



**Open Connector API:**  
*Migrating from CES 4 SP3 to CES 5*



## General Guidelines

The following lists the different modifications brought to the Open Connector API:

Modification	Description
The simplicity of the SimpleCrawler class is incorporated with the CustomCrawler class	The SimpleCrawler class is now deprecated; however, the new CustomCrawler class is as simple to use and much more powerful. If you inherited from SimpleCrawler, from now on simply inherit from CustomCrawler instead.
The Advanced Features are implemented as required in the interfaces	Before, the different methods from the CustomCrawler class were always overridden; however, several of these methods were special features, such as Live Indexing, and were not necessarily required. Now, it is possible to implement only the interfaces corresponding to the required methods. For example, to specifically enable the Live Indexing method, implement the ILiveIndexing interface and the required methods. Without this interface, the <b>Enable Live Indexing</b> button displayed in other connectors is not available. Note that there are many interfaces available to improve the connector or enable features.
Several classes not required by an open connector are internalized	Several classes that should not have been displayed are now hidden. These specific classes are internal to the workings of the API and connectors should not be calling on them directly. By doing so, it is simpler to create new connectors that are only focusing on the appropriate methods.
The log functionality is simplified	In version 5, the additional connectors can only log in the Index Log, not the System Log. Because it is not logical for a connector to register an error in the System Log, it has been disabled. The log-in methods have been simplified; meaning that the majority of log-in procedures are now done automatically, for example the unchanged document message. The generic method to use that renders the most control over what is logged is CrawlingContext.LogMessage.
Methods are moved to strategic locations	In order to simplify the creation of new connectors, several methods have been moved to a more appropriate class. For example, several methods concerning indexing have been moved from CrawlingContext—which is more related to source configuration—to the CustomCrawler class—which is the indexing tool.
The new GetActionForDocument method determines what to do with newly crawled documents	A new method—GetActionForDocument—has been added in the CustomCrawler class in order to determine if the newly crawled file is indexed or ignored. It takes the document in parameter and returns the action to perform. This document does not need to have all of its information entered. It only requires the URI it is filtered by, as well as the ModifiedDate. If it is unchanged or ignored, it is possible to skip the rest of the document creation and access the next one. This method also checks document and content types if the Document.FileExtension and/or Document.ContentType property has been set.
The Open Connector API gives access to the new CES UserIdentities	The new feature—UserIdentities—allows to create an identity containing the username and password values, and access this identity from anywhere in CES. Because crawling often requires an identity, UserIdentities are available in the Open Connectors API. Directly in the CrawlingContext class, there are three properties that allow to access this information: HasUserIdentity, UserName and UserPassword. Note that because it is possible not to set an identity, HasUserIdentity can be false. If so, use the identity of the user currently running the service. If that is not possible for the connector, throw an exception to require an identity.

Rebuild and Refresh operations keep the index as clean and light as possible

Whenever a rebuild/refresh finishes, all the URIs of the index that have not been crawled during the rebuild/refresh are automatically removed. This can generate several problems; meaning that if an error occurs that does not prevent the rebuild/refresh to finish, but prevents many URIs from being found, they will be removed from the index. Adequate error handling is necessary to have a robust connector. When an exception occurs that prevents documents from being found, but that does not stop the crawling process, it is possible to use the `ReportUriToIgnoreForDeletion` method. This method takes the URI as a parameter and tags it, as well as its children. When the rebuild finishes, it considers that the URIs are good and does not remove them. Note that this feature is important in order to remove documents that no longer exist or that the crawling user does not have access to.

## Class Changes – Removed/Modified Items

### Coveo.CES.CustomCrawlers.CommandDispatcher

This class is no longer public. The operations that were available using this class are now performed automatically by the Open Connector API when necessary, either as a reaction to CES, user actions or through higher-level calls in other classes.

Method	Description
AddDocument	This method is called by IndexDocument. If AddDocument is used directly on the command dispatcher, simply use the CustomCrawler.IndexDocument method using the document created.
CancelCurrentOperations	This method is internal.
CommitPendingCommand	This method is internal.
GetChildDirectoriesFromURI	This method is internal.
GetChildFilesFromURI	This method is internal.
GetDocumentInformation	This method is internal. The DocumentInformation class is for internal use only and the operations that were available with the class are now performed in a simpler manner, directly on a document or through the connector.
GetIndexedDocumentType	This method is internal.
GetURIsFromSource	This method is internal.
HasChildren	This method is internal.
IsDocumentIndexed	This method is internal. It is not required to determine if the document is in the index. If it is, and does not need to be indexed again, the IndexDocument method logs an <i>Unchanged</i> entry. Methods applied on a document, such as IsOutOfDate and IsSecurityUpdated, allow to determine if the document must be indexed manually and if the ReportUnchangedDocument must be called to tag it. It is suggested to call IndexDocument.
LogCrawlerMessage	This method no longer exists. Log-in methods are available in the CrawlingContext class.
LogMessage	This method is internal. Logging methods are available in the CrawlingContext class.

RemoveDocument	This method is internal. A method that removes documents from the index is available in the CustomCrawler class.
RemoveOlderThan	This method is internal. It has been moved to the CustomCrawler class.
SourceIsStopped	This method is internal. Source status is handled by the Open Connector API.
StopSource	This method is internal and is handled automatically.
CanAddDocument	This method is internal.

### Coveo.CES.CustomCrawlers.CrawlingContext

Method	Description
Constructor	This method is no longer public.
GetActionForDocument	The equivalent method is now a member of the CustomCrawler class.
GetActionForDocumentOnError	This method is internal.
GetConfigValue	This method is now virtual if ever it needs to be overridden. By default, it works as it did before.
GetDocumentInformation	This method is no longer accessible.
GetSourceUriEnumerator	This method returns IEnumerable<string> instead of IEnumerator.
IndexDocument	This function was moved to the CustomCrawler class.
LogSystemMessage	This method no longer exists. Using the LogMessage method with parameters as well as an empty ("") URI, Severity.Normal and Operation.Unspecified, displays the closest result.
LogStartCrawling	This method no longer exists. The CustomCrawler class enters a log when a refresh or rebuild operation starts.
RemoveDocument	This method is no longer a member of the CrawlingContext class. From now on, it is a member of the CustomCrawler class.
LogDoneCrawling	This method no longer exists. The CustomCrawler class creates a log entry automatically when a source refresh or rebuild finishes.
CommitPendingCommands	This method is internal. The mechanic to commit pending commands is now handled by the CustomCrawler class whenever it is appropriate.
LogSystemErrorMessage	This method no longer exists. Using the LogMessage method with parameters as well as an empty ("") URI, Severity.Error and Operation.Unspecified, displays the closest result.
ShouldCrawlUri	This method no longer exists, as it is obsolete because of the CustomCrawler.GetActionForDocument method. By creating a partial document and calling the GetActionForDocument method, the return value indicates what to do with the document. Generally, it needs to be indexed. If it is unchanged or filtered, the appropriate log entry is generated by the CustomCrawler class.
IsDocumentOutOfDate	This method no longer exists from the context and was replaced by the new function Document.IsDocumentOutOfDate. To correctly call this method, set the URI and ModifiedDate parameters.
LogFilteredByUriMessage	This method no longer exists. When calling GetActionForDocument, if a document is ignored because of the filters, a log entry is automatically created.

LogUnchangedMessage	This method no longer exists. An <i>Unchanged</i> log entry is automatically created when IndexDocument is called on a unchanged document or no changes are reported when using ReportUnchangedDocument
LogMessage	This method no longer requires a parameter of LogType type. The LogType class is no longer accessible. The messages a connector can log are now always of Crawler type. The System type is no longer available. Remove the obsolete parameter to fix compile errors.
LogRemovedMessage	This method no longer exists. A <i>Removed</i> log entry is automatically added when a document is removed from the index using the RemoveDocument method.
SourcelsStopped	This method no longer exists. The connector status is automatically handled.
StopSource	This method no longer exists. The connector status is automatically handled.

### Coveo.CES.CustomCrawlers.CustomCrawler

Method	Description
PerformStart	Before, the main crawling thread had to be started if it was inherited from CustomCrawler. The main crawling thread is now handled by the Open Connector API. Override the Run method instead and insert the code you had in the threadstart in the Run method. Run becomes the main thread and is called when a rebuild or refresh operation is started.
PerformStop	The main thread is now handled by the CustomCrawler class; therefore remove this method. When a stop is required, an exception of ThreadAbortException type is thrown and the crawling process is stopped. It is important not to catch all exceptions with a generic catch (Exception) that blocks exceptions it does not handle. It also catches the thread abortion and prevents the connector from stopping before it finishes.
PerformPause	This method is no longer required; the CustomCrawler class now handles the <i>Pause</i> of the crawling process. To use the Pause/Resume feature, implement the ISaveRestoreState interface.
PerformResume	Refer to the PerformPause method
PerformEnableLiveIndexing	This method is no longer required. To use live indexing, the ILiveIndexing interface must be implemented.
PerformDisableLiveIndexing	Refer to the PerformEnableLiveIndexing method.
PerformUpdateConfig	This method is no longer required. The Open Connector API handles the operations necessary to update the configuration. To keep the connector up to date, override the Update call, which is called after a modification to a source configuration and before the next operation the connector performs. In the Update call, make sure to update all the configuration values the connector requires as well as the necessary operations that go along with them. The operations performed in PerformUpdateConfig should be similar to those in Update, but without any need for locks, for example.
PerformGetCapabilities	This method is no longer required. The capabilities are automatically determined by the interfaces the connector implements.
PerformBeautifyURI	This method is no longer required. To use the BeautifyUri method, implement the IBeautifyUri interface.

PerformDeleteURI	This method is no longer required. To allow the deletion of an URI in the index, implement the IDeleteDocument interface.
PerformRefreshURI	This method is no longer required. To allow the refresh of single documents, implement the IRefreshDocument interface.
PerformInitialize	This method is now “protected virtual” instead of “protected abstract”; meaning it does not need to be implemented if it is not required.
PerformShutdown	Refer to the PerformInitialize method.
FinishedRefreshing	This method no longer exists. The Open Connector API knows the Refresh is executed when the Run method returns.
BeautifyURI	This method is internal. To beautify the URI, implement the IBeautifyUri interface in the connector.
DeleteUri	This method is internal. To remove documents from the index, use the RemoveDocument, RemoveDocumentAndChildren or RemoveOlderThan method.
DisableLiveIndexing	This method is internal. Disabling the live indexing is now handled automatically by the Open Connector API.
EnableLiveIndexing	This method is internal. Enabling the live indexing is now handled automatically by the Open Connector API.
GetCapabilities	This method is internal. The capabilities are automatically determined by the interfaces the connector implements.
Initialize	This method is internal.
Pause	This method is internal. The state of the connector is automatically handled by the CustomCrawler class.
RefreshUri	This method is internal. To allow the refresh of single documents, implement the IRefreshDocument interface.
Resume	This method is internal. To use the Pause/Resume feature, implement the ISaveRestoreState interface.
Shutdown	This method is internal.
Start	This method is internal.
Stop	This method is internal.
UpdateSourceConfig	This method is internal.
Interrupted	This property no longer exists.
LiveIndexingEnabled	This method no longer exists. To validate if live indexing is currently enabled, call IsLiveIndexingThread. Enabling or disabling the live indexing is done automatically.
SourceState	This method no longer exists. The state is handled automatically.
Status	This method no longer exists.

## Coveo.CES.CustomCrawlers.Document

The Document class implements the .NET interface IDisposable. It is strongly suggested to use the “using” keyword when instantiating this class or dispose of it as soon as it is no longer required, as it holds COM resources that the .NET garbage collector is not aware of. By disposing of these objects and releasing the resources rapidly, you are avoiding potential memory and handle issues.

Method	Description
Fields	This attribute no longer exists. It is replaced by the Document.SetField method. Use this new method to set the fields on a document.
Lock	This method no longer exists. When a document is ready, index it using CustomCrawler.IndexDocument
ResolveModifiedDate	This method no longer exists. Documents are not forced to have a modified date. The IsOutOfDate method always returns <i>True</i> for a document that has no modified date.
Unlock	This method no longer exists.
Data	This property no longer has a get accessor. It is only possible to set the data on the document.
Metadata	This property no longer has a get accessor. It has gained a set accessor in order to allow a Hashtable containing metadata to be set on a document.
Security	This property no longer has exists.

## Coveo.CES.CustomCrawlers.DocumentInformation

This class is now internal. The necessary information is available in the Document class.

## Coveo.CES.CustomCrawlers.DocumentMapping

Method	Description
ResolveMapping	The second parameter of this method is no longer IDictionary, it is now Document. To get metadata value from a document, use the Document.GetMetaDataValue method.

## Coveo.CES.CustomCrawlers.InformationSource

This class no longer exists. It only duplicates ways of getting information that already exists in other classes. Configuration values can be fetched using the Context. URI checks can be done on the document.

Method	Description
GetExtensionSettingsAction	Use the CustomCrawler.GetActionForDocument method to enable the appropriate action. It takes into account the extension of the file.
GetExtensionSettingsAction OnError	This method no longer exists.
GetProperty	Use the Context.GetConfigValue method to retrieve a source parameter.
GetUID	These methods no longer exist. The UID is not something a connector should worry about.
isURIAllowed	This method no longer exists; use GetActionForDocument to test the filters.
Beautify	This property no longer exists, to modify the URI before crawling, implement the IBeautifyUri interface.
CaseSensitive	This property no longer exists; use the Context.CaseSensitive property to validate the source parameter.

CollectionID	This property no longer exists. The collection ID is not something a connector has to validate.
CollectionName	This property no longer exists. The Collection Name is not required by a connector.
Crawler	This property no longer exists. Inheriting from a connector gives access to the connector either by using the keywords base or this.
CrawlingDepth	This property no longer exists.
FilterDuplicates	This property no longer exists.
GenerateHTML	This property no longer exists.
Id	This property no longer exists.
IndexMeta	This property no longer exists. To validate if it is required to index the metadata, use the Context.IndexMeta property.
IndexSecurity	This property no longer exists. To validate if it is required to index the securities, use the Context.IndexSecurity property.
Name	This property no longer exists.
NbCustomFields	This property no longer exists.
PhysicalIndexId	This property no longer exists.
Priority	This property no longer exists.
Recursive	This property no longer exists. To validate if the index needs to be indexed recursively, verify the Context.Recursive property.
StartingURIs	This property no longer exists. To get the source URIs, use the Context.Uris property, which returns an array of string.
Summarize	This property no longer exists.
UID	This property no longer exists.

### **Coveo.CES.CustomCrawlers.SimpleCrawler**

This class is deprecated. If you inherited from the SimpleCrawler class, modify the connector to inherit from the CustomCrawler class. The current CustomCrawler class is as simple as the SimpleCrawler class; however, it offers more methods and properties.

## **Class Changes – New Items**

### **Coveo.CES.CustomCrawlers.CrawlerException**

### **Coveo.CES.CustomCrawlers.CrawlerIgnorableException**

### **Coveo.CES.CustomCrawlers.CrawlerFatalException**

These three exceptions are types made available by the Open Connector API. CrawlerException is the basic type. Almost all exceptions pertaining to crawling can be put into two categories: IgnorableExceptions and FatalExceptions. Ignorable Exceptions are errors that do not stop the crawling process, for example finding a corrupted document or permission denial within the scope of the crawling process. Basically, anything that does not prevent this crawling pass from finishing. These exceptions are usually logged as warnings. However, there are cases of fatal exceptions, such as connection issues, where the crawling process is stopped. As explained previously, all URIs that are not found at the end of a rebuild are eliminated from the index. A connection issue makes the crawling process stop promptly and eliminates all

documents from the index. Throwing a fatal error prevents this from happening, logs a message of Error type and stops the crawling process.

## Coveo.CES.CustomCrawlers.CrawlerStopRefreshException

This class is a special type of exception that should be thrown in specific situations during a RefreshUri operation. This operation is available if the IRefreshDocument interface is implemented and occurs in a separate thread than the normal refresh/rebuild or live indexing. If an error occurs that prevents the completion of the operation, throw this error to notify the Open Connector API that it can remove the thread.

## Coveo.CES.CustomCrawlers.CrawlingContext

Method	Description
AsCrawlerSecurityProvider	This method returns the object if it is a CrawlerSecurityProvider, null otherwise.
AsCrawlingContext	This method returns the object if it is a CrawlingContext, null otherwise.
LogCrawlerWarning	This is a shortcut method to log a message directly with Severity.Warning and Operation.Unspecified.
ExternalSecurityProviderName	When a security of External type is added to a document, specify the name of the security provider that is used. This value is in the source configuration and can be accessed through this property.
HasUserIdentity	With the new UserIdentity feature, this property returns True if a UserIdentity has been set on this source. When True, the UserName and UserPassword properties have a value.
UserName	This property returns the UserName parameter of a UserIdentity if one is set on the source being crawled.
UserPassword	This property returns the UserPassword parameter of a UserIdentity if one is set on the source being crawled.

## Coveo.CES.CustomCrawlers.CustomCrawler

Method	Description
CheckPoint	<p>This new method is important to create a good connector. It performs several operations that allow the connector to determine if it should keep going. The following lists the different operations it does:</p> <ul style="list-style-type: none"><li>• It validates if a state flag has changed;</li><li>• It indicates if the crawling should be stopped;</li><li>• It saves the state of the crawling if the connector implements the ISaveRestoreState interface.</li></ul> <p>A flag is raised when CES realizes that the crawling for this source should stop (for example if a user asks for a Stop operation or CES must close). This flag is validated in the CheckPoint method. It is frequently called internally, but if the connector has a long operation during which it does not call IndexDocument, CheckPoint should be called manually in order to ensure that the connector can save its state (if applicable) and exit.</p>
ClearAllPersistentFiles	Calling this method clears all the files associated with a source when the Open Connector API is used to persist data by using an interface or stringset.

CreateWorkingFolder	This method creates a unique folder for a source and returns the path in a string. This folder is deleted, if possible, during a shutdown. It is meant for temporary files.
GetActionForDocument	This method tests a document against the filters using the URI type specified in the Document.FilterBy property. Also, it tests the documenttype and filetype, and returns an Action to perform on the document.
GetThreadType	Returns the current ThreadType to identify what kind of thread is currently running, for example ThreadType.RefreshThread, or ThreadType.LiveIndexingThread. This can be useful if you plan on making a multithreaded connector.
IndexDocument	This method pushes a document to CES for indexing. The Open Connector API ensures the document has been modified or that its securities have been modified before performing the actual indexing operation.
IsLiveIndexingThread	This method returns True if the current thread is performing Live Indexing.
IsRefreshingThread	This method returns True if the current thread is performing a Rebuild/Refresh.
IsUriFiltered	This method tests an URI against the filters. If it is filtered, a message is logged concerning it. It is suggested to use GetActionForDocument over this method.
LogExceptionAsSystemError	This function is deprecated. It actually calls the Context.LogCrawlerErrorMessage method to log an entry, with some formatting applied to the exception's message.
PerformInitialize	This method is virtual, which means it only needs to be declared when it is required. If it is declared, it is the first method that the open connector calls.
PerformShutdown	This method is virtual, which means it only needs to be declared when it is required. If it is declared, it is called during shutdowns.
RegisterThread	This method is required if you plan on having multiple crawling threads. Register the ThreadType to reflect what operation is going on, for example Rebuild/Refresh or Live Indexing.
RemoveDocument	This method removes from the index the document at the specified URI. The document has to be in the current source.
RemoveDocumentAndChildren	This method removes from the index the document as well as any child of the specified URI. For example, use this method to remove folders. The URIs to delete have to be in the current source.
RemoveOlderThan	This method removes all documents older than the specified date from the current source.
ReportUnchangedDocument	This method tags a document as <i>Unchanged</i> ; therefore, when the rebuild finishes, it does not automatically remove the document. All documents that are not found (either indexed or tagged <i>Unchanged</i> ) are removed when the rebuild ends. This also logs a message specifying that the document has not changed.

ReportUriToIgnoreForDeletion	This method tags all the URIs and children in order for them not to be deleted at the end of a rebuild. This is useful if, for example, a failure in the system prevents the connector from finding these documents. You may not want them to be deleted at the end of the crawling process; therefore by calling this method on the URI, it prevents them from being removed. This method does not log anything.
Run	This abstract method is mandatory. It is called on a rebuild refresh.
Update	This method is virtual; therefore it is only required to declare it if it is necessary. It is called after PerformInitialize and whenever the connector is used and the source configuration was modified.
UpdateDocumentSecurity	This method updates the index with the new securities of the document.
Validate	This call is obsolete.
PersistentData	This property corresponds to the object that is persisted for this connector.
SaveStateInterval	This property returns the interval value between each call to ISaveRestoreState.SaveState. This value is incremented in the call to CheckPoint.
SourceConfig	This property returns the SourceConfig object for this source. It contains parameters for Live Indexing, the delay as well as the cushion.

### Coveo.CES.CustomCrawlers.CrawlerSecurityProvider

This class is a new interface that fits over the C interface to create external security providers. This allows the creation of a C# Security Provider. The basic idea is to inherit from the CrawlerSecurityProvider abstract class and override the methods required for your security provider.

### Coveo.CES.CustomCrawlers.Document

Method	Description
SetField	This new method allows to set the value of a specific system field. It takes strings and objects as parameters. The string is the name of the field (take advantage of the SystemFields class that has const strings for the name of the available fields). The object is the value of the field.
AddAttachment	This method allows to set an attachment relationship between two documents. The document in parameter is the child of the one whose method is called.
Dispose	This method has to be called before the destruction of the object to ensure that all resources allocated are correctly released. To make sure that happens, instantiate new Document objects with the using keyword.
GetFieldValue	This method returns the value of a field that was previously set.
GetMetaDataValue	This method returns the value of the metadata with the specified name.
IsOutOfDate	This method returns True if the modified date in the index is older than the one on the current document.
IsSecurityUpdated	This method returns True if the security of the current document is different than the one of the index.
SetField	This method allows to set the value on a systemfield. Use the SystemFields enum to identify the valid field names.

StreamData	This method is useful to handle big files. By getting a stream to the big file in parameter, it can be set to the document without having to load the file into memory and risking running out of it.
Attachments	This property returns an ArrayList containing this document's attachments.
Data	This property can be used to directly set the data of the document as an array of bytes. For arrays that could be big, it is suggested to use the StreamData method.
FilterBy	This property allows to set with which URI the document is tested against filters. The FilterByUri enum contains the valid options.
HasAllowedSecurity	This property returns True if at least one security can view the document.
IndexModifiedDate	This property is a user-friendly modified date. If it not set manually, this value is automatically set from the ModifiedDate field. This can be useful if a document changes, but its ModifiedDate value is not modified.
MetaData	This property is used to set the MetaData hashtable on the document.
UseCESCRCCheck	The index computes a CRC on the document. If this value is set to True, it is used to check if a document is modified or not.

### Coveo.CES.CustomCrawlers.DocumentMapping

Method	Description
IndexModifiedDate	Get and set accessors were added for the new m_IndexModifiedDate member. This member was added to the DocumentMapping class. It represents a modified date set by the index or the user when the ModifiedDate is not updated automatically by the system.

### Coveo.CES.CustomCrawlers.SourceConfig

This class represents the live indexing parameters for the source.

Method	Description
LiveIndexingCushion	This value corresponds to the cushion subtracted from the last LiveIndexing time to make sure no changes are missed.
LiveIndexingDelay	This value corresponds to the amount of time to wait between each Live Indexing pass.

## New Interfaces

### Coveo.CES.CustomCrawlers.IBeautifyUri

This interface allows you to implement a BeautifyUri method. This method is called when the source URIs are modified. Before being saved in the configuration, the method is called; therefore, the code can validate or edit what was entered in order to ensure the format is what the connector expects.

Method	Description
BeautifyUri	This method is called when the source URIs are modified.

## Coveo.CES.CustomCrawlers.IConfigValues

This interface is necessary for the CrawlingContext and CrawlerSecurityProvider classes. To create a new connector, it is of no use.

## Coveo.CES.CustomCrawlers.IDeleteDocument

This interface allows you to implement the DeleteDocument method, which removes a single document from the index from that source. It enables this option from the Administrator UI (a new button is displayed, letting someone delete one document).

Method	Description
DeleteDocument	This method takes a URI and removes it from the index if it is in this source.

## Coveo.CES.CustomCrawlers.IDeleteFolder

This interface allows you to implement the DeleteFolder method, which removes a URI and all of its children from the index from that source. The option becomes available in the Administrator UI, like IDeleteDocument.

Method	Description
DeleteFolder	This method takes a URI and removes it, as well as all of its children, from the index from that source.

## Coveo.CES.CustomCrawlers.IHasPersistentData

This interface allows you to implement the multiple methods required to synchronize persistent data with the CES 2 phases commit procedure. You should be well versed in the procedure of a 2 phases commit if you want to correctly implement this solution. To keep persistent data in a simpler way, the ISaveLoadPersistentData interface is a better solution.

Method	Description
ClosePersistentDataFile	This method is called when it is time to release everything that has to do with persistent data (for example, a shutdown).
LockData	This method is called when it is time to lock the persistent data, to ensure it is not modified until the call to UnlockData.
UnlockData	This method is called when the data can be unlocked.
PreCommitData	This method is called when the pre-commit phase starts.
CommitData	This method is called when the commit phase starts.
RollbackData	This method is called if a rollback has to be done.

## Coveo.CES.CustomCrawlers.ILiveIndexing

This interface allows you to implement the RunLiveIndexing method. It enables live indexing features for the connector. This creates a thread that calls RunLiveIndexing every time the delay is up. In a live indexing run, the modifications brought since the last time the method was called are displayed. This is extremely useful to keep the index up to date if the system crawled allows you to get this information quickly and efficiently.

Method	Description
RunLiveIndexing	This method is called every time the LiveIndexingDelay is over. The date and time on which the connector must start logging modifications is sent in parameter.

## Coveo.CES.CustomCrawlers.IRefreshDocument

This interface allows you to implement the RefreshDocument method. This enables the RefreshDocument UI in the Administrator UI in order to allow a single document to be refreshed.

Method	Description
RefreshDocument	This method refreshes the single URI that was passed in parameter.

### Coveo.CES.CustomCrawlers.IRefreshFolder

This interface allows you to implement the RefreshFolder method. This enables the RefreshFolder UI in the Administrator UI in order to allow a folder to be refreshed.

Method	Description
RefreshFolder	This method refreshes the URI and all of its children.

### Coveo.CES.CustomCrawlers.ISaveLoadPersistentData

This interface allows you to manage some persistent data necessary for the crawling process. Implement the LoadData and SaveData methods used in order to manage a single object that can be serialized. This object contains whatever persistent data to store.

Method	Description
LoadData	This method signals the connector that the PersistentData property on the CustomCrawler contains its object, ready to be read.
SaveData	This method returns the serializable object that needs to be persisted.

### Coveo.CES.CustomCrawlers.ISaveRestoreState

This interface allows to enable the Pause/Resume feature for the connector. Basically, this interface allows you to load, save and reset a serializable object that contains enough information to reposition the connector if its operation is interrupted.

Method	Description
ResetState	This method is called when it is time to reset the crawling state (on the start of a new operation or at the completion of one).
RestoreState	This method is called when the connector has to resume after it was interrupted. You will receive the last object saved in parameter, which must contain information to reposition yourself.
SaveState	This method is called periodically to ask for your object and save it in case the crawling is interrupted.

## Suggested Approach

Our suggested approach to fixing a connector developed for the CES4sp3 Open Connector API is to load the connector's project with the new API and compile it. Errors will be displayed and can then be validated by using this document.