as undeployed.

UndeployEx method

Undeploy this instance.

Class

IInstance

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   ECascade cascade,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

```
// Undeploy this Instance.
HRESULT UndeployEx(
   // Indicates whether to force an object off-scan
   // if it's running on-scan prior to undeployment.
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   // Cascade
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   // On Failure, set deploy status as undeployed
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UndoCheckOut method

Reverses the check out operation without making changes.

Class

IInstance

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
// Undo check-out of this object.
HRESULT UndoCheckOut();
```

Unload method

Unloads the object cache, releasing internal resources obtained by the object during operations. After executing this method, the object is still available. Internal resources will be reloaded as necessary.

Class

IInstance

Syntax

[C#]

```
void Unload();
```

[Visual C++]

```
// Unloads the gObject cache. This method doesn't
// affect functionality.
HRESULT Unload();
```

UpdateUDA method

Saves the UDA information.

Class

IInstance

Syntax

[C#]

```
void UpdateUDA(
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Update a UDA
// Returns:
// S_OK - Successfully update a UDA
HRESULT UpdateUDA(
   // Same XML format. But only one is allowed. The
   // new info will overwrite the old info.
   [in] BSTR wszUDAInfo,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Upload method

Uploads AutomationObject configuration changes made at run time.

Class

IInstance

Syntax

[C#]

```
void Upload(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

[Visual C++]

```
HRESULT Upload(
   [in, optional, defaultvalue( doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

Parameters

automaticallyUndocheckout

Automatically undo the check out.

skipOtherUsersCheckedOutObjects

Skip other users' checked out objects.

skipObjectsWithPendingUpdates

Skip objects with pending updates.

UploadEx method

Saves the AutomationObject configuration changes.

Class

IInstance

Syntax

[C#]

```
void UploadEx(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   string strCustomData
);
```

[Visual C++]

```
// Uploads AutomationObject configuration changes
// made during runtime.
```

```
HRESULT UploadEx(
   // Automatically Undocheckout
   [in, optional, defaultvalue(doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   // Skip other users CheckedOuts
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   // Skip objects with pending updates
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

ValidationStatus property

Returns this object's validation status.

Class

IInstance

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
// The ValidationStatus of this object based on the
// last time the object was validated and saved.
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

IInstance

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
// The list of warnings that got generated the last time the object was validated and
```

```
saved.
HRESULT Warnings(
   // The list of warnings that were generated the
   // last time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

IMxValue

Represents the fundamental variant object that contains a single data value or single array of data values of like type. The IMxValue is used to get and set data within the ArchestrA framework.

IMxValue class members

This class has the following members.

Operations

Clone method

Creates a duplicate MxValue object with a state identical to the current MxValue.

Empty method

Changes the MxValue's data type to MxNoData.

PutBoolean method

Stores a Boolean value in the MxValue.

PutInteger method

Stores a long value in the MxValue.

PutFloat method

Stores a float value in the MxValue.

PutDouble method

Stores a double value in the MxValue.

PutString method

Stores a Unicode string value in the MxValue.

PutTime method

Stores a VBFILETIME value in the MxValue.

PutElapsedTime method

Stores a VB_LARGE_INTEGER value in the MxValue.

PutMxReference method

Stores an MxReference in the MxValue.

PutMxStatus method

Stores an MxStatus in the MxValue.

PutMxDataType method

Stores an MxDataType in the MxValue.

PutMxSecurityClassification method

Stores an MxSecurityClassification in the MxValue.

PutMxDataQuality method

Stores an MxDataQuality in the MxValue.

PutCustomStruct method

Stores a struct in the MxValue.

PutCustomEnum method

Stores an MxCustomEnum in the MxValue.

GetDataType method

Returns the data type of the value stored in the MxValue.

GetBoolean method

Returns the value stored in the MxValue as a VARIANT_BOOL.

GetInteger method

Returns the value stored in the MxValue as a long.

GetFloat method

Returns the value stored in the MxValue as a float.

GetDouble method

Returns the value stored in the MxValue as a double.

GetString method

Returns the value stored in the MxValue as a string.

GetTime method

Returns the value stored in the MxValue as a VBFILETIME.

GetElapsedTime method

Returns the value stored in the MxValue as a VB_LARGE_INTEGER.

GetMxReference method

Returns the value stored in the MxValue as an IMxReference *.

GetMxStatus method

Returns the MxStatus stored in the MxValue.

GetMxDataType method

Returns the value stored in the MxValue as an MxDataType.

GetMxSecurityClassification method

Returns the value stored in the MxValue as an MxSecurityClassification.

GetMxDataQuality method

Returns the value stored in the MxValue as an MxDataQuality.

GetCustomStruct method

Returns the qualified struct stored in the MxValue.

GetCustomEnum method

Returns the value stored in the MxValue as a qualified enum.

GetDimensionCount method

Indicates whether the MxValue holds an array.

PutElement method

Puts an element into an array held by an MxValue.

GetElement method

Gets an element from an array held in an MxValue.

GetDimensionSize method

Gets the size of the array held by the MxValue.

PutCustomStructVB method

Stores a qualified struct in the MxValue for a VB client.

GetCustomStructVB method

Returns the qualified struct stored in the MxValue for a VB client.

PutInternationalStrings method

Initializes an MxValue with a set of internationalized strings.

PutInternationalStringsVB method

Initializes an MxValue with a set of internationalized strings for a VB client.

GetInternationalStrings method

Returns the internationalized strings stored by the MxValue.

GetInternationalStringsVB method

Returns the internationalized strings stored by the MxValue for a VB client.

GetInternationalString method

Returns the internationalized string associated with a particular local.

PutInternationalString method

Adds the internationalized string of a specific locale to the existing internationalized strings.

Clone method

Creates a duplicate MxValue object with a state identical to the current MxValue.

Class

IMxValue

Syntax

[C#]

```
void Clone(
   out MxValue ppMxValue
);
```

[Visual C++]

```
HRESULT Clone(
    [out] IMxValue **ppMxValue
);
```

Parameters

ppMxValue

Receives the IMxValue interface pointer of the newly cloned object.

Returns

S_OK - Success

E_POINTER - An invalid pointer was passed in.

E_OUTOFMEMORY - Unable to allocate memory for the new MxValue object.

Empty method

Changes the MxValue's data type to MxNoData.

Class

IMxValue

Syntax

[C#]

```
void Empty();
```

[Visual C++]

```
HRESULT Empty();
```

Returns

S_OK - Success

PutBoolean method

Stores a Boolean value in the MxValue. The MxValue's data type will be MxBoolean.

Class

IMxValue

Syntax

[C#]

```
void PutBoolean(
   bool newVal
);
```

[Visual C++]

```
HRESULT PutBoolean(
   [in] VARIANT_BOOL newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutInteger method

Stores a long value in the MxValue. The MxValue's data type will be MxInteger.

Class

IMxValue

Syntax

[C#]

```
void PutInteger(
   int newVal
);
```

[Visual C++]

```
HRESULT PutInteger(
   [in] long newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutFloat method

Stores a float value in the MxValue. The MxValue's data type will be MxFloat.

Class

IMxValue

Syntax

[C#]

```
void PutFloat(
   float newVal
);
```

[Visual C++]

```
HRESULT PutFloat(
   [in] float newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object. A value that is NaN (not a number) is permitted.

Returns

S_OK - Success

PutDouble method

Stores a double value in the MxValue. The MxValue's data type will be MxDouble.

Class

IMxValue

Syntax

[C#]

```
void PutDouble(
   double newVal
);
```

[Visual C++]

```
HRESULT PutDouble(
   [in] double newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object. A value that is NaN (not a number) is permitted.

Returns

S_OK - Success

PutString method

Stores a Unicode string value in the MxValue. The MxValue's data type will be MxString.

Class

IMxValue

Syntax

[C#]

```
void PutString(
   string newVal
);
```

[Visual C++]

```
HRESULT PutString(
   [in, string] LPCWSTR newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

E_OUTOFMEMORY - Unable to allocate memory for the string.

PutTime method

Stores a VBFILETIME value in the MxValue. The MxValue's data type will be MxTime.

Class

IMxValue

Syntax

[C#]

```
void PutTime(
   ref VBFILETIME pNewVal
);
```

[Visual C++]

```
HRESULT PutTime(
   [in] VBFILETIME *pNewVal
);
```

Parameters

pNewVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutElapsedTime method

Stores a VB_LARGE_INTEGER value in the MxValue. The MxValue's data type will be MxElapsedTime.

Class

IMxValue

Syntax

[C#]

```
void PutElapsedTime(
   ref VB_LARGE_INTEGER pNewVal
);
```

[Visual C++]

```
HRESULT PutElapsedTime(
   [in] VB_LARGE_INTEGER *pNewVal
);
```

Parameters

pNewVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxReference method

Stores an MxReference in the MxValue. The MxValue's data type will be MxReferenceType.

Class

IMxValue

Syntax

[C#]

```
void PutMxReference(
   IMxReference newVal
);
```

[Visual C++]

```
HRESULT PutMxReference(
   [in] IMxReference *newVal
);
```

Parameters

newVal

The MxReference that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxStatus method

Stores an MxStatus in the MxValue. The MxValue's data type will be MxStatusType.

Class

IMxValue

Syntax

[C#]

```
void PutMxStatus(
   ref MxStatus newVal
);
```

[Visual C++]

```
HRESULT PutMxStatus(
   [in] const MxStatus *newVal
);
```

Parameters

newVal

The MxStatus that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxDataType method

Stores an MxDataType in the MxValue. The MxValue's data type will be MxDataTypeEnum.

Class

IMxValue

Syntax

[C#]

```
void PutMxDataType(
   MxDataType newVal
);
```

[Visual C++]

```
HRESULT PutMxDataType(
   [in] MxDataType newVal
);
```

Parameters

newVal

The MxDataType that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxSecurityClassification method

Stores an Mx SecurityClassification in the MxValue. The MxValue's data type will be MxSecurityClassificationEnum.

Class

IMxValue

Syntax

[C#]

```
void PutMxSecurityClassification(
   MxSecurityClassification newVal
);
```

[Visual C++]

```
HRESULT PutMxSecurityClassification(
   [in] MxSecurityClassification newVal
);
```

Parameters

newVal

The MxSecurityClassification that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxDataQuality method

Stores an MxData Quality in the MxValue. The MxValue's data type will be MxDataQualityEnum.

Class

IMxValue

Syntax

[C#]

```
void PutMxDataQuality(
   ref short newVal
);
```

[Visual C++]

```
HRESULT PutMxDataQuality(
   [in] const MxDataQuality *newVal
);
```

Parameters

newVal

The MxDataQuality that will be stored in the MxValue object.

Returns

S_OK - Success

PutCustomStruct method

Stores a struct in the MxValue. The MxValue's data type will be MxCustomStruct.

Class

IMxValue

Syntax

[C#]

```
void PutCustomStruct(
   int guid,
   int structSize,
   ref byte pStruct
);
```

[Visual C++]

```
HRESULT PutCustomStruct(
   [in] long guid,
   [in] long structSize,
   [in, size_is(structSize)] const unsigned char *pStruct
);
```

Parameters

guid

A quasi-GUID that uniquely identifies this struct. The BRO does not validate the GUID.

structSize

Size of the struct.

pStruct

Pointer to the struct. MxValue will create its own copy of the struct.

Returns

S_OK - Success

PutCustomEnum method

Stores an MxCustomEnum in the MxValue. The MxValue's data type will be MxCustomEnum.

Class

IMxValue

Syntax

[C#]

```
void PutCustomEnum(
   string value,
   short ordinal,
   short primitiveId,
   short attributeId
);
```

[Visual C++]

```
HRESULT PutCustomEnum(
   [in] LPCWSTR value,
   [in] short ordinal,
   [in] MxPrimitiveId primitiveId,
   [in] MxAttributeId attributeId
);
```

Parameters

value,

The enumeration's string value. This parameter is used by Message Exchange clients. BRO can pass in 0 for internal clients.

ordinal,

The enumeration's ordinal value. This parameter is used by primitives. Other clients can pass in 0.

primitiveId,

The primitive corresponding to the attribute which holds an array of all the possible enumeration string values, or just pass in 0. This parameter is ignored by the BRO on external sets. This parameter is used by the BRO to initialize an MxValue.

attributeId

The attribute holding an array of all the possible enumeration string values, or just pass in 0. This parameter is ignored by the BRO on external sets. This parameter is used by the BRO to initialize an MxValue.

Returns

S_OK - Success

GetDataType method

Returns the data type of the value stored in the MxValue.

Class

IMxValue

Syntax

[C#]

```
MxDataType GetDataType();
```

[Visual C++]

```
HRESULT GetDataType(
   [out, retval] MxDataType *pVal
);
```

Parameters

pVal

Receives the data type.

Returns

S_OK - Success

E_POINTER - An invalid pointer was passed in.

GetBoolean method

Returns the value stored in the MxValue as a VARIANT_BOOL.

Class

IMxValue

Syntax

[C#]

```
bool GetBoolean();
```

[Visual C++]

```
HRESULT GetBoolean(
   [out, retval] VARIANT_BOOL *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to VARIANT_BOOL.

MX_E_UnableToConvertString - The string held in the MxValue is not "true" or "false".

E_POINTER - An invalid pointer was passed in.

GetInteger method

Returns the value stored in the MxValue as a long.

Class

IMxValue

Syntax

[C#]

```
int GetInteger();
```

[Visual C++]

```
HRESULT GetInteger(
   [out, retval] long *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to long.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a long.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetFloat method

Returns the value stored in the MxValue as a float.

Class

IMxValue

Syntax

[C#]

```
float GetFloat();
```

[Visual C++]

```
HRESULT GetFloat(
   [out, retval] float *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to float.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a float.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetDouble method

Returns the value stored in the MxValue as a double.

Class

IMxValue

Syntax

[C#]

```
double GetDouble();
```

[Visual C++]

```
HRESULT GetDouble(
   [out, retval] double *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to double.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a double.

E_POINTER - An invalid pointer was passed in.

GetString method

Returns the value stored in the MxValue as a string.

Class

IMxValue

Syntax

[C#]

```
string GetString();
```

[Visual C++]

```
HRESULT GetString(
   [out, retval] BSTR *pVal
);
```

Parameters

pVal

Receives the value. Use SysFreeString to free the BSTR when finished with it.

If the value stored in the MxValue is not of type MxString, a string representation of the value is returned as follows:

- MxNoData - "No Data"
- MxBoolean - "true" or "false"
- MxInteger - "1234"

- MxFloat - "3.5"
- MxDouble - "3.5"
- MxString - "Hello World"
- MxTime - "Sun May 01 20:27:01 1994"
- MxElapsedTime - "89384" (elapsed time in 100 nanosecond increments.)
- MxReferenceType - "Valve1.pv"
- MxStatusType - "Error detected by Requesting NMX: Request timed out."
- MxDataTypeEnum - "MxInteger"
- MxSecurityClassificationEnum - "MxSecurityFreeAccess"
- MxDataQualityType - "Uncertain"
- MxCustomEnum - "Open"
- MxCustomStruct - "478602" (guid for the struct)

Returns

S_OK - Success

E_OUTOFMEMORY - Unable to allocate memory for the string.

E_POINTER - An invalid pointer was passed in.

GetTime method

Returns the value stored in the MxValue as a VBFILETIME.

Class

IMxValue

Syntax

[C#]

```
void GetTime(
   out VBFILETIME pVal
);
```

[Visual C++]

```
HRESULT GetTime(
   [out] VBFILETIME *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to VBFILETIME.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a VBFILETIME.

E_POINTER - An invalid pointer was passed in.

GetElapsedTime method

Returns the value stored in the MxValue as a VB_LARGE_INTEGER.

Class

IMxValue

Syntax

[C#]

```
VB_LARGE_INTEGER GetElapsedTime();
```

[Visual C++]

```
HRESULT GetElapsedTime(
   [out, retval] VB_LARGE_INTEGER *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not of type MxElapsedTime.

E_POINTER - An invalid pointer was passed in.

GetMxReference method

Returns the value stored in the MxValue as an IMxReference *.

Class

IMxValue

Syntax

[C#]

```
IMxReference GetMxReference();
```

[Visual C++]

```
HRESULT GetMxReference(
   [out, retval] IMxReference **pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not an MxReference or MxString.

E_POINTER - An invalid pointer was passed in.

GetMxStatus method

Returns the MxStatus stored in the MxValue.

Class

IMxValue

Syntax

[C#]

```
MxStatus GetMxStatus();
```

[Visual C++]

```
HRESULT GetMxStatus(
   [out, retval] MxStatus *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not an MxStatus.

E_POINTER - An invalid pointer was passed in.

GetMxDataType method

Returns the value stored in the MxValue as an MxDataType.

Class

IMxValue

Syntax

[C#]

```
MxDataType GetMxDataType();
```

[Visual C++]

```
HRESULT GetMxDataType(
   [out, retval] MxDataType *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to MxDataType.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxDataType.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetMxSecurityClassification method

Returns the value stored in the MxValue as an MxSecurityClassification.

Class

IMxValue

Syntax

[C#]

```
MxSecurityClassification GetMxSecurityClassification();
```

[Visual C++]

```
HRESULT GetMxSecurityClassification(
   [out, retval] MxSecurityClassification *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to MxSecurityClassification.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxSecurityClassification.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetMxDataQuality method

Returns the value stored in the MxValue as an MxDataQuality.

Class

IMxValue

Syntax

[C#]

```
short GetMxDataQuality();
```

[Visual C++]

```
HRESULT GetMxDataQuality(
   [out, retval] MxDataQuality *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to MxMxDataQualityataType.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxDataQuality.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetCustomStruct method

Returns the qualified struct stored in the MxValue.

Class

IMxValue

Syntax

[C#]

```
void GetCustomStruct(
   out int pGuid,
   out int pStructSize,
   System.IntPtr pStruct
);
```

[Visual C++]

```
HRESULT GetCustomStruct(
   [out] long *pGuid,
   [out] long *pStructSize,
   [out, size_is(,*pStructSize)] unsigned char **pStruct
);
```

Parameters

pGuid,

Receives the quasi-GUID that uniquely identifies this struct. The BRO does not validate the GUID.

pStructSize,

Receives the size of the struct held by the MxValue.

pStruct

Address of the pointer which will receive the struct. The caller must call CoTaskMemFree on pStruct when it is finished using it.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not a qualified struct.

E_POINTER - An invalid pointer was passed in.

GetCustomEnum method

Returns the value stored in the MxValue as a qualified enum.

Class

IMxValue

Syntax

[C#]

```
void GetCustomEnum(
   out string pValue,
   out short pOrdinal,
   out short pPrimitiveId,
   out short pAttributeId
);
```

[Visual C++]

```
HRESULT GetCustomEnum(
   [out] BSTR *pValue,
   [out] short *pOrdinal,
   [out] MxPrimitiveId *pPrimitiveId,
   [out] MxAttributeId *pAttributeId
);
```

Parameters

pValue,

Receives the enumeration's string value.

pOrdinal,

Receives the enumeration's ordinal value.

pPrimitiveId,

Receives primitive ID of the primitive that contains the attribute holding an array of all the possible enumeration string values.

pAttributeId

Receives attribute ID of the attribute holding an array of all the possible enumeration string values.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to a qualified enum.

E_POINTER - An invalid pointer was passed in.

GetDimensionCount method

Indicates whether the MxValue holds an array. MxValue holds an array if PutElement has been called on it and there have been no subsequent calls to Empty or PutXxx (where Xxx is an Integer, Boolean or any of the other data type).

Class

IMxValue

Syntax

[C#]

```
void GetDimensionCount(
   out short nDimensions
);
```

[Visual C++]

```
HRESULT GetDimensionCount(
   [out] short *nDimensions
);
```

Parameters

nDimensions

Returns 1 if the MxValue is an array or 0 if MxValue does not hold an array.

PutElement method

Puts an element into an array held by an MxValue. Call this method consecutively to populate an MxValue with an array of one dimension. The first call on this method will establish the MxDataType for the array. Subsequent calls to this method must supply MxValues of the same MxDataType.

Note: An array's dimension cannot be reduced. Therefore it is not a good idea to cache an MxValue. The array should be destroyed and recreated if its dimension needs to be reduced. Array elements must be written sequentially when initializing array values.

Syntax

[C#]

```
void PutElement(
   int index,
   MxValue pMxValue
);
```

[Visual C++]

```
HRESULT PutElement(
   [in] long index,
   [in] IMxValue *pMxValue
);
```

Parameters

index

Index into the array. Must be between 1 and 2147483647 inclusive.

pMxValue

The value to be put as an array element.

Returns

MX_E_InvalidIndex - Index is out of range.

MX_E_IndexOutOfOrder - Index is not sequential.

MX_E_WrongDataType - A prior call to PutElement supplied an MxValue of a different data type.

GetElement method

Gets an element from an array held in an MxValue.

Class

IMxValue

Syntax

[C#]

```
void GetElement(
   int index1,
   MxValue pMxValue
);
```

[Visual C++]

```
HRESULT GetElement(
   [in] long index1,
   [in] IMxValue *pMxValue
);
```

Parameters

index1

Index into the one-dimensional array.

pMxValue

Receives the specified element's value.

Returns

MX_E_InvalidIndex - Index is out of range.

MX_E_WrongNumberOfDimensions - This is not a one-dimensional array.

GetDimensionSize method

Gets the size of the array held by the MxValue. MxValue determines the size of the array based on the max index value of prior calls to PutElement.

Class

IMxValue

Syntax

[C#]

```
void GetDimensionSize(
   out int pSize
);
```

[Visual C++]

```
HRESULT GetDimensionSize(
   [out] long *pSize
);
```

Parameters

pSize

Receives the specified dimension's size.

PutCustomStructVB method

Stores a qualified struct in the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void PutCustomStructVB(
   int guid,
   ref byte[] pStruct
);
```

[Visual C++]

```
HRESULT PutCustomStructVB(
   [in] long guid,
   [in,out] SAFEARRAY(BYTE) *pStruct
);
```

GetCustomStructVB method

Returns the qualified struct stored in the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void GetCustomStructVB(
   out int pGuid,
   ref byte[] pStruct
);
```

[Visual C++]

```
HRESULT GetCustomStructVB(
   [out] long *pGuid,
   [in,out] SAFEARRAY(BYTE) *pStruct
);
```

PutInternationalStrings method

Initializes an MxValue with a set of internationalized strings.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalStrings(
   int count,
   ref InternationalizedString strings
);
```

[Visual C++]

```
HRESULT PutInternationalStrings(
   [in] long count,
   [in, size_is(count)] struct InternationalizedString strings[]
);
```

Parameters

count

The number of internationalized strings the MxValue is being initialized with.

strings[]

The internationalized strings the MxValue is being initialized with.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

PutInternationalStringsVB method

Initializes an MxValue with a set of internationalized strings for a VB client.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalStringsVB(
   ref InternationalizedString[] ppsa
);
```

[Visual C++]

```
HRESULT PutInternationalStringsVB(
   [in] SAFEARRAY(struct InternationalizedString) *ppsa
);
```

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

GetInternationalStrings method

Returns the internationalized strings stored by the MxValue.

Class

IMxValue

Syntax

[C#]

```
void GetInternationalStrings(
   out int count,
   System.IntPtr locals
);
```

[Visual C++]

```
HRESULT GetInternationalStrings(
   [out] long *count,
   [out, size_is(,*count)] struct InternationalizedString **locals
);
```

Parameters

count

The number of internationalized strings that are stored in the MxValue.

locals

The internationalized strings that are stored in the MxValue.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType - the stored value isn't an internationalized string.

GetInternationalStringsVB method

Returns the internationalized strings stored by the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void GetInternationalStringsVB(
   ref InternationalizedString[] ppsa
);
```

[Visual C++]

```
HRESULT GetInternationalStringsVB(
   [in,out] SAFEARRAY(struct InternationalizedString) *ppsa
);
```

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType- the stored value isn't an internationalized string.

GetInternationalString method

Returns the internationalized string associated with a particular locale. The internationalized string for the locale passed in must have been added to the MxValue via calls to PutInternationalString, PutInternationalStringsVB, or PutInternationalStrings prior to calling this method.

Class

IMxValue

Syntax

[C#]

```
string GetInternationalString(
   int locale
);
```

[Visual C++]

```
HRESULT GetInternationalString(
   [in] long locale,
   [out, retval] BSTR *internationalizedString
);
```

Parameters

locale,

The locale of the internationalized string that is to be retrieved from the MxValue.

internationalizedString

The internationalized string that corresponds to the locale passed in.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType - the stored value isn't an internationalized string.

MX_E_LocaleNotSupported - the internationalized text doesn't exist for the locale passed in.

PutInternationalString method

Adds the internationalized string of a specific locale to the existing internationalized strings. Held by the MxValue. If the locale is already supported by the internationalized string then the string passed in overwrites the existing string for that locale.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalString(
   int locale,
   string InternationalizedString
);
```

[Visual C++]

```
HRESULT PutInternationalString(
   [in] long locale,
   [in] BSTR internationalizedString
);
```

Parameters

locale,

The locale of the internationalized string that is being set into the MxValue.

internationalizedString,

The internationalized string that is to be set into the MxValue.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

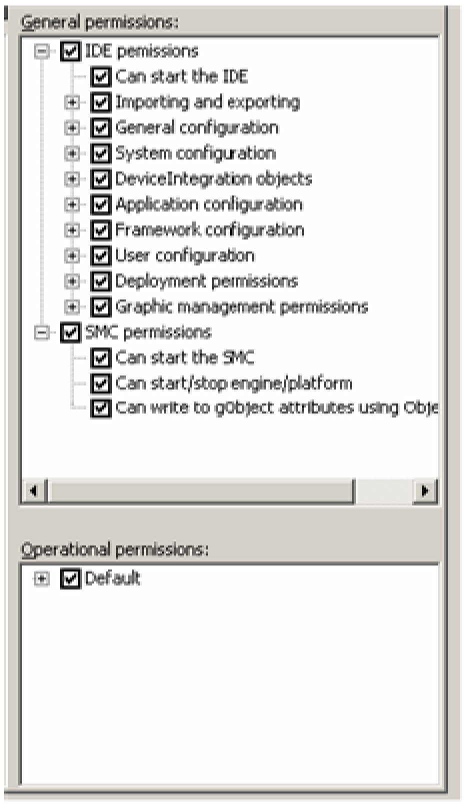
IPermission

IPermission provides read-only access to the properties of a single permission, including name, parent name, whether the permission is configured, if it has any children permissions, collection children permissions, whether it is a security group (root nodes in operation permission are). If the permission is a security group, IPermission can access the security group object.

Each role has two collections of permissions, a collection of general permissions and a collection of operational permissions, from which the client program can drill down to each sub-permission.

For example, for a role configured with the options shown in the sample **Options** dialog box, IPermission returns the following:

Operation	**What it returns**
PermissionName	General Permission
PermissionParentName	Empty string
IsConfigured	Returns VARIANT_TRUE, or VARIANT_FALSE if any of its permission in the tree is configured
HasChildren	Returns VARIANT_TRUE
ChildPermissions	Returns the collection of IPermissions, which in this case is IDE Permissions and SMC Permissions
IsASecurityGroup	Returns VARIANT_FALSE. For "Operational Permissions" children, this property is VARIANT_TRUE.
SecurityGroup	Returns the ISecurityGroup object interface through which the user can access security group related properties. This property is valid for Operational Permission children. In other cases, this returns a NULL pointer.


General permissions:

- ☑ IDE permissions
  - ☑ Can start the IDE
  - ☑ Importing and exporting
  - ☑ General configuration
  - ☑ System configuration
  - ☑ DeviceIntegration objects
  - ☑ Application configuration
  - ☑ Framework configuration
  - ☑ User configuration
  - ☑ Deployment permissions
  - ☑ Graphic management permissions
- ☑ SMC permissions
  - ☑ Can start the SMC
  - ☑ Can start/stop engine/platform
  - ☑ Can write to gObject attributes using Obje

Operational permissions:

- ☑ Default

IPermission Class members

This class has the following members.

Operations

PermissionName property

Returns the permission name.

PermissionParentName property

Returns the name of the permission's parent permission.

IsConfigured property

Returns whether the permission is configured; that is, whether the permission is active.

HasChildren property

Returns true if the permission has child permissions.

ChildPermissions property

Returns the collection of child permissions.

IsSecurityGroup property

Returns true if the permission is a security group.

SecurityGroup property

Retrieves the security group object.

PermissionName property

Returns the name of the permission.

Class

IPermission

Syntax

[C#]

```
string PermissionName { get; };
```

[Visual C++]

```
HRESULT PermissionName(
   [out,retval]BSTR *permissionName
);
```

PermissionParentName property

Returns the name of the permission’s parent, if any. (For top-level permissions, this is an empty string.)

Class

IPermission

Syntax

[C#]

```
string PermissionParentName { get; };
```

[Visual C++]

```
HRESULT PermissionParentName(
   [out,retval]BSTR *permissionParentName
);
```

IsConfigured property

Returns true if the permission is configured; that is, if the permission is active.

Class

IPermission

Syntax

[C#]

```
bool IsConfigured { get; };
```

[Visual C++]

```
HRESULT IsConfigured(
   [out,retval]VARIANT_BOOL *isConfigured
);
```

HasChildren property

Returns true if the permission has child permissions.

Class

IPermission

Syntax

[C#]

```
bool HasChildren { get; };
```

[Visual C++]

```
HRESULT HasChildren(
   [out,retval]VARIANT_BOOL *hasChildren
);
```

ChildPermissions property

Returns the collection of child permissions.

Class

IPermission

Syntax

[C#]

```
IPermissions ChildPermissions { get; };
```

[Visual C++]

```
HRESULT ChildPermissions(
   [out,retval]IPermissions **childPermissions
);
```

IsSecurityGroup property

Returns true if the permission is a security group.

Class

IPermission

Syntax

[C#]

```
bool IsASecurityGroup { get; };
```

[Visual C++]

```
HRESULT IsASecurityGroup(
   [out,retval]VARIANT_BOOL *isSecurityGroup
);
```

SecurityGroup property

Retrieves the security group object.

Class

IPermission

Syntax

[C#]

```
ISecurityGroup SecurityGroup { get; };
```

[Visual C++]

```
HRESULT SecurityGroup(
   [out,retval]ISecurityGroup **secGroup
);
```

IPermissions

Represents a collection of security permissions.

IPermissions class members

This class has the following members.

Operations

Item Property

Returns a permission.

Count Property

Returns the number of permissions in the collection.

Item property

Returns a permission.

Class

IPermissions

Syntax

[C#]

```
IPermission this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IPermission**
);
```

Parameters

galaxyIdentifier

The index of a permission. This is a numeric value from 1 to Count, or the name of the permission.

IPermission

The returned permission.

Count property

Returns the number of permissions in the collection.

Class

IPermissions

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

IScriptLibrary

Represents a script library.

IScriptLibrary class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call's result.

Export method

Export this script library.

Name Property

Returns the name of the script library.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IScriptLibrary

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Export method

Export this script library.

Class

IScriptLibrary

Syntax

[C#]

```
void Export(
   string path
);
```

[Visual C++]

```
HRESULT Export(
   [in] BSTR path
);
```

Parameters

path

Path to copy the .aaslib file.

Name property

Returns the name of the script library.

Class

IScriptLibrary

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* theScriptLibrary
);
```

Parameters

theScriptLibrary

The script library name.

IScriptLibraries

Represents a collection of script libraries.

IScriptLibraries class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call's result.

Item Property

Return a script library.

Count Property

Returns the number of script libraries in the collection.

Add Method

Add a script library to the Galaxy.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IScriptLibraries

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Item property

Returns a script library.

Class

IScriptLibraries

Syntax

[C#]

```
IScriptLibrary this[object scriptLibraryIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT scriptLibraryIdentifier,
   [out, retval] IScriptLibrary**
);
```

Parameters

scriptLibraryIdentifier

The index of a script library. This is a numeric value from 1 to count, or the name of the script library.

IScriptLibrary

The returned script library object.

Count property

Returns the number of script library objects in the collection.

Class

IScriptLibraries

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Add method

Add a new script library to the Galaxy.

Class

IScriptLibraries

Syntax

[C#]

```
void Add(
   string scriptLibraryPath
);
```

[Visual C++]

```
HRESULT Add(
   [in] BSTR scriptLibraryPath
);
```

Parameters

scriptLibraryPath

The name of the new script library.

ISecurityGroup

Accesses a security group object and its properties, including the security group name and the list of gObjects.

ISecurityGroup class members

This class has the following members.

Operations

GroupName property

Returns a security group name.

gObjects property

Returns the list of gObjects in this security group.

GroupName property

Returns the security group name.

Class

ISecurityGroup

Syntax

[C#]

```
string GroupName { get; };
```

[Visual C++]

```
HRESULT GroupName(
   [out, retval] BSTR *groupName
);
```

Parameters

groupName

The returned security group name.

gObjects property

Returns the list of gObjects in this security group.

Class

ISecurityGroup

Syntax

[C#]

```
IgObjects gObjects { get; };
```

[Visual C++]

```
HRESULT gObjects(
   [out, retval] IgObjects** *gObjects
);
```

Parameters

gObjects

The list of gObjects in this security group.

ISecurityGroups

Represents a collection of security groups.

ISecurityGroups class members

This class has the following members.

Operations

Item Property

Returns a security group.

Count Property

Returns the number of security groups in the collection.

Item property

Return a security group.

Class

ISecurityGroups

Syntax

[C#]

```
ISecurityGroup this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] securityGroup**
);
```

Parameters

galaxyIdentifier

The index of a galaxy. This is a numeric value from 1 to count, or the name of the security group.

securityGroup

The returned security group.

Count property

Returns the number of security groups in the collection.

Class

ISecurityGroups

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

Returns the number of security groups in the collection.

ISettings

Represents a settings object. Client configures the data for this settings object using the attributes of the instance property. After completed, the client calls Close to save and check-in the changes, or Cancel to undo the changes. While the settings object is being configured, no other object in the GR may be checked-out.

ISettings class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call’s result.

Cancel method

Cancels the configuration of this object.

Instance property

Gets the instance for configuring this object.

Close method

Saves and checks in the settings object.

CommandResult property

Returns the CommandResult object, which has the last method call’s result.

Class

ISettings

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Cancel method

Cancels the configuration of this object.

Class

ISettings

Syntax

[C#]

```
void Cancel();
```

[Visual C++]

```
HRESULT Cancel();
```

Instance property

Get the instance for configuring this object.

Class

ISettings

Syntax

[C#]

```
IInstance Instance { get; };
```

[Visual C++]

```
HRESULT Instance(
   [out, retval] IInstance** anInstance
);
```

Parameters

anInstance

The instance to be used to configure the object.

Close method

Saves and checks in the settings object.

Class

ISettings

Syntax

[C#]

```
void Close();
```

[Visual C++]

```
HRESULT Close();
```

ITemplate

Represents a Template object.

ITemplate class members

This class has the following members.

Operations

AddExtensionPrimitive method

Adds an extension primitive.

AddUDA method

Adds a UDA.

Area property

Returns or sets the area for this object.

Attributes property

Returns the collection of the object’s attributes.

BasedOn property

Returns the name of the object’s base template.

Category property

Returns the category of this object.

CategoryGUID property

Returns the category GUID for this object.

CheckIn method

Checks in the object.

CheckedOutBy property

Returns the user who has checked out this object.

CheckOut Method

Checks out the object.

CheckoutStatus property

Returns the checkout status of this object.

CommandResult Property

Returns the CommandResult object, which has the last method call results.

CommandResults property

Returns the CommandResults object, which has the last method call's result. Used for CreateInstances.

ConfigurableAttributes property

Returns a collection of the configurable attributes for this object.

ConfigVersion property

Returns the configuration version number.

ContainedName property

Returns or sets the contained name for this object.

Container property

Returns or sets the container for this object.

CreateInstance method

Create a new instance from this template.

CreateInstances method

Creates a collection of new instances from a template.

CreateTemplate method

Derive a new template from this template.

DeleteExtensionPrimitive method

Delete the extension primitive.

DeleteTemplate method

Deletes the template.

DeleteUDA method

Deletes a user defined attribute.

DerivedFrom property

Returns the name of the template this object was derived from.

EditStatus property

Returns the status of this object.

Errors property

Returns a list of errors generated the last time this object was validated and saved.

GetExtendedAttributes method

Returns a list of attributes for this object.

GetObjectHelpURL method

Returns the URL where the object’s help is stored in the Galaxy repository.

HierarchicalName property

Returns the hierarchical name of this object.

Host property

Returns the name of this object’s host.

RenameExtensionPrimitive method

Change the name of an extension primitive..

RenameUDA method

Changes the name of a user defined attribute.

Save method

Saves an object.

Tagname property

Returns the tagname of this object.

Toolset property

Returns the Toolset that this template belongs to.

UndoCheckOut method

Changes the object’s checked out status without saving changes.

Unload method

Unloads the object’s cache. The object remains usable.

UpdateUDA method

Change a user defined attribute.

ValidationStatus property

Returns the validation status information from the last time the object was validated and saved.

Warnings property

Returns a list of warnings from the last time the object was validated and saved.

AddExtensionPrimitive method

Adds an extension primitive to this object.

Class

ITemplate

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of extension primitive to be added -
   // For attribute extension primitive - Fully
   // qualified name of attribute being extended
   // For object extension primitive - New name
   // specified by user for object extension
   // primitive
   [in] BSTR ExtensionPrimitiveName,
   // If wszName is the new name specified by user,
   // this field will be set to true. For example,
   //when add script pass true
   [in, optional, defaultvalue( -1 )]VARIANT_BOOL IsObjectExtension
);
```

AddUDA method

Adds a UDA.

Class

ITemplate

Syntax

[C#]

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Add a UDA
//
// Returns:
// S_OK - Successfully add a UDA
HRESULT AddUDA(
   // Same XML format. But only one is allowed.
   [in] BSTR UDAName,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Area property

Get or set the area of this object. An AppObject area must be set using the SetContainer or SetHost methods. The object does not need to be checked out, and the operation is immediate.

Class

ITemplate

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
// Returns name of the Area object, or blank if the
// object doesn't have an Area.
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
// Set the Area of this object.
[propput]
HRESULT Area(
   // The name of the Area, or blank to unassign
   // from existing Area.
   [in] BSTR NewArea
);
```

Attributes property

Returns the collection of attributes of this object.

Class

ITemplate

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object.
// Configuring attributes requires that the object
// be checked out.
HRESULT Attributes(
   // The collection of Attributes of this object.
   [out, retval] IAttributes** theAttributes
);
```

BasedOn property

Returns the name of the base template for this object.

Class

ITemplate

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
// The name of the base template (root ancestor) of
// this object.
HRESULT BasedOn(
   // The name of the base template (root ancestor)
   // of this object.
   [out, retval] BSTR* theBaseTemplate
);
```

Category property

Returns the category of this object.

Class

ITemplate

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
// The Category of the object.
HRESULT Category(
   // The Category of the object.
   [out, retval] enum ECATEGORY* theCategory
);
```

CategoryGUID property

Gets the category GUID of this object.

Class

ITemplate

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
// The Category GUID of the object.
HRESULT CategoryGUID(
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

CheckIn method

Check in this object.

Class

ITemplate

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
// Checks in an object.
// Called on a checked-out object after it is
// configured and saved.
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

CheckOut method

Checks out an object.

Class

ITemplate

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
// Checks out an object.
HRESULT CheckOut();
```

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

ITemplate

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
// The name of the user that this object is checked
// out by.
HRESULT CheckedOutBy(
   // The name of the user that this object is
   // checked out by.
   [out, retval] BSTR* CheckedOutByUserName
);
```

CheckoutStatus property

Returns the check out status of the object.

Class

ITemplate

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
// The check-out status of the object.
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

CommandResult property

Returns the CommandResult object, which has the results from the last method call.

Class

ITemplate

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
// Returns the CommandResult Object, which has last
// method call result
HRESULT CommandResult(
   // Returns the CommandResult
   [out, retval] ICommandResult** CommandResult
);
```

CommandResults property

Returns the CommandResults object, which has the last method call's result. Used for CreateInstances

Class

ITemplate

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

Parameters

CommandResults

Returns the CommandResults object.

ConfigurableAttributes property

Returns a collection of this object's attributes that can be configured.

Class

ITemplate

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object that
// can be configured.
// Configuring attributes requires that the object
// be checked out.
HRESULT ConfigurableAttributes(
   // The collection of Attributes of this object
   // that can be configured.
   [out, retval] IAttributes** theConfigurableAttributes
);
```

ConfigVersion property

Returns this object's configuration version number. The configuration version is incremented each time the object is checked in.

Class

ITemplate

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

```
// The configuration version of the object.
HRESULT ConfigVersion(
   // The configuration version of the object.
   [out, retval] LONG* theConfigVersion
);
```

ContainedName property

Sets or returns this object’s contained name. When setting the contained name, the object does not need to be checked out, and the operation is immediate.

Class

ITemplate

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
//
// Returns Contained name of this object (eg. Inlet)
[propget]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [out, retval] BSTR* theContainedName
);
// Sets the contained name of this object (eg. Inlet)
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [in] BSTR newContainedName
);
```

Container property

Returns or sets the container for this object. When setting the container, the object does not need to be checked out, and the change is immediate. For an ApplicationObject, the ApplicationObject’s Area and Host area are also updated.

Class

ITemplate

Syntax

[C#]

```
string Container { set; get; };
```

[Visual C++]

```
// Returns name of the Container object, or blank if
//object is not contained.
[propget]
HRESULT Container(
   // The name of the Container object, or blank if
   // the object is not contained.
   [out, retval] BSTR* theContainer
);
// Set the Container for this object.
[propput]
HRESULT Container(
   // The name of the container, or blank to uncontain.
   [in] BSTR NewContainer
);
```

CreateInstance method

Create a new Instance from this template.

Class

ITemplate

Syntax

[C#]

```
IInstance CreateInstance(
   string Name,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateInstance(
   [in] BSTR Name,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] IInstance**
);
```

Parameters

Name

The name of the newly created instance.

createContainedObjects

Set to True to create contained objects.

IInstance

The instance that was created.

CreateInstances method

Creates a collection of new instances from a template.

Class

ITemplate

Syntax

[C#]

```
IgObjects CreateInstances(
   ref string[] tagnames,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateInstances(
   [in, out] SAFEARRAY(BSTR)* tagnames,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] IgObjects**
);
```

Parameters

tagnames

An array of unique tagnames for the newly created instances.

createContainedObjects

Set to true to create contained objects.

IgObjects

The instances that were created.

CreateTemplate method

Derive an a new template from this template.

Class

ITemplate

Syntax

[C#]

```
ITemplate CreateTemplate(
   string Name,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateTemplate(
   [in] BSTR Name,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] ITemplate**
);
```

Parameters

Name

The name of the newly created template.

createContainedObjects

Set to true to create contained objects.

ITemplate

The template that was created.

DeleteExtensionPrimitive method

Deletes an extension.

Class

ITemplate

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
// Delete an extension primitive
//
// Returns S_OK
HRESULT DeleteExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of Extension primitive to be deleted.
   [in] BSTR ExtensionPrimitiveName
);
```

DeleteTemplate method

Deletes the template.

Class

ITemplate

Syntax

[C#]

```
void DeleteTemplate(
   EForceDeleteTemplateOption ForceDeleteTemplateOption
);
```

[Visual C++]

```
HRESULT DeleteTemplate(
   [in, optional, defaultvalue( dontForceTemplateDelete )] EForceDeleteTemplateOption
   ForceDeleteTemplateOption
);
```

Parameters

ForceDeleteTemplateOption

Indicates whether descendant instances should be undeployed prior to deletion.

DeleteUDA method

Deletes a UDA.

Class

ITemplate

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

```
// Delete a UDA
//
// Returns:
// S_OK - Successfully delete a UDA
HRESULT DeleteUDA(
   // Delete an UDA by UDA name.
   [in] BSTR UDAName
);
```

DerivedFrom property

Returns the name of the object's template.

Class

ITemplate

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
// Returns the name of the template that this object
// was created from.
[propget]
HRESULT DerivedFrom(
   // The name of the template that this object was
   // created from.
   [out, retval] BSTR* theParentTemplate
);
```

EditStatus property

Edit the status of the object.

Class

ITemplate

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
// The edit status of the object.
HRESULT EditStatus(
   // The edit status of the object.
   [out, retval] enum EEditStatus* theEditStatus
);
```

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

ITemplate

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
// The list of errors that got generated the last
// time the object was validated and saved.
HRESULT Errors(
   // The list of errors that was generated the last
   // time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

GetExtendedAttributes method

Get the extended attributes in the hierarchy.

Class

ITemplate

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

```
//Get Extended Attributes in Hierarchy
HRESULT GetExtendedAttributes(
   // Attribute full name which is extended
   [in] BSTR AttributeName,
   //Up to how many level user needs extended
   //primitives
   //-1 will return all levels
   [in] int upto_level,
   //Return attributes which are only these
   //categories. If safearray is empty returns all
   //attributes.
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   //Return attributes
   [out,retval]IAttributes** theAttributes
);
```

GetObjectHelpURL method

Returns the URL where the object help is stored on the Galaxy repository.

Class

ITemplate

Syntax

[C#]

```
string GetObjectHelpURL();
```

[Visual C++]

```
// Returns the URL to where Object Help is
// stored on the Galaxy Repository.
// The client can display the URL in
// InternetExplorer or use web publishing
// to change the data.
HRESULT GetObjectHelpURL(
   // Returns the URL to where the Object Help is
   //stored on the GR.
   [out, retval] BSTR*
);
```

HierarchicalName property

Returns the hierarchical name of this object.

Class

ITemplate

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
// The Hierarchical name of this object. (eg.
// Reactor.Tank.Inlet)
HRESULT HierarchicalName(
   // The hierarchical name of this object. (eg.
   // Reactor.Tank.Inlet)
   [out, retval] BSTR* theHierarchicalName
);
```

Host property

Returns or sets the name of this object's host.

Class

ITemplate

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
// Retrieve the name of the Host for this object, or
// blank if unassigned.
[propget]
HRESULT Host(
   // The name of the Host for this object.
   [out, retval] BSTR* theHost
);
// Set the Host for this object.
// If this object is an ApplicationObject, its Area
// and container area also updated.
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT Host(
   // The name of the Host, or blank to unassign
   // from existing Host.
   [in] BSTR NewHost
);
```

RenameExtensionPrimitive method

Rename the object extension primitive.

Class

ITemplate

Syntax

[C#]

```
void RenameExtensionPrimitive(
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
// Rename an object extension primitive
//
// Returns S_OK
HRESULT RenameExtensionPrimitive(
   // Old name of primitive extension
   [in] BSTR OldPrimitiveName,
   // New name specified by user
   [in] BSTR NewPrimitiveName
);
```

RenameUDA method

Change the name of a UDA.

Class

ITemplate

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
// Rename a UDA
//
// Returns:
// S_OK - Successfully rename a UDA
HRESULT RenameUDA(
   // The old external name of the primitive
   [in] BSTR OldUDAName,
   // The new external name of the primitive
   [in] BSTR NewUDAName
);
```

Save method

Save the object.

Class

ITemplate

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
// Saves the object after it is configured.
// Object must be checked out.
HRESULT Save();
```

Tagname property

Returns or sets this object’s tagname.

Class

ITemplate

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
// Returns the Tagname of this object.
[propget]
HRESULT Tagname(
   // The Tagname of this object.
   [out, retval] BSTR* theTagname
);
// Sets the Tagname of this object.
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT Tagname(
   // The Tagname of this object.
   [in] BSTR newTagName
);
```

Toolset property

Returns the Toolset that this template belongs to.

Class

ITemplate

Syntax

[C#]

```
string Toolset { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Toolset(
   [out, retval] BSTR* theToolset
);
[propput]
HRESULT Toolset(
   [in] BSTR theToolset
);
```

Parameters

theToolset

The Toolset that this template belongs to.

UndoCheckOut method

Reverses the check out operation without making changes.

Class

ITemplate

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
// Undo check-out this object.
HRESULT UndoCheckOut();
```

Unload method

Unload the object cache, releasing internal resources obtained by the object during operations. After executing this method, the object is still available. Internal resources will be reloaded as necessary.

Class

ITemplate

Syntax

[C#]

```
void Unload();
```

[Visual C++]

```
// Unloads the gObject cache. This method does not
// affect functionality. It is used to release
// internal resources obtained by the object during
// operations.
// After this method is called, the object is still
// usable and will reload the internal resources as
// needed.
HRESULT Unload();
```

UpdateUDA method

Save the UDA information.

Class

ITemplate

Syntax

[C#]

```
void UpdateUDA(
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Update a UDA
// Returns:
// S_OK - Successfully update a UDA
HRESULT UpdateUDA(
   // Same XML format. But only one is allowed. The
   // new info will overwrite the old info.
   [in] BSTR wszUDAInfo,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

ValidationStatus property

Returns this object’s validation status.

Class

ITemplate

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
// The ValidationStatus of this object based on the
// last time the object was validated and saved.
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

ITemplate

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
// The list of warnings that was generated the last
// time the object was validated and saved.
HRESULT Warnings(
   // The list of warnings that was generated the
   //last time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

IToolset

Represents a toolset.

IToolset class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult object, which has the last method call’s result.

GetChildToolsets method

Retrieves the list of child toolsets.

MoveToToolset method

Moves the toolset to a new parent toolset.

Name property

Returns the name of the toolset.

Rename method

Renames this toolset.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IToolset

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

GetChildToolsets method

Retrieves the list of child toolsets.

Class

IToolset

Syntax

[C#]

```
IToolsets GetChildToolsets(
   int depth
);
```

[Visual C++]

```
// Retrieves the list of child Toolsets.
HRESULT GetChildToolsets(
// the level of children to retrieve: pass -1 for N.
   [in] int depth,
   [out, retval] IToolsets** toolsets
);
```

MoveToToolset method

Moves the toolset to a new parent toolset.

Class

IToolset

Syntax

[C#]

```
void MoveToToolset(
   string parentToolset
);
```

[Visual C++]

```
// Moves toolset to a new parent toolset
HRESULT MoveToToolset(
   [in] BSTR parentToolset
);
```

Name property

Returns the name of the toolset.

Class

IToolset

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* theToolset
);
```

Parameters

theToolset

The toolset name.

Rename method

Renames this toolset.

Class

IToolset

Syntax

[C#]

```
void Rename(
   string newName
);
```

[Visual C++]

```
HRESULT Rename(
   [in] BSTR newName
);
```

Parameters

newName

The new name of the toolset.

IToolsets

Represents a collection of toolsets.

IToolsets class members

This class has the following members.

Operations

Add method

Add a new toolset to the Galaxy.

AddToolSet method

Add a new toolset to the Galaxy.

DeleteToolSet method

Delete the toolset from the Galaxy.

Item property

Returns a toolset.

Count property

Returns the number of toolsets in the collection.

Add method

Adds a new toolset to the Galaxy.

Class

IToolsets

Syntax

[C#]

```
void Add(
   string newToolset
);
```

[Visual C++]

```
HRESULT Add(
   [in] BSTR newToolset
);
```

Parameters

newToolset

The name of the new toolset.

AddToolSet method

Adds the toolset to the Galaxy.

Class

IToolsets

Syntax

[C#]

```
void AddToolSet(
   string parentToolset,
   string newToolset
);
```

[Visual C++]

```
// Add a new toolset to the Galaxy.
HRESULT AddToolSet(
   [in] BSTR parentToolset,
   [in] BSTR newToolset
);
```

DeleteToolSet method

Deletes the toolset from the Galaxy.

Class

IToolsets

Syntax

[C#]

```
void DeleteToolSet(
   string toolsetToDelete
);
```

[Visual C++]

```
// Delete an existing toolset from the Galaxy.
HRESULT DeleteToolSet(
   [in] BSTR toolsetToDelete
);
```

Item property

Returns a toolset.

Class

IToolsets

Syntax

[C#]

```
IToolset this[object toolsetIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT toolsetIdentifier,
   [out, retval] IToolset**
);
```

Parameters

toolsetIdentifier

The index of a toolset. This is a numeric value from 1 to Count. Or it is the name of the toolset.

IToolset

The returned toolset.

Count property

Returns the number of toolsets in the collection.

Class

IToolsets

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Type definitions

This section describes the enumerations defined and exposed through GRAccess. These include:

EATTRIBUTEPROPERTY

EActionForCurrentlyDeployedObjects

EAuthenticationMode

EAutomaticallyUndocheckout

EBASERUNTIMEOBJECT

ECATEGORY

ECOMMONATTRIBUTES

ECOMMONPRIMITIVE

ECascade

ECheckoutStatus

EConditionType

EDeployOnScan

EDeploymentStatus

EEXECUTIONGROUP

EEditStatus

EExecutionOrder

EExportType

EFileType

EForceDeleteInstanceOption

EForceDeleteTemplateOption

EForceOffScan

EGRCommandResult

EMatch

EPACKAGESTATUS

ERESERVEDPRIMITIVEIDS

ESkipIfCurrentlyUndeployed

ESkipObjectsWithPendingUpdates

ESkipOtherUsersCheckedOutObjects

ETestOnly

EUserDefault

EgObjectIsTemplateOrInstance

MxAttributeCategory

MxPropertyLockedEnum

EATTRIBUTEPROPERTY

Lists property IDs of all attributes.

enum **EATTRIBUTEPROPERTY**

```
{idxAttribPropUndefined = -1
idxAttribPropName = 0,
idxAttribPropAttributeCategory = 1,
idxAttribPropCfgSethandler = 2,
idxAttribPropRtSethandler = 3,
idxAttribPropType = 4,
idxAttribPropUpperBoundDim1 = 5,
idxAttribPropValue = 10,
idxAttribPropSecurityClassification = 11,
idxAttribPropLocked = 12,
idxAttribPropQuality = 13,
idxAttribPropBit00 = 14,
idxAttribPropBit01 = 15,
idxAttribPropBit02 = 16,
idxAttribPropBit03 = 17,
idxAttribPropBit04 = 18,
idxAttribPropBit05 = 19,
idxAttribPropBit06 = 20,
idxAttribPropBit07 = 21,
idxAttribPropBit08 = 22,
idxAttribPropBit09 = 23,
idxAttribPropBit10 = 24,
idxAttribPropBit11 = 25,
idxAttribPropBit12 = 26,
idxAttribPropBit13 = 27,
idxAttribPropBit14 = 28,
idxAttribPropBit15 = 29,
idxAttribPropBit16 = 30,
idxAttribPropBit17 = 31,
idxAttribPropBit18 = 32,
idxAttribPropBit19 = 33,
idxAttribPropBit20 = 34,
idxAttribPropBit21 = 35,
idxAttribPropBit22 = 36,
idxAttribPropBit23 = 37,
idxAttribPropBit24 = 38,
idxAttribPropBit25 = 39,
idxAttribPropBit26 = 40,
idxAttribPropBit27 = 41,
idxAttribPropBit28 = 42,
idxAttribPropBit29 = 43,
idxAttribPropBit30 = 44,
idxAttribPropBit31 = 45,
idxAttribPropValueReadOnly = 46,
idxAttribPropSecurityClassificationReadOnly = 47,
idxAttribPropLockedReadOnly = 48,
idxAttribPropTime = 49,
idxAttribPropBuffer = 50,
idxAttribPropHasBuffer = 51,
idxAttribPropEND = 46
}
```

Members

idxAttribPropType

MxDataTypeEnum

idxAttribPropUndefined

idxAttribPropUpperBoundDim1

MxInteger

idxAttribPropQuality

MxDataQualityEnum

idxAttribPropAttributeCategory

MxInteger

idxAttribPropLocked

MxInteger

idxAttribPropBit05

idxAttribPropBit02

idxAttribPropBit00

idxAttribPropBit01

idxAttribPropBit03

idxAttribPropBit04

idxAttribPropBit06

idxAttribPropBit07

idxAttribPropBit08

idxAttribPropBit09

idxAttribPropBit10

idxAttribPropBit11

idxAttribPropBit12

idxAttribPropBit13

idxAttribPropBit14

idxAttribPropBit15

idxAttribPropBit16

idxAttribPropBit17

idxAttribPropBit18

idxAttribPropBit19

idxAttribPropBit20

idxAttribPropBit21

idxAttribPropBit22

idxAttribPropBit23

idxAttribPropSecurityClassification

MxSecurityClassificationEnum

idxAttribPropBit24

idxAttribPropBit25

idxAttribPropBit26

idxAttribPropBit27

idxAttribPropBit28

idxAttribPropBit29

idxAttribPropBit30

idxAttribPropRtSethandler

MxBoolean

idxAttribPropBit31

idxAttribPropValueReadOnly

MxBoolean

idxAttribPropSecurityClassificationReadOnly

MxBoolean

idxAttribPropLockedReadOnly

MxBoolean

idxAttribPropTime

MxDouble

idxAttribPropBuffer

MxBoolean

idxAttribPropHasBuffer

MxCustomStruct

idxAttribPropEND

idxAttribPropValue

Depends on datatype

idxAttribPropName

MxString

idxAttribPropCfgSethandler

MxBoolean

EActionForCurrentlyDeployedObjects

Action for currently deployed objects.

enum **EActionForCurrentlyDeployedObjects**

{

```
skipDeploy = 0,
redeployOriginal = 1,
deployChanges = 2
}
```

Members

deployChanges

Deploy updated versions.

skipDeploy

Skip if currently deployed.

redeployOriginal

redeploy orignal version.

EAuthenticationMode

Indicates whether operating system or Galaxy authentication should be used.

enum **EAuthenticationMode**

{

```
osAuthenticationMode = 0,
galaxyAuthenticationMode = 1,
}
```

Members

galaxyAuthenticationMode

Galaxy Authentication Mode

osAuthenticationMode

OS Authentication Mode

EAutomaticallyUndocheckout

Automatically undoes checkout.

enum **EAutomaticallyUndocheckout**

{

```
dontAutomaticallyUndocheckout = 0,
doAutomaticallyUndocheckout = 1
```

}

Members

dontAutomaticallyUndocheckout

Fail if any object is checked out to the current user.

doAutomaticallyUndocheckout

Automatically undoes checkout of any check-outs to the current user before processing.

EBASERUNTIMEOBJECT

Specifies the IDs of the attributes of the base run time object to be used for deployment.

enum **EBASERUNTIMEOBJECT**

```
{
idxBROAttributeUndefined = -1,
idxBROPrimitiveList = 100,
idxBRODeployComplete = 101,
idxBROCreateDynamicAttributes = 102,
idxBRODumpState = 103
```

}

Members

idxBROAttributeUndefined

idxBROPrimitiveList

Array of MxQualifiedStruct - primitiveId, parentId, primitiveGUID.

idxBRODeployComplete

MxBoolean

idxBROCreateDynamicAttributes

idxBRODumpState

ECATEGORY

Lists categories of attributes.

enum **ECATEGORY**

{

```
idxCategoryUndefined = 0,
idxCategoryPlatformEngine = 1,
idxCategoryClusterEngine = 2,
idxCategoryApplicationEngine = 3,
idxCategoryViewEngine = 4,
idxCategoryProductEngine = 5,
idxCategoryHistoryEngine = 6,
idxCategoryPrintEngine = 7,
idxCategoryOutpost = 8,
idxCategoryQueryEngine = 9,
idxCategoryApplicationObject = 10,
idxCategoryIONetwork = 11,
idxCategoryIODevice = 12,
idxCategoryArea = 13,
idxCategoryUserProfile = 14,
idxCategoryDisplay = 15,
idxCategorySymbol = 16,
idxCategoryViewApp = 17,
idxCategoryProductionObject = 18,
idxCategoryReport = 19,
idxCategorySharedProcedure = 20,
idxCategoryInsertablePrimitive = 21,
idxCategoryIDEMacro = 22,
idxCategoryGalaxy = 23
}
```

Members

idxCategoryViewApp

idxCategoryViewEngine

idxCategoryOutpost

idxCategoryApplicationObject

idxCategoryUserProfile

idxCategoryReport

idxCategoryGalaxy

idxCategoryPlatformEngine

idxCategoryIDEMacro

idxCategoryUndefined

idxCategoryHistoryEngine

idxCategoryPrintEngine

idxCategoryClusterEngine

idxCategoryDisplay

idxCategoryProductEngine

idxCategoryApplicationEngine

idxCategorySymbol

idxCategoryProductionObject

idxCategoryInsertablePrimitive

idxCategoryQueryEngine

idxCategoryIODevice

idxCategoryArea

idxCategorySharedProcedure

idxCategoryIONetwork

ECOMMONATTRIBUTES

Lists the attributes common to all primitives.

enum **ECOMMONATTRIBUTES**

{

```
idxComAttUnknown = -1,
idxComAttExternalName = 1,
idxComAttInternalName = 2
}
```

Members

idxComAttExternalName

idxComAttInternalName

idxComAttUnknown

ECOMMONPRIMITIVE

Specifies the IDs of the attributes of the common primitive.

enum **ECOMMONPRIMITIVE**

```
{
idxCommonUndefined = -1,
idxCommonExternalName = 1,
idxCommonInternalName = 2,
idxCommonTagname = 100,
idxCommonShortDescription = 101,
idxCommonCategoryIDEnum = 102,
idxCommonAttributes = 103,
idxCommonCmdScanState = 104,
idxCommonScanState = 105,
idxCommonSecurityGroup = 106,
idxCommonError = 107,
idxCommonRelativeOrderEnum = 108,
idxCommonArea = 109,
idxCommonContainer = 110,
idxCommonHost = 111,
idxCommonAlarmModeEnum = 112,
idxCommonAlarmMode = 113,
idxCommonAlarmModeCmd = 114,
idxCommonAlarmInhibit = 115,
idxCommonInAlarm = 116,
idxCommonBasedOn = 117,
idxCommonIsTemplate = 118,
idxCommonCodeBase = 119,
idxCommonUDAs = 120,
idxCommon_InheritedUDAs = 121,
idxCommonExtensions = 122,
idxCommon_InheritedExtensions = 123,
idxCommon_Category = 124,
idxCommon_ConfigVersion = 125,
idxCommon_Warnings = 126,
idxCommonContainedName = 127,
idxCommonExecutionRelatedObject = 128,
idxCommonExecutionRelativeOrder = 129,
idxCommonHierachicalName = 130,
idxCommonMinorVersion = 131
}
```

Members

idxCommonAlarmMode

idxCommonBasedOn

idxCommonExecutionRelatedObject

idxCommonContainedName

idxCommonInAlarm

idxCommonCategoryIDEnum

idxCommonCategoryEnum

idxCommonExecutionRelativeOrder

idxCommonUndefined

idxCommonSecurityGroup

idxCommon_Warnings

idxCommonAlarmModeEnum

idxCommonInternalName

idxCommon_InheritedUDAs

idxCommon_InheritedExtensions

idxCommonShortDescription

idxCommonExtensions

idxCommonHierachicalName

idxCommonExternalName

idxCommonCodeBase

idxCommonRelativeOrderEnum

idxCommonExecutionRelativeOrderEnum

idxCommonUDAs

idxCommonContainer

idxCommonAlarmInhibit

idxCommonArea

idxCommonHost

idxCommonTagname

idxCommonError

idxCommonErrors

idxCommonAttributes

idxCommon_Category

idxCommonCmdScanState

idxCommonScanStateCmd

idxCommon_ConfigVersion

this is for ConfigVersion

idxCommonScanState

idxCommonScanStateActual

idxCommonMinorVersion

idxCommonAlarmModeCmd

idxCommonIsTemplate

ECascade

Cascade operation.

enum **ECascade**

{

```
dontCascade = 0,
doCascade = 1,
```

}

Members

dontCascade

Don't cascade

doCascade

Do force off scan

ECheckoutStatus

Indicates the check-out status of an object.

enum ECheckoutStatus

{

```
notCheckedOut = 0,
checkedOutToMe = 1,
checkedOutToSomeoneElse = 2
```

}

Members

notCheckedOut

The object is not checked-out.

checkedOutToSomeoneElse

The object is checked out to someone else.

checkedOutToMe

The object is checked out the currently logged-in user.

EConditionType

Used by the querying methods to get a list of objects from the GR. Indicates the type of condition used by the query.

enum **EConditionType**

{

```
derivedOrInstantiatedFrom = 1,
basedOn = 2,
containedBy = 3,
hostEngineIs = 4,
belongsToArea = 5,
assignedTo = 6,
withinSecurityGroup = 7,
createdBy = 8,
lastModifiedBy = 9,
checkedOutBy = 10,
namedLike = 11,
validationStatusIs = 12,
deploymentStatusIs = 13,
checkoutStatusIs = 14,
objectCategoryIs = 15,
hierarchicalNameLike = 16,
NameEquals = 17
NameSpaceldls = 18
```

}

Members

derivedOrInstantiatedFrom	Search for objects that are derived or instantiated from the given template (first level children only.)
objectCategoryIs	Search for objects with the check-out status. Value parameter should be the numeric value represented by ECATEGORY.
deploymentStatusIs	Search for objects with the given deployment status. Value parameter should be the numeric value represented by EDeploymentStatus.
hierarchicalNameLike	Search for objects whose hierarchical name is like the given text.
belongsToArea	Search for objects whose Area is the given object.
withinSecurityGroup	Search for objects that match the given security group.
lastModifiedBy	Search for objects that were modified by the given user.
checkedOutBy	Search for objects that are checked out by the given user.
validationStatusIs	Search for objects with the given validation status. Value parameter should be the numeric value represented by EPACKAGESTATUS.
hostEngineIs	Search for objects hosted by the given engine (Domain objects of an AppEngine or Engines of a Platform.)
namedLike	Search for objects whose name is like the given text.
NameEquals	Search for objects whose name is equal to the given text.
NameSpaceIdIs	Search for NameSpaceId equal to the value of AutomationObject = 1, BackupObject = 2 or VisualElement = 3
containedBy	Search for objects contained by the given object.
checkoutStatusIs	Search for objects with the check-out status. Value parameter should be the numeric value represented by ECheckoutStatus.
createdBy	Search for objects that were created by the given user.
assignedTo	Search for objects hosted by the given object (ApplicationObjects or IONetworks of an Area, IODevices of an

IONetwork, Domain objects of an AppEngine, or Engines of a Platform)

basedOn

Search for objects that are based on or instantiated from the given base template (all descendents).

ShowInArchestraBrowser

Search for Objects that "show_ in_archestra_browser" = = given value

objectCategoryGUIDIs

Search Objects with "object category guid" = = given value.

HasExtensionType

Search automation objects for extension type. Possible values are:

"alarmextension"	"analogextension"
"AttributeExtension"	"badvaluealarmextension"
"booleanextension"	"DisplayExtension"
"historyextension"	"inputextension"
"inputoutputextension"	"logdatachangeeventextension"
"ObjectExtension"	"outputextension"
"ScriptExtension"	"SymbolExtension"

EDeployOnScan

Deploy on scan.

enum **EDeployOnScan**

{

```
dontDeployOnScan = 0,
doDeployOnScan = 1,
```

}

Members

doDeployOnScan

Do deploy on scan.

dontDeployOnScan

Don't deploy on scan.

EDeploymentStatus

Indicates the whether the object is deployed.

enum **EDeploymentStatus**

{

```
   notDeployed = 0,
   deployed = 1,
   deployedWithPendingChanges = 2
```

}

Members

notDeployed

The object is not deployed.

deployed

The object is deployed.

deployedWithPendingChanges

The object is deployed and it has pending changes.

EEXECUTIONGROUP

Lists ExecutionGroup options.

enum **EEXECUTIONGROUP**

{

```
idxExeGroupUnknown = -1,
idxExeGroupCommon = 0,
idxExeGroupInput = 1,
idxExeGroupPreScript = 2,
idxExeGroupCustom = 3,
idxExeGroupCustom1 = 4,
idxExeGroupCustom2 = 5,
idxExeGroupCustom3 = 6,
idxExeGroupCustom4 = 7,
idxExeGroupCustom5 = 8,
idxExeGroupCustom6 = 9,
idxExeGroupCustom7 = 10,
idxExeGroupCustom8 = 11,
idxExeGroupCustom9 = 12,
idxExeGroupCustom10 = 13,
idxExeGroupExpression = 14,
idxExeGroupScript = 15,
idxExeGroupPostScript = 16,
idxExeGroupOutput = 17,
idxExeGroupHistory = 18,
idxExeGroupAlarm = 19,
```

}

Members

idxExeGroupCustom6

idxExeGroupCustom4

idxExeGroupCustom5

idxExeGroupCustom2

idxExeGroupCustom3

idxExeGroupCustom1

New Default

idxExeGroupCommon

idxExeGroupCustom7

idxExeGroupCustom8

idxExeGroupCustom9

idxExeGroupScript

idxExeGroupOutput

idxExeGroupExpression

idxExeGroupInput

idxExeGroupPostScript

idxExeGroupCustom10

idxExeGroupCustom

idxExeGroupAlarm

idxExeGroupPreScript

idxExeGroupHistory

idxExeGroupUnknown

EEditStatus

Indicates the edit status of an object.

enum **EEditStatus**

{

```
notBeingEdited = 0,
editedByThisSession = 1,
editedByYouInOtherSession = 2,
editedByAnotherUser = 3,
```

}

Members

editedByThisSession

The object is being edited by this session.

editedByYouInOtherSession

The object is being edited by you in another session.

notBeingEdited

The object is not being edited.

editedByAnotherUser

The object is being edited by another user.

EExecutionOrder

Indicates the execution order of an object as related to the object specified by ExecutionRelatedObject.

enum EExecutionOrder

{

```
The execution order doesn't matter and is determined by the Engine hosting the object.
NONE = 1,
The object should run before the object referenced by ExecutionRelatedObject.
BEFORE,
The object should run after the object referenced by ExecutionRelatedObject.
AFTER
```

}

Members

NONE

BEFORE

AFTER

EExportType

Indicates the type of export: aaPKG.

enum **EExportType**

{

```
exportAsPDF = 0,
exportAsCSV = 1,
```

}

Members

exportAsPDF

Export to a .PDF file. The resulting cab file contains a list of all templates and instances (along with required templates) plus any code modules.

exportAsCSV

Export instances to a text .CSV which can be opened in Excel.

EFileType

Describes the type of file that is in the PIM file repository. PIM uses this to determine how to install the file.

enum **EFileType**

{

```
eUndefinedFileType = 0,
eNormal = 1,
eComDLL = 2,
eComEXE = 3,
eNTService = 4,
eMergeRegistryScript = 5,
eForHTMLEditor = 6,
eDictionary = 7,
eMsiMergeModule = 8,
eNETFrameworkAssembly
```

}

Members

eForHTMLEditor

File is an HTML editor code module file. Store in Vendorname\Primitive GUID subdirectory instead of Vendorname subdirectory.

eNormal

Normal file. Do nothing extra except install file.

eMsiMergeModule

File is an MSI merge module, which will be merged into an installable MSI file.

eComEXE

File is a COM local server .EXE. Run "<filename> /RegServer".

eUndefinedFileType

eComDLL

File is a COM in-proc server .DLL. Run regsvr32 on the file.

eNETFrameworkAssembly

File is a .NET Framework assembly.

eDictionary

File is an ArchestrA Dictionary file.

eNTService

File is an NT Service .EXE. Run "<filename> -Service".

eMergeRegistryScript

File is a .REG registry script. Merge it into system registry.

EForceDeleteInstanceOption

Indicates whether a deployed instance should be undeployed prior to deletion.

enum **EForceDeleteInstanceOption**

{

```
dontForceInstanceDelete = 0,
undeployIfNotRunningOnScan = 1,
undeployIfDeployed = 2
```

}

Members

undeployIfDeployed

If object is deployed, set it off-scan if needed, and undeploy prior to deletion.

dontForceInstanceDelete

Don't force the deletion of the instance.

undeployIfNotRunningOnScan

If object is deployed, undeploy it prior to deletion. Object must be off-scan.

EForceDeleteTemplateOption

Indicates how to handle descendents prior to deletion.

enum **EForceDeleteTemplateOption**

{

```
dontForceTemplateDelete = 0,
cascadeDeleteDontUndeploy = 1,
cascadeDeleteUndeployDescendentsIfRunningOffScan = 2,
cascadeDeleteUndeployDescendentsIfRunning = 3
```

}

Members

cascadeDeleteUndeployDescendentsIfRunning

Prior to deletion, decendent instances are first undeployed (and forced off-scan if needed) and then deleted.

cascadeDeleteUndeployDescendentsIfRunningOffScan

Prior to deletion, decendent instances are first undeployed and then deleted. Undeployed objects must be off-scan.

cascadeDeleteDontUndeploy

Do delete descendent objects first, but only if they're not deployed.

dontForceTemplateDelete

Don't force the deletion of the template. The template cannot have any derived descendents.

EForceOffScan

Force off-scan.

enum **EForceOffScan**

{

```
dontForceOffScan = 0,
doForceOffScan = 1,
```

}

Members

dontForceOffScan

Don't force off-scan.

doForceOffScan

Do force off-scan.

EGRCommandResult

Indicates the result of a command issued to the GR.

enum **EGRCommandResult**

{

```
cmdUnknownError = -1,
cmdSuccess = 0,
cmdInsufficientPermissions = 101,
cmdNoSuchGRNode = 102,
cmdNoSuchUser = 103,
cmdPasswordIncorrect = 104,
cmdLicenseUnavailable = 105,
cmdNoSuchFile = 201,
cmdCouldntCreateFile = 202,
cmdObjectIsCheckedOut = 301,
cmdObjectIsCheckedOutToSomeoneElse = 302,
cmdObjectNotCheckedOutToMe = 303,
cmdObjectCannotBeOverwritten = 401,
cmdTemplateInUse = 402,
cmdObjectIsAContainer = 403,
cmdObjectHostNotFound = 404,
cmdInstanceIsHost = 405,
cmdObjectIsRequired = 406,
cmdObjectNotAnAutomationObject = 407,
cmdObjectInBadState = 408,
cmdCustomConfigurationError = 409,
cmdObjectInReadOnlyMode = 410,
cmdObjectHostNotDeployed = 501,
cmdInstanceIsDeployed = 502,
cmdInvalidGRLoadMode = 503,
cmdOEMVersionIncompatible = 504,
cmdRetryClientSync = 505,
cmdSyncSemaphore = 506,
cmdOutOfSync = 507
```

}

Members

cmdLicenseUnavailable

Operation failed because a license problem occurred.

cmdCustomConfigurationError

The operation failed because of configuration error.

cmdObjectHostNotFound

The operation failed because the given instance does not have a host.

cmdInsufficientPermissions

Operation failed because the user has insufficient permission.

cmdNoSuchGRNode

Operation failed because the given GR Node is invalid.

cmdObjectIsAContainer

The operation failed because the object is a container of other objects.

cmdObjectHostNotDeployed

The operation failed because the host of the given instance is not deployed.

cmdNoSuchFile

The specified file was not found.

cmdObjectNotAnAutomationObject

The operation failed because the object is not an AutomationObject.

cmdObjectIsRequired

The object could not be deleted because it is a required object.

cmdSuccess

Success code.

cmdUnknownError

Unknown error.

cmdObjectInBadState

The operation failed because the object is in a bad state.

cmdObjectCannotBeOverwritten

The specified object already exists and cannot be overwritten.

cmdObjectIsCheckedOutToSomeoneElse

The operation failed because the object is currently checked out to someone else.

cmdObjectInReadOnlyMode

The operation failed because the object is in read-only mode.

cmdTemplateInUse

The operation failed because the template has one or more objects derived from it.

cmdInstanceIsHost

The operation failed because the instance is a host of other objects.

cmdCouldntCreateFile

The operation failed because the file could not be created.

cmdInstanceIsDeployed

The operation failed because the instance is currently deployed.

cmdObjectNotCheckedOutToMe

The operation failed because the object is not checked out to the current user.

cmdObjectIsCheckedOut

The operation failed because the object is currently checked out.

cmdNoSuchUser

Operation failed because the given user name is invalid.

cmdPasswordIncorrect

Operation failed because the given password is invalid.

cmdInvalidGRLoadMode

The operation failed because enum GRLoad mode is invalid.

cmdOEMVersionIncompatible

The operation failed because client and galaxy OEMVersions are incompatible.

cmdRetryClientSync

The operation failed because enum cmdRetryClientSync mode is invalid.

cmdSyncSemaphore

The operation failed because enum cmdForceSynchronize mode is invalid.

cmdOutOfSync

The operation failed because enum cmdOutOfSync mode is invalid.

EMatch

Indicates which results should be returned: condition match or negate result.

enum **EMatch**

{

```
NotMatchCondition = 0,
MatchCondition = 1
```

}

Members

MatchCondition

Condition match results.

NotMatchCondition

Condition match negate results.

EPACKAGESTATUS

Indicates the status of an object after it is validated.

enum **EPACKAGESTATUS**

{

```
Package configured properly.
ePackageUnknownStatus = -1,
The object is properly configured and can be deployed.
```

```
ePackageGood = 0,
The object is not properly configured, but it can be deployed.
ePackageBad = 1,
The object is not properly configured, and cannot be deployed.
ePackageWarning = 2
```

}

Members

ePackageBad

ePackageGood

ePackageWarning

ePackageUnknownStatus

ERESERVEDPRIMITIVEIDS

Specifies the primitive IDs of the common primitives and the topmost custom primitive.

enum **ERESERVEDPRIMITIVEIDS**

{

```
eReservedPrimitiveIdUndefined = -1,
eBaseRuntimeObjectId = 1,
eCommonPrimitiveId = 2,
eTopLevelCustomPrimitiveId = 100
```

}

Members

eCommonPrimitiveId

eTopLevelCustomPrimitiveId

eReservedPrimitiveIdUndefined

eBaseRuntimeObjectId

Primitive ID reserved for BRO use during deployment.

ESkipIfCurrentlyUndeployed

Skip if currently undeployed.

enum **ESkipIfCurrentlyUndeployed**

{

```
dontSkipIfCurrentlyUndeployed = 0,
doSkipIfCurrentlyUndeployed = 1,
```

}

Members

doSkipIfCurrentlyUndeployed

Do skip if currently undeployed.

dontSkipIfCurrentlyUndeployed

Don't skip if currently undeployed.

ESkipObjectsWithPendingUpdates

Skip objects with pending updates.

enum **ESkipObjectsWithPendingUpdates**

{

```
dontSkipObjectsWithPendingUpdates = 0,
doSkipObjectsWithPendingUpdates = 1
```

}

Members

dontSkipObjectsWithPendingUpdates

Upload will fail if any object has pending updates.

doSkipObjectsWithPendingUpdates

Automatically skip objects that have pending updates.

ESkipOtherUsersCheckedOutObjects

Skip other users' CheckedOuts.

enum **ESkipOtherUsersCheckedOutObjects**

{

```
dontSkipOtherUsersCheckedOutObjects = 0,
doSkipOtherUsersCheckedOutObjects = 1
```

}

Members

doSkipOtherUsersCheckedOutObjects

Automatically skip objects that are checked out to another user.

dontSkipOtherUsersCheckedOutObjects

Upload will fail if any objects are checked out to other users.

ETestOnly

Indicates whether the operation is to be performed or just tested to determine if it can be performed.

enum **ETestOnly**

{

```
performOperation = 0,
testOnly = 1,
```

}

Members

performOperation

Perform the operation.

testOnly

Only test to determine if the operation can be performed.

EUserDefault

The user default.

enum **EUserDefault**

{

```
defaultPlatform = 1,
defaultAppEngine = 2,
defaultHistorianNode = 3,
defaultArea = 4,
defaultSecurityGroup = 5
```

}

Members

defaultSecurityGroup	The name of the default security group of the user.
defaultHistorianNode	The name of the historian node name of the user.
defaultAppEngine	The name of the default AppEngine of the user.
defaultArea	The name of the default Area of the user.
defaultPlatform	The name of the default Platform of the user.

EgObjectIsTemplateOrInstance

Indicates whether a gObject is a template or an instance.

enum **EgObjectIsTemplateOrInstance**

{

```
gObjectIsTemplate = 0,
gObjectIsInstance = 1,
```

}

Members

gObjectIsTemplate

GObject is a Template.

gObjectIsInstance

GObject is a Instance.

MxAttributeCategory

Indicates the category of an attribute.

enum **MxAttributeCategory**

{

```
MxCategoryUndefined = -1,
Can only be configured at IDE time and not Lockable. Attribute doesn't exist at Runtime.
MxCategoryPackageOnly = 0,
Can only be configured at IDE time and is Lockable. Attribute doesn't exist at Runtime.
(eg. CmdDisagreeAlarm.Available)
MxCategoryPackageOnly_Lockable = 1,
Not configurable at IDE time. Calculated and readable at runtime, and not Checkpointed.
(eg. Valve1.PVAuto)
MxCategoryCalculated = 2,
Not configurable at IDE time. Calculated and readable at runtime and Checkpointed. (eg.
Timer or Counter Accumulator)
MxCategoryCalculatedRetentive = 3,
Only the Security Classification is configurable at IDE time. User writable but not
Supervisory writable. Checkpointed. (eg. Valve1.CmdTImeoutAlarm.Acknowledged)
MxCategoryWriteable_U = 4,
Not configurable at IDE time. Supervisory writable but not User writable. Checkpointed.
MxCategoryWriteable_S = 5,
Only the Security Classification is configurable at IDE time. User and Supervisory writable
and Checkpointed. (eg. Valve1.CMD, PID1.SP)
MxCategoryWriteable_US = 6,
Can be configured at IDE time but not lockable. User writable but not Supervisory writable.
Checkpointed. (eg. Valve1.PVOverrideEnabled)
MxCategoryWriteable_UC = 7,
Can be configured at IDE time but not lockable. User and Supervisory writable and
Checkpointed. (eg. Valve1.PVMode)
MxCategoryWriteable_USC = 8,
Can be configured at IDE time and is Lockable. User writable but not Supervisory writable.
```

```
Checkpointed. (eg. Valve1.FieldInputToPVMap)
MxCategoryWriteable_UC_Lockable = 9,
Can be configured at IDE time and is Lockable. User and Supervisory writable and
Checkpointed. (eg. Valve1.CmdTImeoutAlarm.Limit)
MxCategoryWriteable_USC_Lockable = 10,
Can be configured at IDE time and is Lockable. Not writable at runtime or checkpointed.
(eg. Valve1.NumInputs)
MxCategoryWriteable_C_Lockable = 11,
Not configurable or Lockable at IDE time. User or Supervisory writable, but System
writable. Not checkpointed. (eg. GREngine._BindReference)
MxCategory_SystemSetsOnly = 12,
Not configurable or Lockable at IDE time. User or Supervisory writable, but System
writable. Not readable at Runtime. Not checkpointed. (eg. _InternalName, _ExternalName)
MxCategory_SystemInternal = 13,
Not configurable or Lockable, but readable at IDE time. Not writable but readable at
Runtime. Not checkpointed. (eg. Tagname)
MxCategory_SystemWriteable = 14,
Defined at SDK time and not modified after that.
MxCategory_Constant = 15,
Defined at Config time. This attribute does not exist at runtime
MxCategory_SystemInternal_Browsable = 16,
Defined at SDK time. This attribute does not exist at runtime
MxCategory_CalculatedPackage = 17,
Defined at SDK time. This attribute is used to send data to runtime.
MxCategory_DeleteAfterStartup = 18,
MxAttributeCategoryEND = 19
```

}

Members

MxCategoryCalculated

MxCategoryWriteable_USC

MxCategoryWriteable_UC_Lockable

MxCategoryWriteable_USC_Lockable

MxCategoryWriteable_C_Lockable

MxCategoryCalculatedRetentive

MxCategoryUndefined

MxCategory_SystemWriteable

MxCategory_Constant

MxCategory_SystemInternal_Browsable

MxCategoryWriteable_US

MxCategoryWriteable_UC

MxCategory_SystemSetsOnly

MxCategory_CalculatedPackage

MxCategory_DeleteAfterStartup

MxAttributeCategoryEND

MxCategoryPackageOnly_Lockable

MxCategory_SystemInternal

MxCategoryPackageOnly

MxCategoryWriteable_U

MxCategoryWriteable_S

MxPropertyLockedEnum

Indicates the lock status of an attribute.

enum **MxPropertyLockedEnum**

{

```
MxUndefinedLockedStatus = -1,
The Attribute is Unlocked. Data can be modified.
MxUnLocked = 0,
The Attribute is Locked in this object and propagated to all descendents. Data can be
modified.
MxLockedInMe = 1,
The Attribute is LockedByMe in an ancestor object. Data cannot be modified.
MxLockedInParent = 2,
MxPropertyLockedEnumEND = 3
```

}

Members

MxUndefinedLockedStatus

MxLockedInParent

MxPropertyLockedEnumEND

MxLockedInMe

MxUnLocked

GRAccess code examples

This section provides examples related to performance and other best practices.

GRAccess code examples are located in the following folder:

C:\Program Files (x86)\ArchestrA\Samples

The code examples are:

- CreateGalaxyCPP.

  See Example: define the entry point for a console application
- CreateGalaxyCSharp

  See Example: create and populate a galaxy
- GUISampleApp

  The GUISampleApp implements all aspects of the GRAccess API, and is intended only to demonstrate the functions that can be used in a GRAccess application. It allows you, as a customer or system integrator, to examine the source code in Visual Studio, and to map the results of UI interaction to specific GRAccess API calls.

WARNING! The GUISampleApp is not intended as a production-ready application. **DO NOT** use the GUISampleApp with production galaxies. It is intended only to give you insight into the GRAccess source code, and let you see how user actions within the GUI affect GRAccess API calls.

Programming tips

Follow these programming tips:

- GRAccessApp m_gr = new GRAccessApp(). m_gr object must be active when you want to use any of GRAccess Objects. This means that it should not go out of scope from your code path.
- CommandResult and CommandResults are error checking methods like GetLastError in C++ and The Err Object in VB. After each GRAccess object API call, you can check that the API succeeded or not. APIs that deal with multiple objects return CommandResults.
- If you are using GRAccess from a remote machine, make sure you have run the OSConfigUtility.
- If needed, modify the import statement in stdafx.h to include (x86) in the file path, as shown in the following example:

```
#import "C:\Program Files (x86)\ArchestrA\Framework\Bin\ArchestrA.GRAccess.dll"
no_namespace, raw_interfaces_only
```

- The import path may vary depending on variables such as the InTouch installation directory. Modify the import statement in stdafx.h to import from the correct library path, as shown in the following example:

```
#import "C:\Program Files (x86)\Wonderware\InTouch\AppServerSecurity.dll"
no_namespace, raw_interfaces_only
```

Example: define the entry point for a console application

The following example shows the entry point for the console application with ATL support:

Note: This code can be found in C:\Program Files (x86)\ArchestrA\Samples\CreateGalaxyCPP.

```
// ------------------------------------------------------
// <copyright company="AVEVA Software, LLC" file="CreateGalaxyCPP.cpp">
// © 2022 AVEVA Software, LLC. All rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
// </copyright>
// </summary>
//------------------------------------------------------
// CreateGalaxyCPP.cpp : Defines the entry point for the console application with ATL
support.
//
#include "stdafx.h"
#include <time.h>
#import "GRAccess111.tlb" no_namespace
```

```
class COMINIT
{
public:
COMINIT()
   {
      // Initialize COM.
      comInit = CoInitializeEx(
            0,
         //COINIT_APARTMENTTHREADED
            COINIT_APARTMENTTHREADED | COINIT_DISABLE_OLE1DDE
            );
   }
   ~COMINIT()
   {
      // Uninitialize COM.
      if (SUCCEEDED(comInit))
      {
         CoUninitialize();
      }
   }
private:
   HRESULT comInit;
};
int _tmain(int argc, _TCHAR* argv[])
{
   COMINIT SmartComInitialize;
   CComPtr<IGRAccess> spGRApp;
   HRESULT hr = spGRApp.CoCreateInstance ( __uuidof(GRAccessApp));
   if (spGRApp == NULL)
      return -1;
   CComPtr<IGalaxies> spGalaxies;
   DWORD dwsize = MAX_COMPUTERNAME_LENGTH + 1;
   TCHAR szGRNodeComputerName[MAX_COMPUTERNAME_LENGTH + 1];
   ::GetComputerName(szGRNodeComputerName, &dwsize);
   CComBSTR bstrGRNodeComputerName = szGRNodeComputerName ;
   spGalaxies = spGRApp->QueryGalaxies (bstrGRNodeComputerName.m_str);
   CComPtr<ICommandResult> spCommandResult;
   spGRApp->get_CommandResult (&spCommandResult);
   if(spCommandResult->Successful == VARIANT_FALSE || spGalaxies == NULL)
   {
      wprintf(L"%s : %s \n",spCommandResult->CustomMessage.GetBSTR (), spCommandResult-
      >Text.GetBSTR ());
      return 1;
   }
   CComPtr<IGalaxy> spGalaxy;
   _variant_t vtGalName = L"Example1";
   //OR you can use index if you have only one galaxy use 1. All collections are 1 base
   //_variant_t vtGalName = L"1";
   spGalaxies->get_Item (vtGalName,&spGalaxy);
   if(spGalaxy == NULL)
   {
         spGRApp->CreateGalaxy (vtGalName.bstrVal
         ,bstrGRNodeComputerName.m_str,FALSE,galaxyAuthenticationMode,L"");
            CComPtr<ICommandResult> spCommandResult;
            spGRApp->get_CommandResult (&spCommandResult);
            if(spCommandResult->Successful == VARIANT_FALSE || spGalaxies == NULL)
            {
            wprintf(L"Create Galaxy Named Example1 Failed: %s : %s \n",spCommandResult-
            >CustomMessage.GetBSTR (), spCommandResult->Text.GetBSTR ());
            return 1;
         }
         spGalaxies = spGRApp->QueryGalaxies (bstrGRNodeComputerName.m_str);
         spGalaxies->get_Item (vtGalName,&spGalaxy);
      }
      spGalaxy->Login (L"",L"");
      CComPtr<IgObjects> spGObjects;
      _variant_t vtTemplateName = L"$UserDefined";
      spGObjects = spGalaxy->QueryObjects
      (gObjectIsTemplate,NameEquals,&vtTemplateName,MatchCondition );
      CComPtr<IgObject> spGObject;
      spGObjects->get_Item (vtTemplateName,&spGObject);
      CComPtr<IInstance> spInstance;
      CComQIPtr<ITemplate> spTemplate = spGObject;
      time_t ltime;
      time( &ltime );
      WCHAR UTCStr [32];
      wsprintf(UTCStr,L"%ld",ltime);
      _bstr_t instanceName(L"UD_");
      instanceName += UTCStr;
      spInstance = spTemplate->CreateInstance (instanceName,VARIANT_FALSE);
      spInstance->CheckOut ();
      _variant_t vtCount = 5;
      spInstance->AddUDA
      (L"Names",MxString,MxCategoryWriteable_USC_Lockable,MxSecurityOperate,VARIANT_TRUE,vt
      Count);
      CComPtr<IAttributes> spAttributes;
      spInstance->get_ConfigurableAttributes (&spAttributes);
      CComPtr<IAttribute> spAttrib;
      _variant_t vtAttrName = L"Names";
      spAttributes->get_Item (vtAttrName,&spAttrib);
      //Diplay first 5 attribute names from collection
      for (int i = 1; i <= 5; i++)
      {
         CComPtr<IAttribute> spAttrib1 = spAttributes->GetItem(i);
         wprintf(L"%s\n",spAttrib1->GetName().GetBSTR());
      }
      CComPtr<IMxValue> spNamesValue;
      spNamesValue.CoCreateInstance (__uuidof(MxValue));
      for(int i=1;i<=5;i++)
      {
         CComPtr<IMxValue> mxv;
         mxv.CoCreateInstance (__uuidof(MxValue));
         mxv->PutInteger (i);
         // Attribute Data type is string Why Integer ??
         // The reason is MxValue (Just Like Variant) is smart which contains logic
         // to convert from one type to another.
         spNamesValue->PutElement (i,mxv);
      }
      spAttrib->SetValue (spNamesValue);
      spInstance->Save ();
      spInstance->CheckIn (L"");
      spGalaxy->Logout ();
      WCHAR dummy [81];
      wprintf (L"\nPress ENTER to quit: ");
      _getws_s (dummy, 80);
      //char ch = getchar();
      return 0;
   }
```

Example: create and populate a galaxy

The steps below demonstrate how to use GRAccess through C# and Visual Studio.

This procedure creates an ArchestrA Galaxy and populates it with a single instance.

Build and execute the program below. It creates a Galaxy called "Example1" and a single instance of a $UserDefined template with a name based on the current date and time.

Note: The C# console application should work with ATL support.

1. Using Visual Studio, create a new C# console application "CreateGalaxyExample".
2. On the **Main** menu, click **Project / Add Reference** and add a reference to the GRAccess primary interop assembly. By default, this is: C:\Windows\assembly\GAC_MSIL\ArchestrA.GRAccess\2.0.0.0__23106a86e706d0ae\ArchestrA.GRAccess.dll
3. Replace the contents of the generated C# source file with the following text:

   Note: This code can be found in C:\Program Files (x86)\ArchestrA\Samples\CreateGalaxyCSharp.

```
//
------------------------------------------------------------------------------------------
-----------------
// <copyright company="AVEVA Software, LLC" file="Main.cs">
// © 2022 AVEVA Software, LLC. All rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
// </copyright>
// <summary>
//
// </summary>
//
------------------------------------------------------------------------------------------
-----------------
using System;
using ArchestrA.GRAccess;
class CreateGalaxyExample
{
   [STAThread]
   static void Main()
   {
      string nodeName = Environment.MachineName;
      string galaxyName = "Example1";
      // create GRAccessAppClass object
      GRAccessApp grAccess = new GRAccessAppClass();
      // try to get galaxy
```

```
        IGalaxies gals = grAccess.QueryGalaxies(nodeName);
        if (gals == null || grAccess.CommandResult.Successful == false)
        {
    Console.WriteLine(grAccess.CommandResult.CustomMessage + grAccess.CommandResult.Text);
            return;
        }
        IGalaxy galaxy = gals[galaxyName];
        ICommandResult cmd;
        // create galaxy if it doesn't already exist
        if( galaxy == null )
        {
            grAccess.CreateGalaxy(
                galaxyName,
                nodeName,
                false, // no security
                EAuthenticationMode.galaxyAuthenticationMode,
                "" );
            cmd = grAccess.CommandResult;
            if (!cmd.Successful)
            {
                Console.WriteLine("Create Galaxy Named Example1 Failed: " +
                cmd.Text + " : " +
                cmd.CustomMessage);
                return;
        }
        galaxy = grAccess.QueryGalaxies( nodeName )[ galaxyName ];
    }
    // log in
    galaxy.Login( "", "" );
    cmd = galaxy.CommandResult;
    if (!cmd.Successful)
    {
        Console.WriteLine("Login to galaxy Example1 Failed :" +
                cmd.Text + " : " +
                cmd.CustomMessage);
        return;
    }
    // get the $UserDefined template
    string [] tagnames = { "$UserDefined" };
    IgObjects queryResult = galaxy.QueryObjectsByName(
        EgObjectIsTemplateOrInstance.gObjectIsTemplate,
        ref tagnames );
    cmd = galaxy.CommandResult;
    if (!cmd.Successful)
    {
        Console.WriteLine("QueryObjectsByName Failed for $UserDefined Template :" +
                cmd.Text + " : " +
                cmd.CustomMessage);
        return;
    }
    ITemplate userDefinedTemplate = (ITemplate) queryResult[1];
    // create an instance of $UserDefined, named with current time
    DateTime now = DateTime.Now;
    string instanceName = String.Format( "sample_object_{0}_{1}_{2}"
          , now.Hour.ToString( "00" )
          , now.Minute.ToString( "00" )
          , now.Second.ToString( "00" ));
   IInstance sampleinst = userDefinedTemplate.CreateInstance(instanceName, true );
   //How to edit the object ?
   sampleinst.CheckOut();
   sampleinst.AddUDA("Names",
         MxDataType.MxString,
         MxAttributeCategory.MxCategoryWriteable_USC_Lockable,
         MxSecurityClassification.MxSecurityOperate,
         true,
         5);
   IAttributes attrs = sampleinst.ConfigurableAttributes;
   //Diplay first 5 attribute names from collection
   for (int i = 1; i <= 5; i++)
   {
      IAttribute attrb = attrs[i];
      Console.WriteLine(attrb.Name);
   }
   IAttribute attr1 = attrs["Names"];
   MxValue mxv = new MxValueClass();
   // we don't need to check that attribute is array type or not
   // because we set it as array type when we addUDA.
   // I am just showing example, you can do like this.
   if (attr1.UpperBoundDim1 > 0)
   {
      for (int i = 1; i <= attr1.UpperBoundDim1; i++)
      {
         MxValue mxvelement = new MxValueClass();
         mxvelement.PutString("string element number " + i.ToString());
         mxv.PutElement(i, mxvelement);
      }
      attr1.SetValue(mxv);
   }
   sampleinst.Save();
   sampleinst.CheckIn("Check in after addUDA");
   galaxy.Logout();
   Console.WriteLine();
   Console.Write("Press ENTER to quit: ");
   string dummy;
   dummy = Console.ReadLine();
   }
}
```

AVEVA™

AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com
To find your local AVEVA office, visit **www.aveva.com/offices**