Content Repository Connector for Adobe LiveCycle ES3

Content Repository Connector for LiveCycle is a Document Services Component (DSC) that permits LiveCycle to connect with Content Repository Extreme (CRX) 2.3, available with LiveCycle ES3, to be used as a repository. CRX 2.3 is available in the \CRX directory on the DVD installation media and Electronic Software Download (ESD).

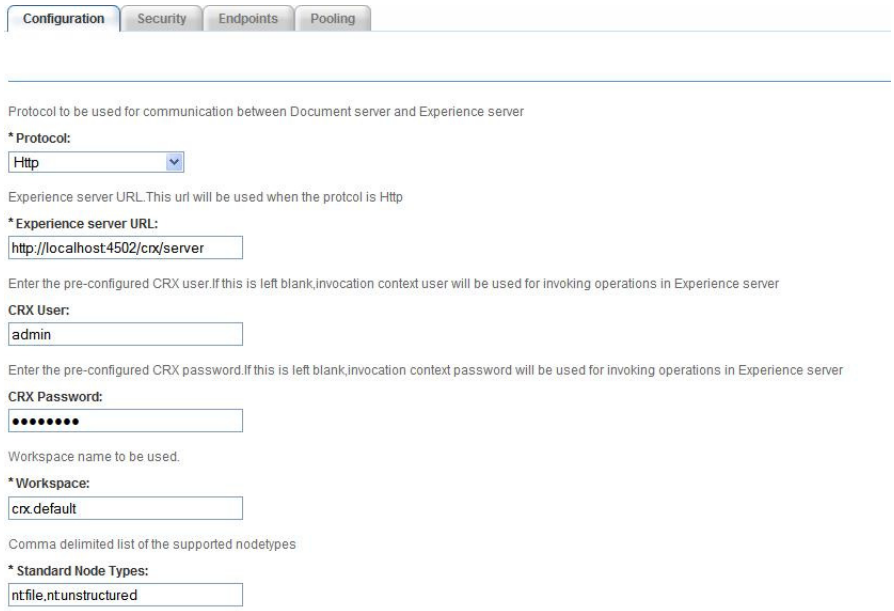
APPLIES TO

Adobe® LiveCycle® Enterprise Suite 3 (ES3)

CONTENTS

Configure Content Repository Connector

1. Log in to the Adobe LiveCycle Administration Console, and select **Home > Services > Applications and Services > Service Management**. Filter out **Foundation** services in the category and click **ContentRepositoryConnector: 1.0**.



Adobe

2. Select HTTP from the **Protocol** list. HTTP is the default protocol and there is no additional configuration required to use the HTTP protocol.

   However, ensure that you have installed standalone CRX (*com.adobe.granite.microcore.jar*) as described in Installing, Starting, and Stopping CRX. The *com.adobe.granite.microcore.jar* to install CRX 2.3 is available in the `\CRX` directory on the DVD installation media and Electronic Software Download (ESD).

   Caution: Installing CRX on an application server is currently not supported.

   Caution: Do not select JNDI from the protocol list as it is not a supported protocol.

3. In the Experience Server URL field, specify the URL to the running CRX instance. For example, http://<*server*>:<*port*>/crx/server.

4. In the CRX User and CRX Password fields, specify the CRX username and password for the CRX administrator to connect to the CRX repository. The default value for username/password is admin/admin.

5. In the **Workspace** field, specify a workspace name. By default, this points to the crx.default workspace.

6. In the **Standard Node Types** field, specify the node types to use in your DSC operations. Typically you will use nt:file and nt:unstructured, but any custom type that is derived from these types may be added to the list.

Operations available with the DSC

createFolder – Creates folders in the content repository. This operation returns the path of the folder created as a String. The folder that is created is of the type sling:OrderedFolder.

```
public String createFolder
    (String folderPath, String folderName, Boolean createParent) throws
    CRCAuthenticationException, CRCAccessDeniedException,
    CRCCommunicationException,CRCException
```

- folderPath – (Required) Specifies the path (including the new folder name and the Application-context) where the folder is to be created.
- folderName – (Required) Specifies the name of the folder to be stored.
- createParent – (Optional) Creates intermediate nodes if the value is true. Default is false.

storeContent – Stores different content-types in the content repository. Returns an instance of NodeInfo, which contains the details of the stored document. This operation cannot overwrite a node when provided with existing version label, and throws an exception if provided with an existing version label.

```
public String storeContent
    (String parentFolderPath, String nodeName, String nodeType,
       Document nodeContent, String versionLabel,
       List<String> mixin-types,
       Map<String, Serializable> propertiesMap) throws
    CRCAuthenticationException, CRCAccessDeniedException,
    CRCCommunicationException, CRCException
```

- parentFolderPath – (Required) Specifies the complete path to the parent folder in which the content is stored. Include the App-context as part of folder-path.

- nodeName – (Required) Specifies the name of the content to be persisted.
- nodeType – (Required) Specifies the content type for the content to be persisted. From the primary node-types, nt:file will be used by default. If custom data-types are registered, they should extend from nt:file. The following types will be shown in the property editor:
  - nt: unstructured
  - nt:file
  - Other node types mentioned in DSC configuration Standard Node Types
- nodeContent – (Required) Specifies the content in com.adobe.idp.Document format. If the type mentioned is nt:file, the node content is stored as a jcr:content node. Otherwise, a property jcr:data is created for other types and content be stored.
- versionLabel – (Optional) Provides a version label to the content. The version label cannot be an empty string. If the version label is set and the node is not versionable, this value is ignored. The version information may be obtained from the getVersionHistory operation.
- mixin-types – (Optional) Specifies a set of mixin-types that is applied on the node. The mixin-types specified are checked against the list of registered mixin-types before applying. If mix:versionable is applied or present, a new version is created, otherwise existing content is overwritten.
- propertiesMap – (Optional) Specifies a map containing key-value pairs for various properties. If the node-type specified supports additional properties, the properties are applied directly. Otherwise, the properties that applied on a node must be a part of node-type definition or a part of mixin-types applied on the node. In this instance, the properties mentioned have to be a subset of this exhaustive set. Multi-valued properties are returned as a Serializable array.

retrieveContent - Retrieves the content from a specific folder from the repository.

```
public NodeInfo retrieveContent
     (String nodePath, String versionLabel, String cutPoints) throws
     CRCAccessDeniedException, CRCAuthenticationException,
     CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- nodePath - (Required) Specifies the complete path for the node (including the node name) to be retrieved.
- versionLabel – (Optional) Provides a version label to the content. The version label cannot be an empty string. If the version label is set and the node is not versionable, this value is ignored. The version information may be obtained from the getVersionHistory operation.
- cutPoints - (Optional) Filters the properties of the content to be retrieved with the content. This is provided as a | (pipe) separated string, such as jcr:title|jcr:createdBy. If the cutpoints are set to blank, * or null, then complete properties of the node are retrieved along with the content of the node.

setProperties - Set a set of mixin-types and properties on a given node in the content repository.

```
public void setProperties
    (String nodePath, List<String> mixin-types,
        Map<String, Serializable> propertiesMap) throws
     CRCAccessDeniedException, CRCAuthenticationException,
     CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- nodePath – (Required) Specifies the complete path to the node.
- mixin-types – (Optional) Specifies a set of mixin-types that can be applied on the node. All the mixin-types mentioned are checked against the list of registered mixin-types before applying. If

mix:versionable is applied or present, a new version is created, otherwise existing content is overwritten.

- propertiesMap – (Optional) Specifies a map containing key-value pairs for various properties. If the node-type specified supports additional properties, the properties are applied directly. Otherwise, the properties that applied on a node must be a part of node-type definition or a part of mixin-types applied on the node In this instance the properties mentioned have to be a subset of this exhaustive set.

getProperties - Gets the set of mixin-types and properties that are applied to a node in the content repository.

```
public NodeInfo getProperties
    (String nodePath, String versionLabel, String cutPoints) throws
     CRCAccessDeniedException, CRCAuthenticationException,
     CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- nodePath – (Required) Specifies the complete path to the node.
- versionLabel – (Required) Provides the version of the content whose properties have to be obtained using its version label.
- cutPoints - (Required) Filters the properties of the content to be retrieved with the content. This is provided as a | (pipe) separated string, such as jcr:title|jcr:createdBy. If set to blank, * or null, complete set of properties of the node are retrieved along with the content of the node.

copy - Copies content from one location to another in the repository and returns a string that is a unique identifier for the created content.

```
public String copy
    (String sourceDocumentPath, String targetContentPath,
       String targetName, Boolean overwrite) throws
     CRCAccessDeniedException, CRCAuthenticationException,
     CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- sourceDocumentPath – (Required) Specifies the fully qualified path to the source document.
- targetContentPath – (Required) Specifies the fully qualified path to the target document.
- targetName – (Optional) Specifies the name of copied document. If unspecified, the name is same as source document.
- overwrite – If true, the copy overwrites the contents with the same name. By default, the copy does not overwrite the contents of the file.

move - Moves content from one location to another in the repository and returns a string that is a unique identifier for the created content. You cannot move content from the same source to the same target.

```
public String move
    (String sourceDocumentPath, String targetContentPath,
       String targetName, Boolean overwrite) throws
     CRCAccessDeniedException, CRCAuthenticationException,
     CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- sourceDocumentPath – (Required) Specifies the fully qualified path to the source document.
- targetContentPath – (Required) Specifies the fully qualified path to the target document.
- targetName – (Optional) Specifies the name of moved document. If unspecified, the name is same as source document.

- overwrite – If true, the copy overwrites of the contents with the same name. By default, the copy does not overwrite the contents of the file.

delete - Removes content from the repository. If the resource is a folder, all artifacts within the folder are deleted.

```
public void delete(String nodePath) throws
    CRXAccessDeniedException, CRXAuthenticationException,
    CRXCommunicationException, CRXInvalidParameterException, CRXException
```

- nodePath – (Required) Specifies the complete path to the node.

removeMixin - Removes the mixin-type on a given node.

```
public void removeMixin(String nodePath, List<String> mixinTypes) throws
    CRXAccessDeniedException, CRXAuthenticationException,
    CRXCommunicationException, CRXInvalidParameterException, CRXException
```

- nodePath – (Required) Specifies the complete path to the node.
- mixinTypes – (Required) Specifies the list of mixin-types to be removed from the node.

retrieveFolderContents - Retrieves all the artifacts within a folder. All files and folders are returned in a List of NodeInfo.

```
public List<NodeInfo> retrieveFolderContents
    (String folderPath, Boolean getOnlyFiles,
       Boolean includeFileContent, String cutpoints) throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- folderPath – (Required) Specifies the fully qualified path of the folder from which contents are retrieved.
- getOnlyFiles – If true, only files are retrieved, otherwise both files and folders are retrieved.
- includeFileContent – If true, the contents of the files are also retrieved.
- cutPoints – (Optional) Filters the properties to be retrieved. This is provided as a | (pipe) separated string such as jcr:title|jcr:createdBy.

checkIn - Checks in a content node in the repository. You cannot perform any operations on a checked-in node, except for a restore operation. For more details, see http://www.day.com/specs/jcr/2.0/15_Versioning.html

```
public void checkIn
    (String nodePath, String versionLabel, Boolean keepCheckedOut) throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- nodePath - (Required) Specifies the complete path for the node (including the node name) to be checked in.
- versionLabel – (Required) Specifies the unique label for the version to be checked in. You cannot check in a node with a version label that exists.
- keepCheckedOut - Allows persisting existing changes as a version and start working on new version without releasing the lock. This is equivalent to experience services checkpoint operation.

checkOut - Checks out a content node in the repository. If the node is not versionable or is already checked out, an exception is thrown.

```
public void checkOut(String nodePath)throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException, CRCInvalidParameterException, CRCException
```

- nodePath - (Required) Specifies the complete path for the node (including the node name).

getVersionHistory - Returns the version details for each node in the repository. When details for a specific version are obtained, its content can be obtained using the retrieveContent operation.

```
public List<NodeVersion> getVersionHistory(String nodePath)throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException,CRCInvalidParameterException, CRCException
```

- nodePath - (Required) Specifies the complete path for the node (including the node name).

restore - Restores the node in the current workspace to a previous version.

```
public void restore
    (String nodePath, String versionLabel, Boolean removeExisting) throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException,CRCInvalidParameterException, CRCException
```

- nodePath - (Required) Specifies the complete path for the node (including the node name).
- versionLabel – (Optional) Provides a version label to the content. The version label cannot be an empty string. If the version label is set and the node is not versionable, this value is ignored. The version information may be obtained from the getVersionHistory operation.
- removeExisting - (Required) Controls restore behavior in case the version to be restored has an existing, referential child node that may be present in current repository.

query - Queries the repository for any specific information. The output is a list of NodeInfo objects corresponding to the result. The properties populated with NodeInfo are reflected with a version label that exists.

```
public List<NodeInfo> query
    (String queryString, Long maxSize, Long offSet,
       Boolean includeFileContent, String cutPoints) throws
    CRCAccessDeniedException, CRCAuthenticationException,
    CRCCommunicationException, CRCException
```

- queryString – (Required) Specifies the query to be executed in SQL2 format.
- maxSize – (Optional) Specifies the maximum size for the values to be returned.
- offset - (Optional) The offset for the result.
- includeFileContent – If true, the contents of the files are also retrieved.
- cutPoints – (Optional) Filters the properties of the content to be retrieved with the content. This is provided as a | (pipe) separated string, such as jcr:title|jcr:createdBy. If set to blank, * or null, complete set of properties of the node are retrieved along with the content of the node.

Sample query strings in SQL2

This query searches for all file nodes whose title is firstTitle.

```
SELECT * FROM [nt:file] as selector_1 WHERE
selector_1.[jcr:title]='firstTitle'
```

This query searches for unstructured nodes in the path /test.

```
SELECT selector_1.* FROM [nt:unstructured] AS selector_1  WHERE
ISDESCENDANTNODE(selector_1, [/test])
```

This query performs a full text search of all files in the /test path for the term LiveCycle.

```
SELECT selector_1.* FROM [nt:file] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) AND
CONTAINS(selector_1.*, 'LiveCycle')
```

This query is similar to the full text search above, but uses a wild character search. The ? character searches for one character and * character searches for one or more occurrences. This query searches for file type nodes having terms such as Search or Scorch in their file content.

```
SELECT selector_1.* FROM [nt:file] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) AND
CONTAINS(selector_1.*, 'S??r*ch')
```

This query search for unstructured nodes and orders the results based on the title.

```
SELECT selector_1.* FROM [nt:unstructured] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) ORDER BY
selector_1.[jcr:title]
```

This query uses the OR operator to search unstructured nodes in the /test path or those that were created by admin.

```
SELECT selector_1.* FROM [nt:unstructured] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) OR
selector_1.[jcr:createdBy] = 'admin'
```

This query uses the AND and LIKE operators to search for unstructured nodes inside /test path which have something similar to firstTitle or tirstFitle where the % character represents a single character.

```
SELECT selector_1.* FROM [nt:unstructured] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) AND
selector_1.[jcr:title] LIKE '%irst%itle'
```

This query searches for unstructured nodes inside /test path whose title is not null.

```
SELECT selector_1.* FROM [nt:unstructured] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) AND
selector_1.[jcr:title] IS NOT NULL
```

This query searches for files inside /test path whose title is null.

```
SELECT selector_1.* FROM [nt:file] AS selector_1 WHERE
ISDESCENDANTNODE(selector_1, [/test]) AND
(NOT selector_1.[jcr:title] IS NOT NULL)
```

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

April 2012