CA DataMinder

External Agent COM API Reference Guide

Release 14.5

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

CA Technologies Product References

This document references the following CA Technologies products:

- CA DataMinder

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 6: API Interfaces: Generating EVF Files 49

Chapter 7: API Typedefs 59

Chapter 8: Input Combinations for ImportObject() Method 63

Chapter 9: Common Parameter Details 67

Chapter 1: Introduction

Note: The interface defined in this specification uses Microsoft COM so is only suitable for use on Windows platforms. A socket-based API is also available for CA DataMinder; this is described in the *CA DataMinder External Agent Socket API,* available from CA Technical Support: http//ca.com/support http://ca.com/support

The External Agent API facilitates third party integration with CA DataMinder and smart tag metadata. Specifically, it enables third party applications to pass messages to CA for policy processing and, if required, to store the resulting smart tag metadata with the archived message. Events corresponding to these policy-processed messages can also be stored on a CMS, from where they can be retrieved, reviewed and audited using the iConsole or Data Management console.

Note use of the following terms:

- *Caller* is a third party application passing e-mails to the External Agent. The caller is sometimes also referred to as the 'calling application' or 'client'.
- *Savable e-mail* is an e-mail that would normally be saved in the CMS database. Specifically, while being processed by a policy engine the e-mail activated a policy trigger that was configured to save e-mail events in the CMS database.
- *Smart tags* are used by CA DataMinder to categorize captured events.

This section contains the following topics:

Which Interfaces Do I Use?

The External Agent includes two sets of interfaces, available by different COM objects:

- **WgnActiveImportConnector:** Use these interfaces to apply policy to messages passed to the External Agent, to optionally create CA DataMinder events and store them on the CMS, and to retrieve any associated smart tags. See the next section for details.
- **WgnImportConnector:** Use these interfaces to generate EVFs (CA event files) for messages passed from a third party archive.

In both cases, the External Agent is not a separate process, but runs as a DLL inside the archive solution's process. It starts automatically when it is called by the archive solution's process. Also, the actual message data is generally not saved permanently on the CMS; instead, to minimize storage requirements, a record in the CMS database for each imported e-mail references the associated entry in the e-mail store (for WgnActiveImportConnector, this requires that the caller passes a remote data location to the External Agent).

More information:

WgnImportConnector Object (see page 13)

WgnActiveImportConnector Object

The interfaces related to the WgnActiveImportConnector COM object are:

- IWgnActiveImportConnector
- IWgnImportConnectorConfig
- IWgnImportConnectorResult
- IWgnImportConnectorCallback
- IWgnImportConnectorDeferredCompletion
- IWgnSmartTag

These interfaces provide customers with extensive access to CA DataMinder's policy processing capabilities. Using these interfaces, third party applications can pass messages to the External Agent for policy processing, and receive back the resulting smart tags associated with each message.

If required, the third party application can then store smart tag metadata with the archived messages. The smart tags returned by the External Agent can also be used to identify messages that must be deleted and not archived. Finally, the External Agent can generate CA DataMinder events from the processed messages and store these events on a CMS.

More information:

API Interfaces: Applying Policy and Generating CA DataMinder Events (see page 33)

Synchronous Versus Asynchronous Operations

WgnActiveImportConnector can be called in synchronous or asynchronous mode. The mode is determined initially by the method called in the IWgnActiveImportConnector interface: ImportObject() and ImportObjectAsync().

- **Synchronous mode:** If an e-mail is passed synchronously to the External Agent by calling ImportObject(), the call is **blocked** while the External Agent processes the e-mail. The call is only returned when processing is complete. The External Agent then returns any smart tags for that e-mail to the calling application.
- **Asynchronous mode:** The calling application passes e-mails to the External Agent by calling ImportObjectAync(), but the call is **not** blocked while the External Agent processes the e-mail. The caller must pass the IWgnImportConnectorCallback interface to the External agent so it can inform the caller of the results when processing is complete.

After smart tags are returned to the caller, what happens next depends on whether deferred completion is enabled.

Single-pass Processing Versus Deferred Completion

Calls to WgnActiveImportConnector can use single-pass processing or two-pass processing (that is, the caller can enable 'deferred completion'. Deferred completion allows the caller greater control over which events are saved in the CMS database.

- **Single-pass processing:** When the call to ImportObject() or ImportObjectAync() returns, policy processing for that e-mail is fully complete and the e-mail may already have been saved in the CMS database (if the e-mail activated any policy triggers configured to capture events).
- **Deferred completion:** Also known as 'two-pass processing'. When deferred completion is enabled, events cannot be saved to the CMS until the caller has assessed the outcome of any policy processing. This allows the caller to filter out communications that do not need archiving (for example, because they are exempt from compliance regulations).

For example, smart tags are applied to an e-mail. When the External Agent returns these smart tags to the calling application, the caller uses these smart tags to categorize the e-mail. If the e-mail is in a category that does not require archiving (such as an Outlook meeting request), the caller can delete the e-mail and signal to the External Agent that the e-mail does not need saving in the CMS database.

Deferred completion is controlled by the allowDeferredCompletion parameter in the ImportObject() and ImportObjectAsync() methods. If deferred completion:

- **Is** enabled, events processed by the policy engine will not be committed to the CMS database until the caller has processed the results of policy analysis. The caller calls methods in the IWgnImportConnectorDeferredCompletion interface to instruct the External Agent to save (**commit**) or not save (**rollback**) the e-mail to the CMS database.
- Is **not** enabled, events processed by the policy engine can be committed to the CMS database without waiting for the caller to process the results of policy analysis.

When deferred completion is enabled, the e-mail can be committed synchronously or asynchronously.

- **Synchronous:** If the e-mail is committed synchronously by calling Commit(), the call is **blocked** while the e-mail is committed. The call is only returned when processing is complete.
- **Asynchrounous:** If the e-mail is committed asynchronously by calling CommitAsync(), the call will **not** block while the External Agent processes the commit. The caller must pass the IWgnImportConnectorCallback interface to the External Agent API so it can inform the caller of the result of the commit.

Important! The synchronous mode of the commit must match that of the import. So if an e-mail is imported using ImportObject(), it must be committed using Commit(), and if an e-mail is imported using ImportObjectAsync(), it must be committed using CommitAsync().

WgnImportConnector Object

The interfaces available via the WgnImportConnector COM object are:

- IWgnImportConnector
- IWgnImportConnector2
- IWgnImportConnector3
- IWgnRDIConfig

These interfaces enable CA DataMinder to integrate with third party e-mail archives. The third party archive solution uses a proprietary interface to pass e-mails to the External Agent, which then converts them to CA DataMinder event files (EVF files) and saves them to a cache. This is typically a shared network folder that provides the source data for the CA DataMinder Event Import utility, which then imports the e-mails as EVF files into the CMS.

More information:

API Interfaces: Generating EVF Files (see page 49)

Interface Versions

Interfaces for generating EVF files (via WgnImportConnector) are periodically superseded, with the new interface typically allowing for a wider variety of imported message types and formats and permitting additional message data to be imported. This technical note covers three interfaces for importing messages. The latest of these, IWgnImportConnector3, supersedes the two earlier interfaces, IWgnImportConnector and IWgnImportConnector2.

Note: We recommend that you always use the latest interface, but we will not remove old interfaces without issuing a prior warning to customers.

More information:

Interface IWgnImportConnector (see page 49)
Interface IWgnImportConnector2 (see page 51)
Interface IWgnImportConnector3 (see page 54)

WgnImportObject Object

In addition to the import connector objects, the External Agent API also provides a WgnImportObject CoClass that can be used to pre-create objects for importing. This is particularly useful when importing files, because additional data streams can be added to the file object prior to importing. WgnImportObject currently supports Smart Tags.

Smart Tags

Smart tags enable CA DataMinder to accurately categorize events at the time of capture. They are added to an event when a trigger activates. To apply smart tags, you must configure a trigger to detect all e-mails with a particular theme and tag them accordingly, for example, as Personal Communication or Client-Attorney Privilege. Smart tags are saved with the event metadata in the CMS database and can be viewed subsequently in the iConsole by reviewers. For full details, see the *Administrator Guide*; search for 'smart tags'.

Chapter 2: Deployment Details

This section summarizes the requirements and initialization details for the External Agent.

This section contains the following topics:

Using the External Agent from a Third Party Application

Add the following to your C++ source file:

```
#import "WgnRDI.dll" named_guids raw_interfaces_only
```

External Agent Initialization

WgnRDI.dll is loaded in-process and expects any necessary initialization to have been performed by the process. For example:

- All import operations assume CoInitialize() has been called on the calling thread.
- MAPI import operations assume MAPIInitialize has been called at least once per process.
- Notes import operations assume NotesInitThread has been called on the calling thread.
- For the External Agent API to work with a Policy Engine or Hub (when using the IWgnActiveImportConnector interface), CoInitializeSecurity() must be called by the EAAPI client to allow the PE/Hub to call back into the client process (either by using a NULL security descriptor to allow access to everyone, or a security descriptor giving execute rights to the PE and/or Hub user).

Chapter 3: External Agent Requirements

Note the following requirements for the External Agent API:

Email solution

The External Agent host machine must have Microsoft Outlook or Lotus Notes installed.

Microsoft **Outlook**

Microsoft Outlook 2003, 2007, 2010, or 2013

Outlook *must* be the default email application on the host machine.

Lotus Notes

Lotus Notes 7, 8, or 8.5

File system

(Applicable to the WgnImportConnector interfaces only) For NTFS file systems, we recommend that you disable creation of 8.3 file names on the machine hosting the External Agent cache folder. If 8.3 file creation remains enabled, performance may be adversely affected.

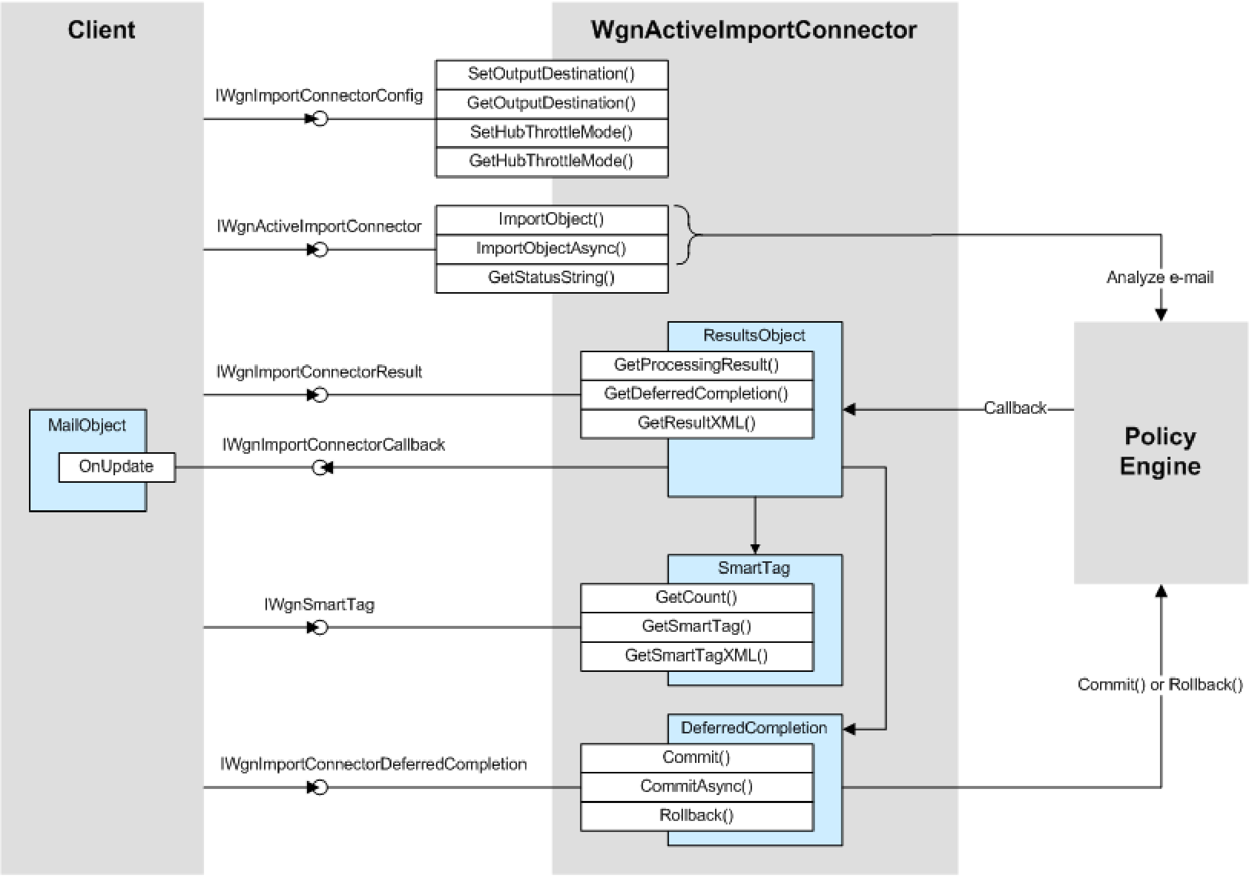
To disable 8.3 file creation, set the registry value NtfsDisable8dot3NameCreation to 1 (one); for further details, search for this registry value on www.support.microsoft.com http://www.support.microsoft.com.

Chapter 4: Integration Procedures

This section summarizes the internal architecture of the External Agent. A series of scenarios illustrate the sequence in which interfaces are used when a message is passed to the External Agent for processing.

This section contains the following topics:

External Agent API Architecture



CA DataMinder External Agent: Interfaces →○ and methods()

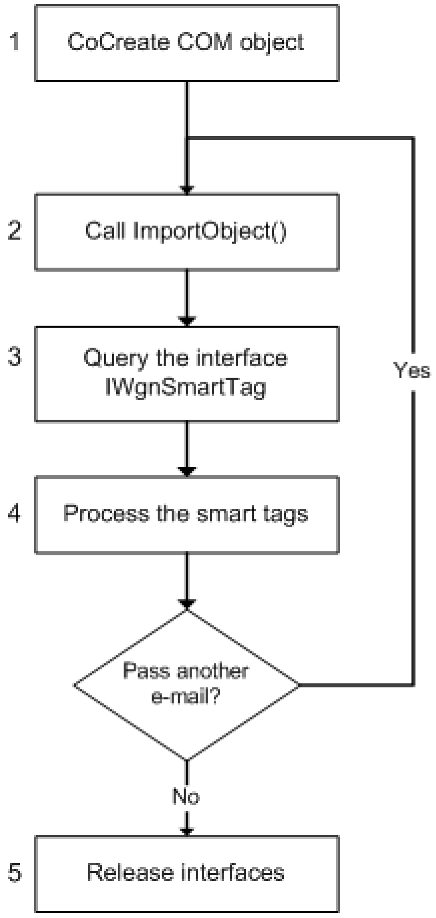
Scenario A: Simple processing: Synchronous, with no deferred completion

This scenario describes the simplest integration with the External Agent. It is included here for illustration purposes only. In practice, customers will almost certainly implement a more complex integration.

In this scenario, the caller (a third party application) passes an e-mail synchronously to the External Agent, which then forwards the e-mail directly to a local policy engine for processing. Policy is configured to return smart tags. Deferred completion is not enabled.

After the message has been processed by the policy engine, the External Agent returns any associated smart tags to the caller. In this simple scenario, the caller uses these smart tags to categorize the e-mails but does not store them in an archive.

The External Agent API processing sequence is summarized below:



Scenario A: Simple processing

More information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)

Scenario A: Processing steps

1. **CoCreate the COM object:** CoCreate WgnActiveImportConnector and then obtain the IWgnImportConnectorConfig interface.
2. **IWgnImportConnectorConfig interface:** In this scenario, the External Agent passes e-mails directly to a policy engine. This is appropriate when calling WgnActiveImportConnector in synchronous mode. This is also the default configuration for WgnActiveImportConnector and so there is no need to use the IWgnImportConnectorConfig interface.
3. **IWgnActiveImportConnector interface:** The calling application calls the ImportObject() method to synchronously pass an e-mail to the External Agent. This method internally passes the e-mail to a policy engine for processing and returns the results of that processing.

   The External Agent returns any associated smart tags to the caller. This scenario assumes that the caller requires these smart tags to categorize the e-mail but does **not** intend to send the e-mail to an archive. Deferred completion is **not** enabled.The parameter values needed to implement this scenario are therefore:

Parameters	Value	Notes
messageId	NULL	The caller does not archive e-mails after policy processing.
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	The caller does not archive e-mails after policy processing.
xmlMessageAttributes	NULL	The caller does not pass additional event attribute data to the External Agent.
allowDeferredCompletion	FALSE	In scenario A, events processed by the policy engine can be committed to the CMS database without waiting for the caller to process the results of policy analysis.

Note: For the full set of parameter values that must be passed to ImportObject() when processing a MAPI message, see Scenario A: ImportObject() Parameters (see page 22).

4. **IWgnSmartTag interface:** If the call to ImportObject() is successful, the caller can call GetInterface() on the resultant IWgnImportConnectorResult interface to obtain the IWgnSmartTag interface.

5. **Process the smart tags:** Any smart tags assigned to the e-mail by the policy engine can be processed by calling the GetSmartTagXML() method, which retrieves the message smart tags encoded in XML, or enumerated by calling methods such as GetCount() and GetSmartTag().

6. **Release the interfaces:** After each message has been processed, the caller must release the 'message processing' interfaces. In this scenario, these are the IWgnImportConnectorResult and IWgnSmartTag interfaces.

   The WgnActiveImportConnector boundary interfaces must be released at the end of the session. These are the IWgnImportConectorConfig (if queried) and IWgnActiveImportConnector interfaces.

More information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)

Scenario A: ImportObject() Parameters

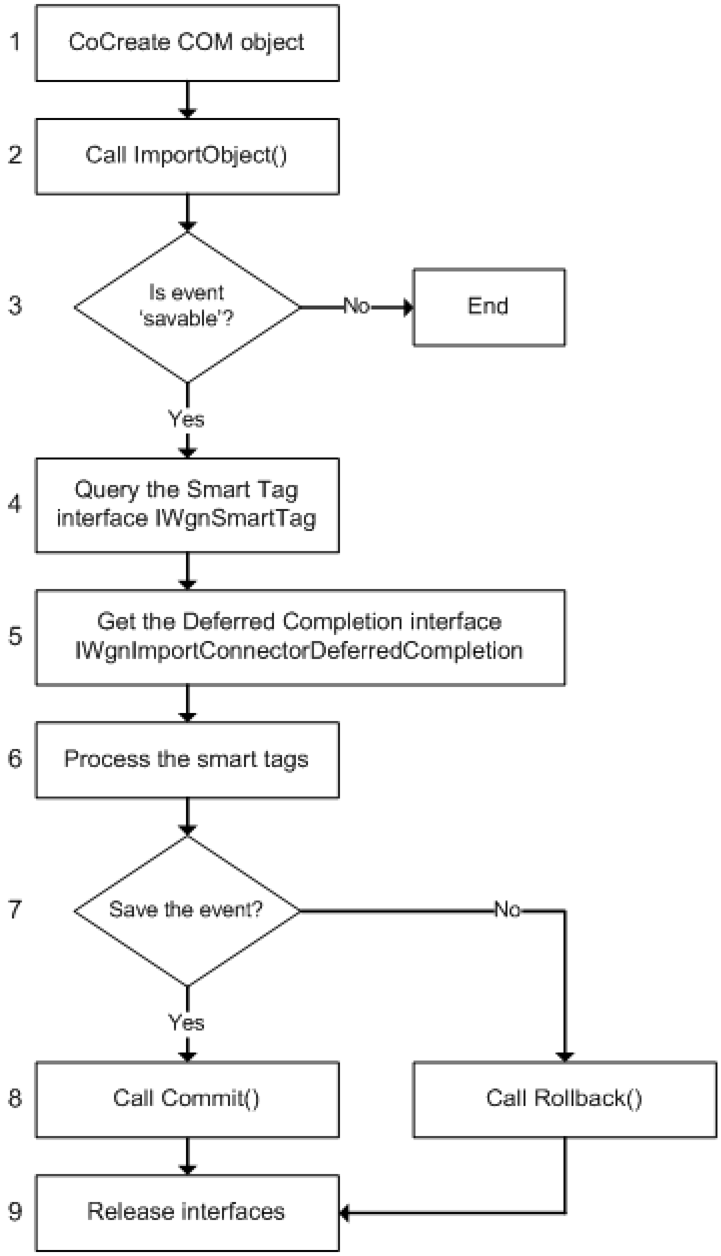
Parameters	Value	Notes
objectType	WGN_IMPORTOBJ_TYPE_EMAIL	Specifies an e-mail. See the WGN_IMPORTOBJ_TYPE typedef (see page 59).
objectFormat	WGN_IMPORTOBJ_FORMAT_MAPI	Specifies a MAPI e-mail. See the WGN_IMPORTOBJ_FORMAT typedef (see page 59).
objectTransport	WGN_IMPORTOBJ_TRANSPORT_COM	Specifies a COM transport. See the WGN_IMPORTOBJ_TRANSPORT typedef (see page 60).
objectData	Actual e-mail	Variant type: VT_UNKNOWN IUnknown* interface from IMessage* interface
For details about other Type-Format-Transport permutations, see Input Combinations for ImportObject() Method (see page 63).		

Parameters	Value	Notes
messageId	NULL	See Scenario A: Processing Steps (see page 21).
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	See Scenario A: Processing Steps (see page 21).
xmlMessageAttributes	NULL	See Scenario A: Processing Steps (see page 21).
eventSizeHintBytes	Zero	Not relevant. This parameter is used to calibrate hub throttling, but in scenario A there is no hub; the External Agent passes e-mails directly to a local policy engine.
allowDeferredCompletion	FALSE	See Scenario A: Processing Steps (see page 21).

Scenario B: Simple processing: Deferred completion

This scenario describes a more complex integration with the External Agent.

In this scenario, the caller (a third party application) still passes an email synchronously to the External Agent, but this time deferred completion is enabled. This provides the caller with greater control over which e-mails are stored on the CMS, allowing it to assess the results of any policy processing and to filter out emails that do not need to be retained.



Scenario B: Deferred completion

More information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)

Scenario B: Processing steps

1. **CoCreate the COM object:** CoCreate WgnActiveImportConnector and then obtain the IWgnImportConnectorConfig interface. As with scenario A, in this scenario there is no need to use the IWgnImportConnectorConfig interface.
2. **IWgnActiveImportConnector interface:** The calling application calls the ImportObject() method to synchronously pass an e-mail to the External Agent. This method internally passes the e-mail to a policy engine for processing and returns the results of that processing (that is, any applicable smart tags) to the caller.

   As with scenario A, this scenario assumes that the caller does not intend to send the e-mail to an archive. But unlike scenario A, in this scenario deferred completion **is** enabled.

   The parameters values needed to implement this scenario are therefore:

Parameters	Value	Notes
messageId	NULL	The caller does not archive e-mails after policy processing.
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	The caller does not archive e-mails after policy processing.
xmlMessageAttributes	NULL	The caller does not pass additional event attribute data to the External Agent.
allowDeferredCompletion	TRUE	In scenario B, events processed by the policy engine cannot be committed to the CMS database until the caller has processed the results of policy analysis.

Note: For the full set of parameter values that must be passed to ImportObject() when processing a MAPI message, see Scenario B: ImportObject() Parameters (see page 27).

3. **Is event savable?** When the results of policy processing are returned to the caller, the next processing step depends on whether the event is savable. That is, did the event activate a policy trigger that would normally cause it to be saved on the CMS? If the event:

   - **Is not savable**, the External Agent does not process the e-mail any further. The caller may, however, use any smart tags that were returned to categorize the e-mail.
   - **Is savable**, the caller queries the 'result' interface—see the next step.

4. **IWgnSmartTag interface:** If the call to ImportObject() is successful, the caller can call GetInterface() on the resultant IWgnImportConnectorResult interface to obtain the IWgnSmartTag interface.

5. **IWgnImportConnectorDeferredCompletion interface:** If the call to ImportObject() is successful, the caller also calls the GetDeferredCompletion() method in the IWgnImportConnectorResult interface to return the IWgnImportConnectorDeferredCompletion interface.

6. **Process the smart tags:** Any smart tags assigned to the e-mail by the policy engine can be processed by calling the GetSmartTagXML() method, which retrieves the message smart tags encoded in XML, or enumerated by calling methods such as GetCount() and GetSmartTag().

7. **Save the event?** The caller assesses the smart tags returned by the External Agent.

   If the e-mail is in a 'must retain' category, the caller calls Commit() method in the **IWgnImportConnectorDeferredCompletion** interface to instruct the External Agent to save the e-mail to the CMS database.

8. If the e-mail does not need to be retained, the caller calls Rollback() method to instruct the External Agent that the e-mail must not be saved in the CMS database.

9. **Release the interfaces:** After each message has been processed, the caller must release the 'message processing' interfaces. In this scenario, these are the IWgnImportConnectorResult and IWgnSmartTag interfaces.

   The WgnActiveImportConnector boundary interfaces must be released at the end of the session. These are the IWgnImportConectorConfig (if queried) and IWgnActiveImportConnector interfaces.

Scenario B: ImportObject() Parameters

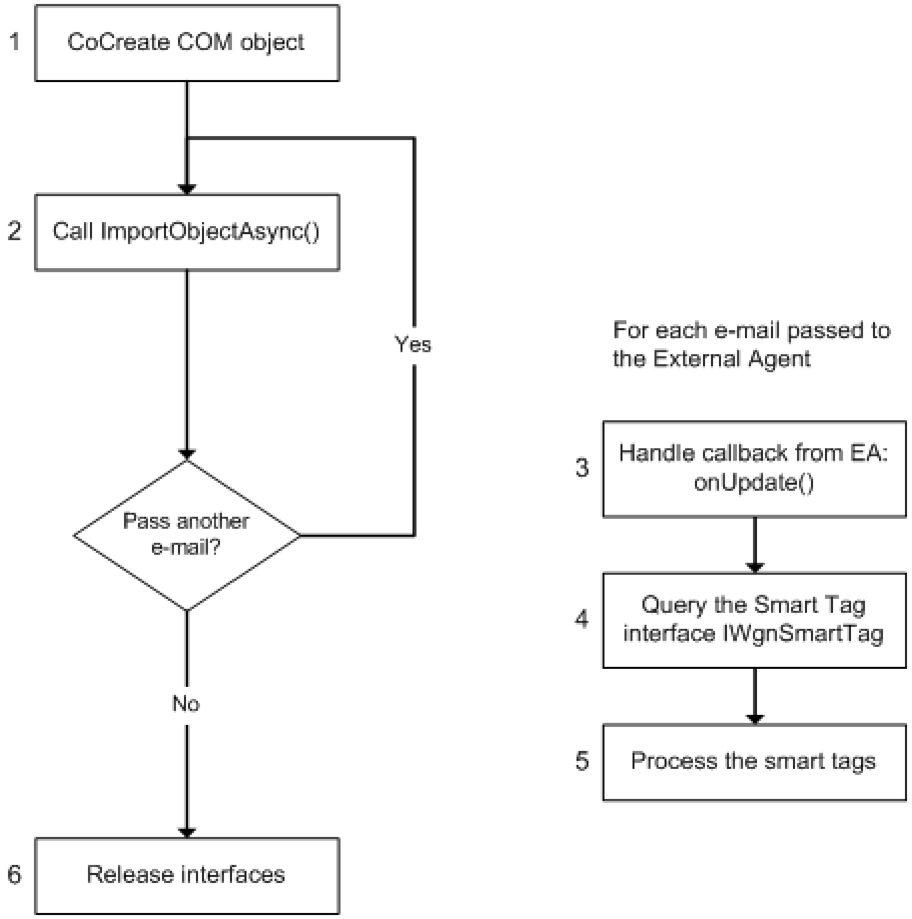
Parameters	Value	Notes
objectType	WGN_IMPORTOBJ_TYPE_EMAIL	Specifies an e-mail. See the WGN_IMPORTOBJ_TYPE typedef (see page 59).
objectFormat	WGN_IMPORTOBJ_FORMAT_MAPI	Specifies a MAPI e-mail. See the WGN_IMPORTOBJ_FORMAT typedef (see page 59).
objectTransport	WGN_IMPORTOBJ_TRANSPORT_COM	Specifies a COM transport. See the WGN_IMPORTOBJ_TRANSPORT typedef (see page 60).
objectData	Actual e-mail	Variant type: VT_UNKNOWN IUnknown* interface from IMessage* interface
For details about other Type-Format-Transport permutations, see Input Combinations for ImportObject() Method (see page 63).		
messageId	NULL	See Scenario B: Processing Steps (see page 25).
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	See Scenario B: Processing Steps (see page 25).
xmlMessageAttributes	NULL	See Scenario B: Processing Steps (see page 25).
eventSizeHintBytes	Zero	Not relevant. This parameter is used to calibrate hub throttling, but in scenario A there is no hub; the External Agent passes e-mails directly to a local policy engine.
allowDeferredCompletion	TRUE	See Scenario B: Processing Steps (see page 25).

Scenario C: Asynchronous processing

This scenario describes a more complex integration with the External Agent, where e-mails are passed asynchronously to the External Agent.

In this scenario, the caller (a third party application) passes e-mails asynchronously to the External Agent. This allows the caller to maximize throughput, making optimum use of available system resources. Each time the caller passes an e-mail to the External Agent, it also passes a callback interface through which the External Agent will inform the caller of the results of processing that e-mail.

Policy is configured to return smart tags and deferred completion is not enabled. After each e-mail has been processed by the policy engine, the External Agent returns any associated smart tags in a callback to the caller. The caller then uses these smart tags to categorize the e-mails.


1
CoCreate COM object
2
Call ImportObjectAsync()
Yes
Pass another e-mail?
No
6
Release interfaces
For each e-mail passed to the External Agent
3
Handle callback from EA: onUpdate()
4
Query the Smart Tag interface IWgnSmartTag
5
Process the smart tags

Scenario C: Asynchronous processing

Steps 3 through 5 are repeated once for each e-mail passed by the caller to the External Agent. At any time, there may be multiple threads each performing steps 3 through 5.

More Information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)

Scenario C: Processing steps

1. **CoCreate the COM object:** CoCreate WgnActiveImportConnector and then obtain the IWgnImportConnectorConfig interface. As with scenario A, in this scenario there is no need to use the IWgnImportConnectorConfig interface.
2. **IWgnActiveImportConnector interface:** The calling application calls the ImportObjectAsync() method to asynchronously pass an e-mail to the External Agent. This method also takes an IWgnImportConnectorCallback interface pointer to an object hosted by the caller so the External Agent can return the results of processing (that is, any applicable smart tags) to the caller when processing is complete.

   As with scenario A, this scenario assumes that the caller does not intend to send the e-mail to an archive and that deferred completion is **not** enabled.

   The parameters values needed to implement this scenario are:

Parameters	Value	Notes
messageId	NULL	The caller does not archive e-mails after policy processing.
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	The caller does not archive e-mails after policy processing.
xmlMessageAttributes	NULL	The caller does not pass additional event attribute data to the External Agent.
allowDeferredCompletion	FALSE	In scenario C, events processed by the policy engine can be committed to the CMS database without waiting for the client to process the results of policy analysis.
contextTag	Any value	An e-mail identifier. The External Agent returns the same value to the caller in the subsequent callback (see step 3), enabling the caller to match the callback to the original e-mail.
asyncCallback	Interface pointer	A pointer to an interface implemented by a caller-defined class, through which WgnActiveImportConnector will inform the caller of the results from processing the message.

Note: For the full set of parameter values that must be passed to ImportObjectAsync() when processing a MAPI message, see Input Combinations for ImportObject() Method (see page 63).

3. **IWgnImportConnectorCallback interface:** When the External Agent has finished processing each e-mail, it notifies the caller and returns the results of any policy processing.
4. To do this, the External Agent calls the onUpdate() method on the IWgnImportConnectorCallback interface to the COM object created by the caller. This call includes the contextTag identifier passed in step 2, enabling the caller to match the callback to an e-mail, plus the IWgnImportConnectorResult interface.
5. **IWgnSmartTag interface:** If the call to ImportObjectAsync() is successful, the caller can call GetInterface() on the IWgnImportConnectorResult interface returned in step 3 to obtain the IWgnSmartTag interface. This enables the caller to retrieve the results of analyzing the e-mail plus any smart tags.
6. **Process the smart tags:** The caller can process any smart tags assigned to the e-mail by calling methods in the IWgnSmartTag interface. The GetSmartTagXML() method retrieves the message smart tags encoded in XML. Or the caller can enumerate the smart tags by calling methods such as GetCount() and GetSmartTag().

   Note: Steps 3-4-5 are performed once for each e-mail. That is, this sequence (3-4-5) is repeated for each e-mail passed to the External Agent. At any time, there may be multiple threads each performing steps 3-4-5.
7. **Release the interfaces:** After each message has been processed, the caller must release the 'message processing' interfaces. In this scenario, these are the IWgnImportConnectorResult and IWgnSmartTag interfaces.

   The WgnActiveImportConnector boundary interfaces must be released at the end of the session. These are the IWgnImportConectorConfig (if queried) and IWgnActiveImportConnector interfaces.

Scenario C: ImportObjectAsync() Parameters

Parameters	Value	Notes
objectType	WGN_IMPORTOBJ_TYPE_EMAIL	Specifies an e-mail. See the WGN_IMPORTOBJ_TYPE typedef (see page 59).
objectFormat	WGN_IMPORTOBJ_FORMAT_MAPI	Specifies a MAPI e-mail. See the WGN_IMPORTOBJ_FORMAT typedef (see page 59).
objectTransport	WGN_IMPORTOBJ_TRANSPORT_COM	Specifies a COM transport. See the WGN_IMPORTOBJ_TRANSPORT typedef (see page 60).

Parameters	Value	Notes
objectData	Actual e-mail	Variant type: VT_UNKNOWN IUnknown* interface from IMessage* interface
For details about other Type-Format-Transport permutations, seeInput Combinations for ImportObject() Method (see page 63) .		
messageId	NULL	See Scenario C: Processing Steps (see page 29).
importSource	WGN_ACTIVEIMPORTSOURCE_NONE	See Scenario C: Processing Steps (see page 29).
xmlMessageAttributes	NULL	See Scenario C: Processing Steps (see page 29).
eventSizeHintBytes	Zero	Not relevant. This parameter is used to calibrate hub throttling, but in scenario A there is no hub; the External Agent passes e-mails directly to a local policy engine.
allowDeferredCompletion	FALSE	See Scenario C: Processing Steps (see page 29).
contextTag	Any value defined by the caller	See Scenario C: Processing Steps (see page 29).

Chapter 5: API Interfaces: Applying Policy and Generating CA DataMinder Events

This section contains the following topics:

COM Object: WGNRDILib::WgnActiveImportConnector

CoCreate this COM object, then query the object for the interfaces described in this section and use the methods listed for each interface.

Interface IWgnImportConnectorConfig

Use the methods on this interface to change the default behavior for WgnActiveImportConnector. If you use this configuration interface, do so **before** supplying messages. After ImportObject() or ImportObjectAsync() have been called, the configuration is fixed. To change the configuration after this time, you will need to CoCreate a new WgnActiveImportConnector object.

Method SetOutputDestination

WgnActiveImportConnector can output messages to a local hub or directly to a local policy engine. It defaults to a policy engine (this setup is the easiest to configure). That is, the OutputDestination property defaults to WGN_ACTIVEIMPORT_DESTINATION_PE.

This method sets the OutputDestination property.

```
HRESULT SetOutputDestination (
    [in]    WGN_ACTIVEIMPORT_DESTINATION    newVal);
```

More information:

Typedef WGN_ACTIVEIMPORT_DESTINATION (see page 60)

Method GetOutputDestination

This method retrieves the OutputDestination property:

```
HRESULT GetOutputDestination (
    [out]    WGN_ACTIVEIMPORT_DESTINATION    *pVal);
```

Method SetHubThrottleMode

You can set the hub throttling mode to Wait or Fail. WgnActiveImportConnector defaults to WGN_ACTIVEIMPORT_THROTTLE_WAIT_EVENT.

This method sets the HubThrottleMode property.

```
HRESULT SetHubThrottleMode (
    [in]    WGN_ACTIVEIMPORT_HUB_THROTTLE_MODE    newVal);
```

More information:

Typedef WGN_ACTIVEIMPORT_DESTINATION (see page 60)

Method GetHubThrottleMode

This method retrieves the HubThrottleMode property.

```
HRESULT GetHubThrottleMode (
    [out]    WGN_ACTIVEIMPORT_HUB_THROTTLE_MODE    *pVal);
```

Interface IWgnActiveImportConnector

Use this interface to pass messages to the External Agent and to receive results, including smart tag details, of any policy applied to that message.

Method ImportObject

This is a **synchronous** call to supply a message to CA DataMinder and to return Smart Tags, as a result of analysis against policy. This may lead to an event corresponding to the message being saved in the CA DataMinder database, depending on how policy is configured.

Note: Asynchronous calls are covered by the ImportObjectAsync() method (see page 38).

```
HRESULT ImportObject(
    [in]         WGN_IMPORTOBJ_TYPE      objectType,
    [in]         WGN_IMPORTOBJ_FORMAT    objectFormat,
    [in]         WGN_IMPORTOBJ_TRANSPORT objectTransport,
    [in]         VARIANT                 objectData,
    [in, unique] LPOLESTR                messageId,  // OPTIONAL
    [in]         DWORD                   importSource,
    [in, unique] LPOLESTR                xmlMessageAttributes,  // OPTIONAL
    [in]         DWORD                   eventSizeHintBytes,
    [in]         BOOL                    allowDeferredCompletion,
    [out]        IWgnImportConnectorResult  **results
    );
```

Where:

objectType

Specifies the type of object to import (e-mail or IM). For details, see the typedef WGN_IMPORTOBJ_TYPE (see page 59).

objectFormat

Specifies the format of the object to import. For details, see the typedef WGN_IMPORTOBJ_FORMAT (see page 59).

objectTransport

Specifies the method used to transfer data to the External Agent. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60).

objectData

Specifies the source data being imported.

messageID

Is a string that specifies the unique identifier of the message within the archive.

This parameter is optional. It can be NULL if no link is to be made to a remote archive or if allowDeferredCompletion is TRUE, in which case the client will supply the messageID when committing the transaction.

This parameter must be set to NULL if and only if the importSource parameter is set to WGN_ACTIVEIMPORTSOURCE_NONE.

importSource

Is a fixed integer that identifies the source archive. If there is no link to a remote archive, it is WGN_ACTIVEIMPORTSOURCE_NONE.

xmlMessageAttributes

Optional XML containing additional metadata for the message.

allowDeferredCompletion

If set to **TRUE**, this indicates that an event must not be committed to the CA DataMinder database until the client has processed the results of the analysis. This enables the client to:

- Use Smart Tags to determine whether or not the message is to be archived. In turn, this determines whether CA DataMinder must save a corresponding event in the CMS database.
- Postpone allocating a unique ID for the message. This is useful in situation a. above, where the unique message ID is not known until the message is actually archived.
- Postpone identifying the importSource archive identifier. This allows the caller to choose which archive system to send the message to (where multiple systems are used), based on the content of the Smart Tags.

When using deferred completion, the caller is able to do all of the above via the IWgnImportConnectorDeferredCompletion interface. This is:

- **Either** obtained from the IWgnImportConnectorResult interface.
- **Or** supplied to the client via the IWgnImportConnectorCallback interface.

The IWgnImportConnectorDeferredCompletion interface will only be supplied if CA DataMinder is intending to save an event in its database. Settings in policy dictate the conditions for this.

If set to **FALSE**, the messageId and importSource parameters must be set to their final values, because these will be stored in the CA DataMinder database if the analysis against policy dictates it.

Return Values for ImportObject Method

WGN_S_ACTIVEIMPORT_AWAITING_COMPLETION

The caller has set allowDeferredCompletion parameter to TRUE. CA DataMinder has analyzed the message and is intending to save an event for this message is its database. It is now waiting for the caller to commit this transaction or roll it back via the IWgnImportConnectorDeferredCompletion interface.

In this situation, the caller is able to retrieve the Smart Tags generated by the analysis of the message prior to committing or rolling back the transaction.

Success code

A return value that passes the SUCCEEDED() macro, indicating that the message has been successfully analyzed. The results of this analysis, including Smart Tags, can be obtained via the returned results interface, which must be released by the caller. The IWgnSmartTag interface can be obtained by calling GetInterface() on the results interface. The GetProcessingResult() method of the results object will return the same value as this return code.

Error code

Any return value that fails the SUCCEEDED() macro can be returned, indicating that the message has not been fully processed. If the reason for failure is due to network or other catastrophic failure, it is possible that an event was saved to the CA DataMinder database, but the results will not be accessible.

If an error code is returned from this method, the results output parameter value will be NULL.

The GetStatusString() method can be called to convert the returned error code to a string.

More information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)
Values for importSource (see page 70)
Specification for XmlMessageAttribute Data (see page 68)

Method ImportObjectAsync

This method supplies a message to CA DataMinder to be analyzed against policy **asynchronously**.

Note: Synchronous calls are covered by the ImportObject() method.

```
HRESULT ImportObjectAsync(
    [in]         WGN_IMPORTOBJ_TYPE        objectType,
    [in]         WGN_IMPORTOBJ_FORMAT      objectFormat,
    [in]         WGN_IMPORTOBJ_TRANSPORT   objectTransport,
    [in]         VARIANT                   objectData,
    [in, unique] LPOLESTR                  messageId,  // OPTIONAL
    [in]         DWORD                     importSource,
    [in, unique] LPOLESTR                  xmlMessageAttributes,  // OPTIONAL
    [in]         DWORD                     eventSizeHintBytes,
    [in]         BOOL                      allowDeferredCompletion,
    [in, unique] LPvoid                    contextTag,
    [in]         IWgnImportConnectorCallback  *asyncCallback
    );
```

Where:

objectType

Specifies the type of object to import (e-mail or IM). For details, see the typedef WGN_IMPORTOBJ_TYPE (see page 59).

objectFormat

Specifies the format of the object to import. For details, see the typedef WGN_IMPORTOBJ_FORMAT (see page 59).

objectTransport

Specifies the method used to transfer data to the External Agent. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60).

objectData

Specifies the source data being imported.

messageID

Is a string that specifies the unique identifier of the message within the archive.

This parameter is optional. It can be NULL if no link is to be made to a remote archive or if allowDeferredCompletion is TRUE, in which case the client will supply the messageID when committing the transaction.

This parameter must be set to NULL if and only if the importSource parameter is set to WGN_ACTIVEIMPORTSOURCE_NONE.

importSource

Is a fixed integer that identifies the source archive. If there is no link to a remote archive, it is WGN_ACTIVEIMPORTSOURCE_NONE.

xmlMessageAttributes

Optional XML containing additional metadata for the message.

eventSizeHintBytes

Specifies an average message size (in bytes). The hub uses this value to estimate the total size of the various hub event queues. (Hub throttling is based either on the total size or the total event count for all hub event queues).

allowDeferredCompletion

See ImportObject()method.

contextTag

This parameter allows the caller to supply contextual data to this asynchronous operation. The WgnActiveImportConnector will transparently pass this through to the callback via the IWgnImportConnectorCallback interface. If deferred completion is used, this value is not automatically passed through to the callback for a CommitAsync() operation. To achieve this, the caller must supply the value to the CommitAsync() method.

asyncCallback

An interface to an object hosted by the caller through which WgnActiveImportConnector will inform the caller of the results from processing the message.

More information:

Synchronous Versus Asynchronous Operations (see page 11)
Single-pass Processing Versus Deferred Completion (see page 12)
Values for importSource (see page 70)
Specification for XmlMessageAttribute Data (see page 68)

Method GetStatusString

Use this method to obtain a description of the supplied HRESULT code.

```
HRESULT GetStatusString(
    [in]          long        hResult,
    [out,retval]  LPOLESTR    *statusString
    };
Where:
```

hResult

The HRESULT code value for which a description is required.

statusString

A pointer to an LPOLESTR string that will retrieve the description of the HRESULT code. The caller is responsible for freeing this string following a successful call by using CoTaskMemFree().

Interface IWgnImportConnectorResult

Use this interface to obtain the result of an import.

Method GetProcessingResult

Use this method to get the HRESULT processing result code.

The ProcessingResult property will be set to WGN_S_ACTIVEIMPORT_SAVE_DEFERRED if the processing has been successful **and** the client is now required to commit or rollback the event. This will only be the case if the allowDeferredCompletion parameter to ImportObject() or ImportObjectAsync() was set to TRUE.

```
HRESULT GetProcessingResult(
    [out]    long    *pVal);
```

Method GetDeferredCompletion

The method obtains an IWgnImportConnectorDeferredCompletion interface that allows the client to commit or rollback an event.

```
HRESULT GetDeferredCompletion(
    [out]    IWgnImportConnectorDeferredCompletion  **deferredCompletion);
```

Method GetResultXML

Use this method to retrieve the results policy analysis, including any smart tags, encoded in XML. For example, you may prefer to use this method instead of using the IWgnSmartTag interface to get the smart tags.

```
HRESULT GetResultXML(
    [out]    LPOLESTR    *pXMLVal);
```

Method GetInterface

Use this method to obtain an IWgnSmartTag interface.

```
HRESULT GetInterface(
    [in] REFIID riid,
    [out, iid_is(riid)] void **ppvInterface
);
```

Where:

riid

Is the Unique ID of the interface being requested (for example __uuid(IWgnSmartTag).

ppvInterface

Is a pointer to a buffer that receives a pointer to the requested interface.

Interface IWgnImportConnectorCallback

If passing messages to the External Agent **asynchronously**, the caller must create its own COM object implementing this interface, and pass this interface to the ImportObjectAsync() method. The External Agent will call back on this interface with the results of policy processing.

Method OnUpdate

The External Agent calls this method to notify the caller that it has finished processing an e-mail.

```
HRESULT OnUpdate(
    [in,unique]  LPVOID                     contextTag,
    [in]         long                       hrStatus,
    [in,unique]  IWgnImportConnectorResult  *results,
    [in,unique]  IWgnImportConnectorDeferredCompletion *deferredCompletion
  );
```

Where:

contextTag

Is the value supplied to either the ImportObjectAsync() or CommitAsync() methods. It allows the client to match the callback to the message passed to the External Agent.

hrStatus

Can be:

S_OK

The message was processed successfully and there is no pending CA DataMinder event. In this situation, the deferredCompletion parameter should be NULL.

WGN_S_ACTIVEIMPORT_AWAITING_COMPLETION

The message has been successfully analyzed, but no data has been saved. The client must now ROLLBACK or COMMIT the transaction by obtaining the IWgnImportConnectorDeferredCompletion interface. This enables the client to set additional data on the captured event depending on the values of smart tags. In this situation, the deferredCompletion parameter should not be NULL.

Error code

Any return value that fails the SUCCEEDED() macro, indicating that the message failed to be fully processed. In this situation, the deferredCompletion parameter should be NULL.

results

An interface allowing onUpdate() to retrieve the results of analyzing the message along with any Smart Tags. This interface can be used to obtain the IWgnSmartTag interface by calling GetInterface().

deferredCompletion

If the client supplied TRUE to the allowDeferredCompletion parameter of the ImportObjectAsync() method **and** CA DataMinder has an event pending being saved in its database, this interface will be populated and the implementer of the callback object must either Commit() or Rollback() the transaction. This need not be performed inline. It is permissable to AddRef() the deferredCompletion interface and invoke it after this method has returned.

Important! CA DataMinder cannot progress the event until this occurs, which can lead to degraded performance if events are not committed or rolled back promptly.

When this method is invoked after a Commit() operation, the deferredCompletion parameter will be NULL, indicating that no further action is necessary.

Return values for onUpdate method

The implementer of the callback object should return S_OK from this method.

Interface IWgnImportConnectorDeferredCompletion

If the allowDeferredCompletion parameter is set to TRUE, the caller must use this interface to indicate to the EA whether or not the event should be committed or rolled back.

Method SetRemoteDataLocation

Use this method to override the remote data location that was set in a call to IWgnActiveImportConnector::ImportObject() or ImportObjectAsync().

```
HRESULT Commit(
    [in]    LPOLESTR    messageID,
    [in]    DWORD       importSource
    );
```

Where:

messageID

Is the unique identifier of the message within the archive. This parameter may be NULL if no link is to be made to a remote archive.

This parameter must be set to NULL if and only if the importSource parameter is set to WGN_ACTIVEIMPORTSOURCE_NONE.

importSource

Is the archive identifier, or WGN_ACTIVEIMPORTSOURCE_NONE if there is no link to a remote archive.

More information:

Values for importSource (see page 70)

Method SetMessageAttributes

Use this method to set additional attributes on an event before commiting it.

```
HRESULT SetMessageAttributes(
    [in, unique] LPOLESTR    xmlMessageAttributes
```

Where:

xmlMessageAttributes

Optional XML containing additional metadata for the message.

Method Commit

This method commits a message that was analyzed using deferred completion.

Important! Commit() must only be called if the object was imported synchronously (by using ImportObject()).

```
HRESULT Commit();
```

Return values for Commit method

S_OK

The synchronous commit operation completed successfully.

Error code

Any return value that fails the SUCCEEDED() macro, indicating that the synchronous commit operation failed.

Method CommitAsync

This method asynchronously commits a message that was analyzed using deferred completion.

Important! CommitAsync() must only be called if the object was imported asynchronously (by using ImportObjectAsync()).

```
HRESULT Commit(
    [in, unique]    LPVOID                       contextTag,
    [in]            IWgnImportConnectorCallback  *completionCallback
    );
```

Where:

contextTag

This parameter allows the client to supply contextual data to this asynchronous operation. The WgnActiveImportConnector will transparently pass this through to the callback via the IWgnImportConnectorCallback interface.

completionCallback

The interface on which the client will be notified of the result of the commit operation.

Return values for CommitAsync method

WGN_S_ACTIVEIMPORT_PENDING

The message is being committed asynchronously. The completionCallback interface will be invoked to inform the client when the operation completes. This may be invoked before this method returns, due to the concurrent nature.

Error code

Any return value that fails the SUCCEEDED() macro, indicating that the synchronous commit operation failed, or that the asynchronous commit failed to be instigated. The completionCallback interface will not be invoked in this situation.

Method Rollback

This method prevents an event that is currently pending from being saved in the CA DataMinder database.

```
HRESULT Rollback();
```

Interface IWgnSmartTag

The IWgnSmartTag interface is used to examine the Smart Tags of an imported Event.

You can obtain this interface by calling the GetInterface() method on the IWgnImportConnectorResult interface.

Method GetCount

Use this method to obtain the number of smart tags.

```
HRESULT GetCount(
    [out]    DWORD    *count);
```

Where:

count

Is the number of smart tags.

Method GetSmartTag

Use this method to return a smart tag from the set of tags generated as a result of analyzing a message.

```
HRESULT GetSmartTag(
    [in]     DWORD          index,
    [out]    WgnSmartTag    *smartTag
    );
```

Where:

index

Is the index of the Smart Tag to obtain, from 0 to count-1 (count is obtained from the GetCount method).

smartTag

Is a client-supplied structure which the EA populates with the name and value of the requested Smart Tag. The members of the structure must be set to NULL before calling this method.

Important! It is the client's responsibility to free the fields of the WgnSmartTag structure via CoTaskMemFree().

Method GetSmartTag XML

Use this method to retrieve the smart tags encoded in XML. The resulting string must be freed by the caller by calling CoTaskMemFree()

```
HRESULT GetSmartTagXML(
    [out]    LPOLESTR     *pXMLVal);
```

Important! For an XML example, see SmartTag XML Specification (see page 70).

Interface IWgnImportFileObject

The IWgnImportFileObject can be used to prepare a file object for importing. It is particularly useful if additional streams need to be added to the object.

Method InitialiseFile

Use this method to initialise the file from your data source

```
HRESULT InitialiseFile(
    [out]    WGN_IMPORTOBJ_TRANSPORT    objectTransport,
    [in]     VARIANT                    objectData
);
```

Where:

objectTransport

Specifies the method used to transfer data to the API. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60).

objectData

Specifies the source data being imported.

Method AddAdditionalFileStream

Use this method to add additional streams to the file object.

```
HRESULT AddAdditionalFileStream(
    [in]    WGN_IMPORTOBJ_TRANSPORT  streamTransport,
    [in]    VARIANT                  streamData,
    [in]    LPCOLESTR                streamName
    );
```

Where:

streamTransport

Specifies the method used to transfer data to the API. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60). Currently the only supported transport for adding additional file streams is WGN_IMPORTOBJ_TRANSPORT_STREAM.

streamData

Specifies the source data being imported.

streamName

Specifies the name of the stream.

Chapter 6: API Interfaces: Generating EVF Files

This section contains the following topics:

COM object: WGNRDILib::WgnImportConnector

CoCreate this COM object, then query the object for the interfaces described in this section and use the methods listed for each interface.

Interface IWgnImportConnector

Use this interface to import messages from a specified source.

Note: This interface has now been superseded.

More information:

Interface Versions (see page 13)

COM Object WGNRDILib::WgnImportConnector

Co-create this COM object, then query the object for the interface and use the methods listed below.

Method ImportMAPIMessage

Use this method to import a message.

```
HRESULT ImportMAPIMessage(
    [in] IUnknown    *message,
    [in] BSTR        messageID,
    [in] DWORD       importSource
);
```

Where:

message

Is the interface pointer to a MAPI IMessage object

messageID

Is a string that uniquely identifies the message in the archive. This will be the string given to CA's Remote Data Manager (RDM) to enable it to retrieve the message contents. But see the warning below.

Important! The messageID parameter is critical! You can use any string as the identifier but it must be **unique** and the identifiers must be managed correctly to ensure that CA DataMinder can retrieve the full details for imported e-mails. Your application must be able to use this string, when called by the RDM, to access the exact same message.

importSource

Is the archive identifier.

Return values for ImportMAPIMessage method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

0xE5630D2E

Indicates an unknown import source. This value is returned if the importSource parameter is set to an invalid number.

0xE5630D18

"A mail event could not be created because the available disk space was less than the configured threshold."

This error message may indicate that the cache is not being cleared quickly enough. If you get this error, wait before retrying to import the e-mail. If you continue to get this error, then you must halt the archive process and flag an alert.

More information:

Values for importSource (see page 70)

Method GetStatusString

Following a successful import operation, the return code is: S_OK. If an error occurs, use this method to find the string value (that is, an explanation) of the error.

```
HRESULT GetStatusString(
    [in] long            hResult,
    [out,retval] BSTR*    statusString
};
```

Return values for GetStatusString method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

More information:

Error Codes (see page 75)

Interface IWgnImportConnector2

Use this interface to import messages from a specified source.

Note: This interface has now been superseded.

More information:

Interface Versions (see page 13)
Error Codes (see page 75)

COM Object WGNRDILib::WgnImportConnector

Co-create this COM object, then query the object for the interface and use the methods listed below.

Method ImportObject

Use this method to import a message.

```
HRESULT ImportObject(
    [in] WGN_IMPORTOBJ_TYPE       objectType,
    [in] WGN_IMPORTOBJ_FORMAT     objectFormat,
    [in] WGN_IMPORTOBJ_TRANSPORT  objectTransport,
    [in] VARIANT                  objectData,
    [in] BSTR                     messageID,
    [in] DWORD                    importSource,
    [in, out,unique] BSTR *       newMessageID,    /* OUT-ONLY */
    [in, out,unique] DWORD*       newImportSource  /* OUT-ONLY */
);
```

Where:

objectType

Specifies the type of object to import (e-mail or IM). For details, see the typedef WGN_IMPORTOBJ_TYPE (see page 59).

objectFormat

Specifies the format of the object to import. For details, see the typedef WGN_IMPORTOBJ_FORMAT (see page 59).

objectTransport

Specifies the method used to transfer data to the RDI. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60).

objectData

Specifies the source data being imported.

messageID

Is a string that uniquely identifies the message in the archive. This will be the string given to CA's Remote Data Manager (RDM) to enable it to retrieve the message contents. But see the warning below.

Important! messageID is critical! You can use any string as the identifier but it must be **unique** and the identifiers must be managed correctly to ensure that CA DataMinder can retrieve the full details for imported e-mails. Your application must be able to use this string, when called by the RDM, to access the exact same message.

importSource

Is the archive identifier.

newMessageID

Is not used in the current version of the RDI.

newImportSource

Is not used in the current version of the RDI.

Input combinations for ImportObject method

WgnRDI.dll does not support all message Format- Type-Transport permutations.

Return values for ImportObject method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

0xE5630D2E

Indicates an unknown import source. This value is returned if the importSource parameter is set to an invalid number.

0xE5630D18

"A mail event could not be created because the available disk space was less than the configured threshold."

This error message may indicate that the cache is not being cleared quickly enough. If you get this error, wait before retrying to import the e-mail. If you continue to get this error, then you must halt the archive process and flag an alert.

More information:

Values for importSource (see page 70)
Input Combinations for ImportObject() Method (see page 63)

Method GetStatusString

Following a successful import operation, the return code is: S_OK. If an error occurs, use this method to find the string value (that is, an explanation) of the error.

```
HRESULT GetStatusString(
    [in] long            hResult,
    [out,retval] BSTR*    statusString
};
```

Return values for GetStatusString method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

More information:

Error Codes (see page 75)

Interface IWgnImportConnector3

Use this interface to import messages from a specified source.

Note: This interface allows customers to pass to the RDI additional message attribute data to be stored with the event on the CMS. It supersedes earlier interfaces.

More information:

Interface Versions (see page 13)
Error Codes (see page 75)

COM Object WGNRDILib::WgnImportConnector

Co-create this COM object, then query the object for the interface and use the methods listed below.

Method ImportObject

Use this method to import a message.

```
HRESULT ImportObject(
    [in] WGN_IMPORTOBJ_TYPE       objectType,
    [in] WGN_IMPORTOBJ_FORMAT     objectFormat,
    [in] WGN_IMPORTOBJ_TRANSPORT  objectTransport,
    [in] VARIANT                  objectData,
    [in] BSTR                     messageID,
    [in] DWORD                    importSource,
    [in] BSTR                     XmlMessageAttributes
);
```

Where:

objectType

Specifies the type of object to import (e-mail or IM). For details, see the typedef WGN_IMPORTOBJ_TYPE (see page 59).

objectFormat

Specifies the format of the object to import. For details, see the typedef WGN_IMPORTOBJ_FORMAT (see page 59).

objectTransport

Specifies the method used to transfer data to the RDI. For details, see the typedef WGN_IMPORTOBJ_TRANSPORT (see page 60).

objectData

Specifies the source data being imported.

messageID

Is a string that uniquely identifies the message in the archive. This will be the string given to CA's Remote Data Manager (RDM) to enable it to retrieve the message contents.

Important! The messageID parameter is critical! You can use any string as the identifier but it must be **unique** and the identifiers must be managed correctly to ensure that CA DataMinder can retrieve the full details for imported e-mails. Your application must be able to use this string, when called by the RDM, to access the exact same message.

importSource

Is a fixed integer that identifies the source archive. If there is no link to a remote archive, it is WGN_ACTIVEIMPORTSOURCE_NONE.

xmlMessageAttributes

Optional XML containing additional metadata for the message.

Input combinations for ImportObject method

WgnRDI.dll does not support all message Format-Type-Transport permutations.

Return values for ImportObject method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

0xE5630D2E

Indicates an unknown import source. This value is returned if the importSource parameter is set to an invalid number.

0xE5630D18

"A mail event could not be created because the available disk space was less than the configured threshold."

This error message may indicate that the cache is not being cleared quickly enough. If you get this error, wait before retrying to import the e-mail. If you continue to get this error, then you must halt the archive process and flag an alert.

More information:

Values for importSource (see page 70)
Input Combinations for ImportObject() Method (see page 63)
Specification for XmlMessageAttribute Data (see page 68)

Method GetStatusString

Following a successful import operation, the return code is: S_OK. If an error occurs, use this method to find the string value (that is, an explanation) of the error.

```
HRESULT GetStatusString(
    [in] long              hResult,
    [out,retval] BSTR*     statusString
};
```

Return values for GetStatusString method

E_INVALIDARG 0x80070057

Indicates that an invalid argument was passed to the method.

More information:

Error Codes (see page 75)

Interface IWgnRDIConfig

Use this interface to override registry values for WgnRDI.dll for a particular COM instance object. The ImportMAPIMessage() or ImportObject() methods use these values instead of the registry ones. The registry values are used if the methods on this interface are not invoked.

COM object WGNRDILib::WgnImportConnector

Co-create this COM object, then query the object for the methods listed below.

Method SetMinDiskSpaceThresholdMB

Use this method to programmatically set the minimum level of free disk space on the machine hosting the EVF file cache.

```
HRESULT SetMinDiskSpaceThresholdMB(
    [in] DWORD    dwMinDiskSpaceMB
);
```

Where:

dwMinDiskSpaceMB

Specifies a minimum level of free disk space on the machine hosting the EVF file cache. If free disk space falls below this level, no further messages are extracted from the e-mail archive to the EVF file cache until free disk space increases. This threshold defaults to 10 MB.

This method overrides the level specified by the MinFreeDiskMb registry value on the RDI host machine. For further details, see the *Archive Integration Guide*; search for 'Configure the External Agent API'.

Return values for SetMinDiskSpaceThresholdMB method

There is no validation for this method, so no errors are returned.

Method SetOutputFolder

Use this method to programmatically specify the full path to the EVF file cache.

Note: If the specified folder does not exist, it will be created automatically.

```
HRESULT SetOutputFolder(
    [in] BSTR    szfolderPath
);
```

Where:

szfolderPath

Specifies the full path to the EVF file cache—the folder into which the e-mails will be written. This path defaults to the \system\data\RDIcache subfolder of the CA DataMinder installation folder.

This method overrides the level specified by the EventFilePath registry value on the RDI host machine. For further details, see the *Archive Integration Guide*; search for 'Configure the External Agent API'.

Return values for SetOutputFolder method

This method can return the same errors as the Win32 CreateDirectory() API.

Chapter 7: API Typedefs

This section contains the following topics:

Typedef WGN_IMPORTOBJ_TYPE

This enumerates the supported types of import operation. The RDI can currently import e-mail and IM data.

```
typedef [v1_enum] enum
{
    WGN_IMPORTOBJ_TYPE_EMAIL       = 0,
    WGN_IMPORTOBJ_TYPE_IM          = 1,
    WGN_IMPORTOBJ_TYPE_FILE        = 2,
    WGN_IMPORTOBJ_TYPE_PRECREATED  = 3
}
WGN_IMPORTOBJ_TYPE;
```

Typedef WGN_IMPORTOBJ_FORMAT

This enumerates the supported formats for imported data

```
typedef [v1_enum] enum
{
    WGN_IMPORTOBJ_FORMAT_MAPI        = 0,
    WGN_IMPORTOBJ_FORMAT_RFC822      = 1,
    WGN_IMPORTOBJ_FORMAT_NOTES       = 2,
    WGN_IMPORTOBJ_FORMAT_EVF         = 3,
    WGN_IMPORTOBJ_FORMAT_FILE        = 4,
    WGN_IMPORTOBJ_FORMAT_PRECREATED  = 5
}
WGN_IMPORTOBJ_FORMAT;
```

Where EVF refers to CA DataMinder event files and RFC822 refers to Internet e-mails.

Typedef WGN_IMPORTOBJ_TRANSPORT

This enumerates the methods used to transfer data to the RDI.

```
typedef [v1_enum] enum
{
    WGN_IMPORTOBJ_TRANSPORT_COM       = 0,
    WGN_IMPORTOBJ_TRANSPORT_FILENAME  = 1,
    WGN_IMPORTOBJ_TRANSPORT_MEMORY    = 2,
    WGN_IMPORTOBJ_TRANSPORT_ISTREAM   = 3,
    WGN_IMPORTOBJ_TRANSPORT_HANDLE    = 4
}
WGN_IMPORTOBJ_TRANSPORT;
```

Typedef WGN_ACTIVEIMPORT_DESTINATION

This enumerates the External Agent output destinations: a local PE hub or a local policy engine.

```
typedef [v1_enum] enum
{
    WGN_ACTIVEIMPORT_DESTINATION_HUB    = 0,
    WGN_ACTIVEIMPORT_DESTINATION _PE    = 1,
}
WGN_ACTIVEIMPORT_DESTINATION;
```

Typedef WGN_ACTIVEIMPORT_HUB_THROTTLE_MODE

This enumerates the PE hub throttling modes: Wait or Fail.

```
typedef [v1_enum] enum
{
    WGN_ACTIVEIMPORT_THROTTLE_FAIL_EVENT    = 0,
    WGN_ACTIVEIMPORT_THROTTLE_WAIT_EVENT    = 1,
}
WGN_ACTIVEIMPORT_HUB_THROTTLE_MODE;
```

Typedef WGN_ACTIVEIMPORT_RESULTXML_FLAGS

These flags can be combined to choose what the XML returned from a call to IWgnImportConnectorResult::GetResultXML() contains. Note that the only result XML currently available is SmartTags.

```
typedef [v1_enum] enum
{
    WGN_ACTIVEIMPORT_RESULTXML_NONE       = 0,
    WGN_ACTIVEIMPORT_RESULTXML_SMARTTAGS  = 0x00000001,
    WGN_ACTIVEIMPORT_RESULTXML_ALL        = 0xFFFFFFFF
}
WGN_ACTIVEIMPORT_RESULTXML_FLAGS;
```

Typedef WgnSmartTag: struct

This enumerates a client-supplied structure populated with the name and value of the requested smart tag.

```
typedef struct
{
    LPOLESTR    szName;
    LPOLESTR    szValue;
}
WgnSmartTag;
```

Chapter 8: Input Combinations for ImportObject() Method

WgnRDI.dll does not support all message Format-Type-Transport permutations. The supported input combinations are summarized in the following sections.

This section contains the following topics:

MAPI Message as IMessage

Support introduced in CA DataMinder 4.5

```
objectFormat     WGN_IMPORTOBJ_FORMAT_MAPI
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_COM
objectData       Variant type: VT_UNKNOWN    IUnknown* interface
                                             from IMessage* interface
```

MAPI Message as IStream

Support introduced in CA DataMinder 4.5

```
objectFormat     WGN_IMPORTOBJ_FORMAT_MAPI
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_ISTREAM
objectData       Variant type: VT_UNKNOWN  IUnknown* interface from
                                           IStream* interface to source data
```

MAPI Message from Filename

Support introduced in CA DataMinder 4.5

```
objectFormat     WGN_IMPORTOBJ_FORMAT_MAPI
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_FILENAME
objectData       Variant type: VT_BSTR    Full path to file being imported
```

Internet Email EML from Filename

Support introduced in CA DataMinder 4.7

```
objectFormat     WGN_IMPORTOBJ_FORMAT_RFC822
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_FILENAME
objectData       Variant type: VT_BSTR    Full path to file being imported
```

Internet Email EML from IStream

Support introduced in CA DataMinder 4.7

```
objectFormat     WGN_IMPORTOBJ_FORMAT_RFC822
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_ISTREAM
objectData       Variant type: VT_UNKNOWN    IUnknown* interface from
                                             IStream* interface to source data
```

Lotus Notes Email from Notes Handle

Support introduced in CA DataMinder 4.7

```
objectFormat     WGN_IMPORTOBJ_FORMAT_NOTES
objectType       WGN_IMPORTOBJ_TYPE_EMAIL
objectTransport  WGN_IMPORTOBJ_TRANSPORT_MEMORY
objectData       Variant type: VT_I4|VT_BYREF  Pointer to NOTEHANDLE,
                                               an IBM Lotus Notes SDK type
```

File from Filename

Support introduced in CA DataMinder 4.7

```
objectFormat     WGN_IMPORTOBJ_FORMAT_FILE
objectType       WGN_IMPORTOBJ_TYPE_FILE
objectTransport  WGN_IMPORTOBJ_TRANSPORT_FILENAME
objectData       Variant type: VT_BSTR  Full path to file being imported
```

File from IStream

Support introduced in CA DataMinder 4.7

```
objectFormat     WGN_IMPORTOBJ_FORMAT_FILE
objectType       WGN_IMPORTOBJ_TYPE_FILE
objectTransport  WGN_IMPORTOBJ_TRANSPORT_ISTREAM
objectData       Variant type: VT_UNKNOWN  IUnknown* interface from
                                           IStream* interface to source data
```

File Pre-Created from Filename

Support introduced in CA DataMinder 4.7

Initialise IWgnImportFileObject by calling InitialiseFile() with:

```
objectTransport  WGN_IMPORTOBJ_TRANSPORT_FILENAME
objectData       Variant type: VT_BSTR  Full path to file being imported
```

Import the pre-created file object with:

```
objectFormat     WGN_IMPORTOBJ_FORMAT_PRECREATED
objectType       WGN_IMPORTOBJ_TYPE_PRECREATED
objectTransport  WGN_IMPORTOBJ_TRANSPORT_COM
objectData       Variant type: VT_UNKNOWN  IUnknown* interface
                                           from IWgnImportFileObject.
```

File Pre-Created from IStream

Support introduced in CA DataMinder 4.7

Initialise IWgnImportFileObject by calling InitialiseFile() with:

```
objectTransport  WGN_IMPORTOBJ_TRANSPORT_ISTREAM
objectData       Variant type: VT_UNKNOWN  IUnknown* interface from
                                           IStream* interface to source data
```

Import the pre-created file object with:

```
objectFormat     WGN_IMPORTOBJ_FORMAT_PRECREATED
objectType       WGN_IMPORTOBJ_TYPE_PRECREATED
objectTransport  WGN_IMPORTOBJ_TRANSPORT_COM
objectData       Variant type: VT_UNKNOWN  IUnknown* interface
                                           from IWgnImportFileObject.
```

Chapter 9: Common Parameter Details

This section provides about parameters common to both the IWgnActiveImportConnector and IWgnImportConnector3 interfaces.

This section contains the following topics:

Specification for XmlMessageAttribute Data

XmlMessageAttributes is an XML string that specifies additional data to be stored on the CMS as attributes of the imported event. For the current release, two attributes are supported—see below.

The XML string specified by XmlMessageAttributes must conform to the following structure. For the current version of this interface, only two message attributes are supported, ProcessAs and IMNetworkName (highlighted below). Note also that the schema definition is installed automatically when you install the External Agent.

The message attrirubte XML schema differs for the IWgnImportConnector3 and IWgnActiveImportConnector interfaces.

The following is an example of additional information XML for the IWgnImportConnector3 interface:

```
<?xml version="1.0" encoding="utf-16" ?>
<ImportConnector>
  <AdditionalInputData>
    <ProcessAs>IM</ProcessAs>
    <IMNetworkName>MSN IM</IMNetworkName>
  </AdditionalInputData>
</ImportConnector>
```

The following is an example of additional information XML for the IwgnActiveImportConnector interface:

```
<?xml version="1.0" encoding="utf-16" ?>
<apm>
  <event>
    <email>
      <embedding>
        <process_as>IM</process_as>
        <network_name>MSN IM</network_name>
      </embedding>
    </email>
  </event>
</apm>
```

Where:

ProcessAs / process_as

This attribute defines the event type for item being imported. It can take three possible values:

IM

Flags the imported event as an IM conversation. If ProcessAs is set to IM, the RDI also saves the IM network name, as specified by IMNetworkName.

eFax

Flags the imported event as an electronic fax.

Set ProcessAs is set to eFax when importing fax messages saved as MAPI messages.

Bloomberg

Flags the imported event as a Bloomberg e-mail.

Set ProcessAs is set to Bloomberg when importing Bloomberg e-mails saved as MAPI messages. These are e-mails that were originally sent using Bloomberg terminals but which have been subsequently saved as MAPI messages.

IMNetworkName / network_name

Note: This attribute is only supported if ProcessAs is set to IM.

This identifies the name of the IM network associated with the IM conversation being imported. The example above (MSN IM) indicates an import operation for MSN instant messaging data.

Specifically, it defines the string values that get saved in the CMS database. To allow iConsole users to search for IM conversations by IM network, the same string values must be specified in an iConsole search definition file. It is essential that you consistently use the same string for a specific IM network.

For this reason, we strongly recommend that you adhere to strings defined in the XML schema for IMNetworkName. The same strings are used in the iConsole Standard Search to allow users to search by IM network. The strings in the schema are:

"AOL IM"

"Bloomberg"

"GoogleTalk"

"HubConnex"

"ICQ IM"

"IMTrader"

"Indii"

"Jabber"

"Microsoft Exchange IM"

"Mindalign"

"MS LCS IM"

"MSN IM"

"Reuters IM"

"SameTime"

"WebEx"

"Yahoo! IM"

"YahooEE IM"

Note: If you want to validate attributes for XmlMessageAttributes against the XML schema definition, note that the schema, ImportConnector.XSD, is installed automatically when you install the RDI. You can find this schema in the \Client subfolder of the CA DataMinder installation folder.

SmartTag XML Specification

The following is an example of XML returned from a call to IWgnImportConnectorResult::GetResultXML() with WGN_ACTIVEIMPORT_RESULTXML_SMARTTAGS or IWgnSmartTag::GetSmartTagXML().

```
<?xml version="1.0" encoding="utf-16" ?>
<apm>
  <policy>
    <state>
      <smart_tags>
        <smart_tag name="SmartTag Name">
          <value>SmartTag Value1</value>
          <value>SmartTag Value2</value>
        </smart_tag>
      </smart_tags>
    </state>
  </policy>
</apm>
```

Values for importSource

importSource is a fixed integer that identifies the import source, such as a third party archive. CA Technologies assigns unique values to each archive, but the following special values are also available:

WGN_ACTIVEIMPORTSOURCE_NONE

Specifies that there is no link to a remote archive.

99

For testing purposes.

80 - 89 (inclusive)

Specifies a custom archive. These values are available for 3rd-party use when developing integrations with custom archives.

More information:

Testing an External Agent Implementation (see page 79)
Custom Sources (see page 73)

Chapter 10: Custom Sources

If integrating with a custom archive not supported by CA DataMinder, set importSource to a value from the custom range (see page 70). It is your responsibility to ensure that the messageId parameter to the Import methods is unique for each item passed to the API, and to ensure that messageId is sufficient to allow retrieval via the Remote Data Manager.

Currently the Remote Data Manager only supports file and email data types for custom sources, so use only these with the objectType parameter.

Chapter 11: Error Codes

This section contains the following topics:

Error Code Summary

Following a successful import operation, the return code from the External Agent is S_OK.

But if the import operation fails, an error code is returned to the source application. This section provides a summary interpretation for these error codes. Note that the returned code codes will include those generated by Microsoft APIs; these are not listed here.

- 0xE5630D18 "A mail event could not be created because the available disk space was less than the configured threshold."

  This error message may indicate that the cache is not being cleared quickly enough. If you get this error, wait before retrying to import the e-mail. If you continue to get this error, then you must halt the archive process and flag an alert.

- Any other error code that starts with 0xe563...

  These are CA DataMinder errors. You must consider these to be a fatal error for the associated e-mail. Typically, this indicates that the data passed to the External Agent was invalid.

- Any other error codes are Microsoft errors. As above, you must consider these to be a fatal error for the associated e-mail.

Note: We strongly recommend that you log all error codes and their associated text representation, to allow problems to be diagnosed subsequently.

The External Agent API Library exports symbols for some of the most common success and error codes. These are described in the topics that follow.

WGN_S_ACTIVEIMPORT_PENDING = 0x25632B00

Result pending asynchronous processing. This result is returned from a successful call to IwgnActiveImportConnector::ImportObjectAsync().

WGN_S_ACTIVEIMPORT_AWAITING_COMPLETION = 0x25631356

The policy engine has successfully completed event processing and is waiting for the caller to commit or abort (rollback) the event.

WGN_E_ACTIVEIMPORT_IMPORT_OBJECT_NOT_FOUND = 0xE5632B01

The Import Object for the supplied interface could not be found. This error can indicate that the object supporting the IWgnImportFileObject interface is invalid.

WGN_E_ACTIVEIMPORT_POLICY_ENGINE_SERVICE_NOT_REG = 0xE5631346

The Policy Engine server service or Hub is not registered.

WGN_E_ACTIVEIMPORT_ATTEMPTED_CHANGE_CLIENT_TYPE = 0xE5631A94

The Policy Engine Hub can currently only support one type of event (Email or File). If mixed event types are imported by the same machine, this error will be returned from IWgnActiveImportConnector::ImportObject() or ImportObjectAsync()

WGN_E_ACTIVEIMPORT_INVALID_XML = 0xE5632B03

The additional information XML passed to IWgnActiveImportConnector::ImportObject() or ImportObjectAsync() is invalid.

WGN_E_ACTIVEIMPORT_COMMIT_OR_ROLLBACK_ALREADY_CALLED= 0xE5632B04

This error will be returned if IWgnImportConnectorDeferredCompletion::Commit(), CommitAsync() or Rollback() has already been successfully called.

WGN_E_ACTIVEIMPORT_IMPORT_CAUGHT_EXCEPTION = 0xE5632B05

An exception was raised within the EA API.

WGN_E_ACTIVEIMPORT_CHANGING_SYNC_MODE = 0xE5632B06

An attempt was made to commit an event using a different synchronous mode to the import. For example, IWgnImportConnectorDeferredCompletion::CommitAsync() was called for an event imported using IWgnActiveImportConnector::ImportObject().

Chapter 12: Testing an External Agent Implementation

This section contains the following topics:

Apply Policy and Generate CA DataMinder Events

To set up the External Agent to apply policy to e-mails origination from a new product

1. Set up a Hub and/or Policy Engines.
2. Configure policy so that smart tags are returned, but the events are not saved. If events are saved, the CMS will become contaminated with the test data.
3. Set the importSource parameter to 99; this value is reserved for testing the External Agent API.

Generate EVF Files

To set up the External Agent to convert e-mails originating from a new product into EVF files

1. Set the importSource parameter to **99**; this value is reserved for testing the RDI. This parameter is used by:
   - ImportMAPI method for the IWgnImportConnector interface.
   - ImportObject method for the IWgnImportConnectors interface.
2. The RDI utility will generate EVF files for each extracted e-mail and write these files to the cache specified by the SetOutputFolder method or the EventFilePath registry value. You can then send the EVF files to CA for confirmation that e-mail events have been successfully created.
3. Configure an Event Import job to import the EVF files from the cache. See the *Archive Integration Guide* for details; search for 'Event Import utility'.

More information:

Method ImportMAPIMessage (see page 50)
Method ImportObject (see page 52)
Method SetOutputFolder (see page 58)

Appendix A: Accessibility Features

CA Technologies is committed to ensuring that all customers, regardless of ability, can successfully use its products and supporting documentation to accomplish vital business tasks. This section outlines the accessibility features that are supported by CA DataMinder.

Display

To increase visibility on your computer display, you can adjust the following options:

Font style, color, and size of items

Defines font color, size, and other visual combinations.

The CA DataMinder iConsole also supports a High Visibility mode. This increases the size of text and images in the iConsole screens.

Screen resolution

Defines the pixel count to enlarge objects on the screen.

Cursor width and blink rate

Defines the cursor width or blink rate, which makes the cursor easier to find or minimize its blinking.

Icon size

Defines the size of icons. You can make icons larger for visibility or smaller for increased screen space.

High contrast schemes

Defines color combinations. You can select colors that are easier to see.

Sound

Use sound as a visual alternative or to make computer sounds easier to hear or distinguish by adjusting the following options:

Volume

Sets the computer sound up or down.

Text-to-Speech

Sets the computer's hear command options and text read aloud.

Warnings

Defines visual warnings.

Notices

Defines the aural or visual cues when accessibility features are turned on or off.

Schemes

Associates computer sounds with specific system events.

Captions

Displays captions for speech and sounds.

Keyboard

You can make the following keyboard adjustments:

Repeat Rate

Defines how quickly a character repeats when a key is struck.

Tones

Defines tones when pressing certain keys.

Sticky Keys

Defines the modifier key, such as Shift, Ctrl, Alt, or the Windows Logo key, for shortcut key combinations. Sticky keys remain active until another key is pressed.

Mouse

You can use the following options to make your mouse faster and easier to use:

Click Speed

Defines how fast to click the mouse button to make a selection.

Click Lock

Sets the mouse to highlight or drag without holding down the mouse button.

Reverse Action

Sets the reverse function controlled by the left and right mouse keys.

Blink Rate

Defines how fast the cursor blinks or if it blinks at all.

Pointer Options

Let you do the following:

- Hide the pointer while typing
- Show the location of the pointer
- Set the speed that the pointer moves on the screen
- Choose the pointer's size and color for increased visibility
- Move the pointer to a default location in a dialog box