How To Use Utilities

An Open Source Asset for use with TIBCO® Data Virtualization

Project Name	AS Assets Utilities
Document Location	This document is only valid on the day it was printed. The source of the document will be found in the ASAssets_Utilities folder (https://github.com/TIBCOSoftware)
Purpose	Self-paced instructional

Revision History

Version	Date	Author	Comments
1.0	08/06/2010	Mike Tinius, Calvin Goodrich, Gordon Rose	Initial revision
1.1	08/11/2010	Calvin Goodrich	Added documentation for string, time, and repository functions. Added initial guidelines for submitting new procedures.
1.2	08/19/2010	Mike Tinius	Added additional string, time and repository procedures.
1.3	09/30/2010	Mike Tinius	Added getNodeFromXML
1.4	10/11/2010	Calvin Goodrich	Added time and repository procedures.
1.5	10/13/2010	Gordon Rose	Added Active Directory and encoding procedures.
2010.3	10/18/2010	Calvin Goodrich	Finalized for 2010 Q3 official release
2010.4	1/14/2011	Calvin Goodrich	Added release notes section. Please detail updates to the Utilities here.
2011.2	4/1/2011	Calvin Goodrich	Updated for release 2011.2
2011.3	7/1/2011	Calvin Goodrich	Updated for release 2011.3
2011.4	10/1/2011	Calvin Goodrich	Updated for release 2011.4
2012.1	1/4/2012	Calvin Goodrich	Updated for release 2012.1
2012.2	4/5/2012	Calvin Goodrich	Updated for release 2012.2
2012.3	8/7/2012	Calvin Goodrich	Updated for release 2012.3
2012.4	11/1/2012	Calvin Goodrich	Updated for release 2012.4
2012.401	11/12/2012	Mike Tinius	Updated for release 2012.401
2012.402	11/20/2012	Mike Tinius	Updated for release 2012.402
2013.1	2/7/2013	Calvin Goodrich	Updated for release 2013.1
2013.2	5/1/2013	Calvin Goodrich	Updated for release 2013.2
2013.3	8/16/2013	Calvin Goodrich	Updated for release 2013.3
2013.301	8/27/2013	Mike Tinius	Updated for release 2013.301
2013.4	11/12/2013	Calvin Goodrich	Updated for release 2013.4
2014.1	2/18/2014	Calvin Goodrich	Updated for release 2014.1
2014.2	5/12/2014	Calvin Goodrich	Updated for release 2014.2
2014.3	8/1/2014	Calvin Goodrich	Updated for release 2014.3
2014.4	11/11/2014	Calvin Goodrich	Updated for release 2014.4
2015.1	2/19/2015	Calvin Goodrich	Updated for release 2015.1
2015.2	5/1/2015	Calvin Goodrich	Updated for release 2015.2

2015.3	8/10/2015	Calvin Goodrich	Updated for release 2015.3
2015.4	11/5/2015	Calvin Goodrich	Updated for release 2015.4
2016.1	1/29/2016	Calvin Goodrich	Updated for release 2016.1
2017.2	5/1/2017	Mike Tinius	Updated for release 2017.2
2017.4	12/13/2017	Mike Tinius	Transitioned to Tibco.
2018.1	3/21/2018	Mike Tinius	Updated for release 2018.1

Related Documents

Name	Version

Supported Versions

Name	Version
TIBCO® Data Virtualization Server	7.0 or later

Third Party Software Licenses

Product Name	Product Description and download site	License Type
jTidy	http://jtidy.sourceforge.net/	MIT Open Source
Apache Commons Net	http://commons.apache.org/proper/commons-net/	Apache
jaxen		The Werken Company
jdom	http://www.jdom.org/	Jason Hunter & Brett McLaughlin
xerces	http://www.apache.org/	Apache

Table of Contents

1 Release Notes for Version 2018 Q1

Introduction

This section provides release notes for the latest release of the Utilities. Please review the Deprecated Resources section carefully for resources that have been removed or will be removed from the Utilities distribution soon.

Regression Test Versions

The following list contains the version/patch/hotfix levels used for each regression test:

`7.0.5.00.04`

New Resources

`environment/getEnvName` – This procedure returns the environment name for this specific server. It is useful when sending emails to be able to identify the environment type in the subject line giving the recipient a quick way of determining which environment is having issues. It may also be used by procedural logic to perform different logic based on the environment type. It is intended to be modified for each environment. For a cluster it will have the same value by default because the DV code is duplicated on each server. The value can be whatever the DV development team wants up to 255 characters.

`generate/generateViews` – This script is used to provide a framework for introspection of a relational datasource and generating views to the various Data Abstraction Best Practices layers. It provides a simple view generation into the four main layers described by the Data Abstraction Best Practices. The caller has flexibility in terms of deciding which layers to use and the path of those layers. This is a simplified version of the open source Data Abstraction Best Practices in that it does not allow a formatting layer logical names to be derived from a spreadsheet.

`generate/destroyDependentLineage` – This procedure recursively destroys all of the dependent resources for a given starting folder (container) or table resource. A cursor of metadata is returned that shows which resources were destroyed. The general use case is to destroy a metadata, data source, schema folder or table which in turn destroys their dependent lineage from bottom to top with respect to the data abstraction layers.

`generate/destroyUsedLineage` – This procedure recursively destroys the “used” resources for a given starting folder (container) or view resource. The general use case is to destroy a published folder or view which in turn destroys their used lineage from top to bottom with respect to the data abstraction layers. The resource may be a published resource or a shared folder resource. The valid resource types are CONTAINER (folder), LINK or TABLE. Published procedures are not supported and will be bypassed. If the target resource is a folder then all resources within the folder and their “used” lineage are destroyed. If the target resource is a LINK or TABLE then only the lineage for that resource is destroyed.

`generate/helpers/createResourceProcess` – The is a helper procedure which is used to manage the creation of the folder and views within the Data Abstraction Best Practices layers. It is invoked by `generateViews`.

`examples/generate` – A series of examples that demonstrate the usage of `generateViews` and `destroyUsedLineage`. The _readme procedure provides details instructions on how to use and execute.

`repository/getDataSourceRootPath` – This procedure returns the file root/url path for a given file datasource path. The root/url path is the actual file system path when no file system security is being used. When file system security is in place, it is the root name mapping.

`repository/getDependentResourcesDirectCursor` – This procedure retrieves the "direct" dependent resource metadata for a given resource. If a dependent resource is a foreign key reference to another view it is not returned. If a dependent resource is a data source reference, it is not returned. A cursor of metadata is returned.

`repository/getDependentResourcesDirectRecurseCursor` – This procedure recursively walks the "direct" descendent tree to discover resource lineage. It only returns "direct" descendants and not foreign key descendants and not cache related descendants.

`repository/getResourceLineageDirectRecursive` – This procedure recursively walks the "direct" descendent tree to discover resource lineage. It only returns "direct" descendants and not foreign key descendants and not cache related descendants.

`repository/getUsedResourcesDirectCursor` – This procedure retrieves a cursor of metadata describing what resources are "directly" "used" by the resource path provided. It only returns "direct" descendants and not foreign key descendants or cache table or data source references. The full resource path and resource type must be provided.

`repository/getUsedResourcesDirectRecurseCursor` – This procedure recursively retrieves a cursor of metadata describing what resources are "directly" "used" by the resource path provided. It only returns "direct" descendants and not foreign key descendants or cache table or data source references.

`repository/introspectResources` – This script is used to provide a consistent and generic interface for introspecting database tables. It creates a necessary transaction around introspectResourcesTask and introspectResourcesResult.

`repository/updateImpactedResource` – This procedure is used to update a single impacted resource. The main objective is to work around issues in CIS that are fixable by simply opening, modifying and saving a resource. Typical issues that this can repair are views or procedures that are impacted after import into CIS. Typical error messages might be "session may not be null" or "session is closed". This procedure will not attempt to fix views or procedures that contain an impact level of "SYNTAX_ERROR". Moved from Best Practices to Utilities.

`repository/updateResourceOwner` – This script is used to change resource ownership of a resource. This script can be used to change ownership of a single resource or a folder with multiple resource. By default, the ownership is pushed recursively to all the resources in a folder.

`string/extractDelimitedText` – This is a generic procedure for extracting values from a list using a single or multi-character delimiter. The results are returned as a cursor of objects. This does not use the expensive REGEX function for parsing the delimited text.

`time/getCurrentTimestamp` – Get the current timestamp. This is a workaround for a bug introduced in 7.0.3 where CURRENT_TIMESTAMP did not work properly within the context of execution in a procedure.

Updated Resources

`getUtilitiesVersion` – Modified with the latest version for this release.

`archive/importArchiveFile`– Added additional input parameters to allow setting various input import archive options including: excludeResources, relocateResources, rebindResources, rebindUsers, remapAttributes and importOptions.

`repository/copyResourceAnnotations` – Modified so that column annotations are not modified on a published resource as this is not valid. This procedure copies the annotations from one resource to another. If both resources are of type "TABLE", the column annotations are copied as well (where the column names are the same, ignoring case.) If the column name matches (case insensitive) and the source annotation has a value, set the destination annotation. otherwise destination annotation is preserved.

`repository/createDataSource` – Modified so that the invoker can pass in the full data source path without the resource name.

`repository/getBasicResourceCursor` – Modified the output cursor to include additional parameters. This procedure is highly used within the ASAssets but there was no impact from this change. The following output cursor parameters were add: version VARCHAR(255), introspectState VARCHAR(255),impactMessage VARCHAR(32768), childCount INTEGER, dataSourceType VARCHAR(255).

`repository/lowerLevelProcedures/getBasicResourceXSLT` – Modified the XSLT transformation and output cursor to include additional parameters as stated in the above “getBasicResourceCursor” description.

`examples/repository/test_getResourceResponseXML` – Modified the output cursor to include additional parameters as stated in the above "getBasicResourceCursor" description.

`repository/getBasicResourceCursor_PROCEDURE_CURSOR` – Increased size of resourcePath to VARCHAR(4096) and columnType to VARCHAR(4096)..

`repository/lowerLevelProcedures/introspectResourcesResult` – **Added a new parameter – dsPath and moved to** `lowerLevelProcedures`. This procedure gathers the results from a call to introspectResourcesTask. If the introspection task is still running, the procedure can be called in such a way as to block execution until the task completes before returning results. **It is recommended that any existing applications using this procedure be modified to use the newly added introspectResources or otherwise modify their application to include the dsPath parameter**.

`repository/lowerLevelProcedures/introspectResourcesResultXSLT` – Modified to perform XSLT transformation on the response for getMostRecentIntrospectionStatus. This procedure transforms the results from getMostRecentIntrospectionStatus. The result output will be multiple rows if the introspection is successful. Otherwise, the additional result rows will indicate what went wrong.

`repository/lowerLevelProcedures/introspectResourcesTask` – Moved to `lowerLevelProcedures`.

Deprecated Resources

2018 Q1

`time/DateUtils/TZConverter` – This function is a duplicate of the DV internal function TZCONVERTOR and therefore is no longer needed. Users should convert over to using the DV internal function. This function will be removed in a future release.

`repository/getBasicResourceCursor_All` – This function is a duplicate of "getBasicResourceCursor" and is outdated as a result of the improvements made to "getBasicResourceCursor". Use "getBasicResourceCursor" instead of this procedure. This will be removed in a future release.

`repository/renameResource` – This function has been implemented in DV itself (`/lib/resource/RenameResource`). This has been removed.

`repository/resourceExists` – This function has been implemented in DV itself (`/lib/resource/ResourceExists`). This has been removed.

2015 Q3

`encoding/EncodingCJP/MD5Hash` – This function has been implemented in CIS itself (`HASHMD5`). This will be removed in a future release.

`encoding/EncodingCJP/SHA1Hash` – This function has been implemented in CIS itself (`HASHSHA1`). This will be removed in a future release.

2015 Q2

`repository/renameResource` – This function has been implemented in CIS itself (`/lib/resource/RenameResource`). This will be removed in a future release.

`repository/resourceExists` – This function has been implemented in CIS itself (`/lib/resource/ResourceExists`). This will be removed in a future release.

2014 Q4

`documentation/getDocConstants` – Replaced with `string/getConstants()`. This has been removed.

`log/errorNotification` – Replaced with `log/auditLogger()`. This has been removed.

`request/DUAL` – This view has been implemented in CIS itself. The view can be found at /services/databases/system/DUAL. This has been removed.

2014 Q3

`repository/RepoUtils/EncryptPassword` – Publishing the source code for this CJP exposes the CIS internals of how passwords are encrypted. This has been removed.

2014 Q1

`repository/RepoUtils62` – This CJP data source's procedures have been folded into `repository/RepoUtils`. This data source has been removed.

2012 Q4

`repository/addRemoveDataSourceChildren` – This function uses the deprecated introspection API (which now appears to be broken in 6.2 SP1.) Please use the new `repository/introspectResourcesTask()` and `repository/introspectResourcesResult()` utilities instead. This has been removed.

`repository/getResourceLineageParent` – This function is no longer being used by the documentation procedure and has been subsequently replaced by repository/getResourceLineageRecursive. This procedure has been removed.

`repository/lowerLevelProcedures/getResourceLineageRecursive` – This function is no longer being used by the documentation procedure and has been subsequently replaced by repository/getResourceLineageRecursive. This procedure has been removed.

`documentation/helpers/findDatabases` – This procedure is no longer being used by the documentation procedures and has been removed.

2012 Q1

`string/LPAD` – This function has been implemented in CIS itself. This has been removed.

`string/RPAD` – This function has been implemented in CIS itself. This has been removed.

`xml/CreateXmlString2CursorXForm` – The `xml/reverseXML` function does a MUCH better job and is MUCH easier to use. This has been removed.

2011 Q3

`repository/applyReservedListToPath` – This has been rewritten as a CJP in `repository/RepoUtils`. This will be removed in a future release.

`repository/applyReservedListToWord` – This has been rewritten as a CJP in `repository/RepoUtils`. This will be removed in a future release.

`repository/configureReservedList` – This will be removed in a future release.

2 Introduction

Purpose

The purpose of this document is to provide guidance on how to use the consolidated custom "Utilities" library.

This document provides documentation on the following functions:

1. **Active Directory** – functions for working with Active Directory.
2. **Archive** – functions for creating backup and package exports.
3. **Calculations** – general calculations.
4. **Conversions** – general conversions.
5. **Documentation** – tools for documenting resources.
6. **Encoding** – encoding conversions.
7. **Environment** – environment values.
8. **Examples** – example resources that illustrate the usage of many of the utilities.
9. **File** – full Create, Read, Update, and Delete for files plus many other useful file capabilities.
10. **Generate** – view generation and deletion scripts for data abstraction best practices layers.
11. **Logging** – general-purpose logging and error notification.
12. **Net** – tools for accessing the network.
13. **PDTool** – tools for generating PDTool deployment plans.
14. **Repository** – general-purpose repository API interaction functions.
15. **Request** – functions for accessing request source code.
16. **Security** – functions for encrypting and decrypting text.
17. **String** – general-purpose string manipulation functions.
18. **Time** – general-purpose time manipulation functions.
19. **Upgrade** – views and procedures that assist with major CIS upgrades.
20. **XML** – general-purpose XML manipulation functions.

History

Over the years, a number of development resources and utility functions have been developed by members of the professional services team, the sales engineering team, and various other technical folks within the former company Composite. In an effort to consolidate these utility procedures and prevent the “reinvention of the wheel”, a small team was formed to collect these highly useful and timesaving procedures into a single distribution and make them open source.

The utilities presented in this distribution are not full implementations or solutions to a particular problem. They are simple tools for accomplishing administrative/development tasks or tasks that are slightly outside of the designed use of Data Virtualization (DV) and not likely to ever be rolled into the DV product itself.

Audience

This document is intended to provide guidance for the following users:

- **Developers**
- **Administrators**

Installation Notes

New Folder Structure

Many Advanced Services (AS) assets are being consolidated under a single folder, /shared/ASAssets. From this point forward, the Utilities will be distributed in a CAR file that expects this structure.

To facilitate the management of the Utilities and other Advanced Services (AS) assets moving forward, the following are some guidelines for fresh and existing installations:

For **FRESH** installs of the Utilities where no other AS assets have been installed yet:

- Create a folder in /shared called “ASAssets”. Spelling and capitalization are important here so please use this exact spelling and case.
- Right click on the new ASAssets folder and select “Import …”, choose the Utilities distribution CAR file in the resulting dialog, and click the “Import>” button.

For **EXISTING** installs of the Utilities where an ASAssets folder **IS** desired:

- Create a folder in /shared called “ASAssets”. Spelling and capitalization are important here so please use this exact spelling and case.
- Edit the following procedures by changing the PATH statement (after the initial BEGIN keyword) from “/shared/Utilities” to “/shared/ASAssets/Utilities”. This will prevent issues with impacted resources after the cut and paste operation below.
    - /shared/ASAssets/Utilities/repository/definitions/RepositoryDefinitions
    - /shared/ASAssets/Utilities/repository/definitions/RepositoryDefinitionsRecursive
- Cut the existing Utilities folder from /shared and paste it into ASAssets. Cut and paste will ensure that any resources using the Utilities will be rebound to use the new location (this

does not rebind references embedded in character string values, however.) **DO NOT COPY** the Utilities folder into ASAssets as this will not rebind dependent resources.

- Right click on the new ASAssets folder and select "Import …", choose the Utilities distribution CAR file in the resulting dialog, check the "Overwrite" checkbox, and click the "Import>" button.
- Using the "Overwrite" option should only overwrite the Utilities folder and leave everything else in ASAssets intact.
- Execute the procedure /shared/ASAssets/reintrospectCJPs to reintrospect all the CJP data sources in the Utilities distribution.

For **EXISTING** installs of the Utilities where an **ASAssets** folder already exists:

- Right click on the ASAssets folder and select "Import …", choose the Utilities distribution CAR file in the resulting dialog, check the "Overwrite" checkbox, and click the "Import>" button.
- Using the "Overwrite" option should only overwrite the Utilities folder and leave everything else in "ASAssets" intact.
- Execute the procedure /shared/ASAssets/reintrospectCJPs to reintrospect all the CJP data sources in the Utilities distribution.

For **EXISTING** installs of the Utilities where the former **PSAssets** folder already exists:

- Rename PSAssets to ASAssets
- Right click on the ASAssets folder and select "Import …", choose the Utilities distribution CAR file in the resulting dialog, check the "Overwrite" checkbox, and click the "Import>" button.
- Using the "Overwrite" option should only overwrite the Utilities folder and leave everything else in "ASAssets" intact.
- Execute the procedure /shared/ASAssets/reintrospectCJPs to reintrospect all the CJP data sources in the Utilities distribution.

For **EXISTING** installs of the Utilities where no other AS assets have been installed yet and an ASAssets folder **IS NOT** desired:

- Right click on the /shared folder and select "Import …", choose the Utilities distribution CAR file in the resulting dialog, check the "Overwrite" checkbox, and click the "Import>" button.

- Using the “Overwrite” option should only overwrite the Utilities folder and leave everything else in “Shared” intact.
- After import, it may be necessary to manually edit some of the Utilities and correct paths that still point to an “ASAssets” folder.
- Execute the procedure /shared/ASAssets/reintrospectCJPs to reintrospect all the CJP data sources in the Utilities distribution.

Reserved Word List

The repository/RepoUtils/applyReservedWordTo* procedures use a flat properties file ($CIS_HOME/conf/customjars/RepoUtils.properties) to get the reserved word list. If the properties file is missing (usually when the Utilities are installed for the first time) it gets generated the first time one of these procedures is used. **It is never replaced** with an updated version under the assumption that a user might manually edit it to include new reserved words after an upgrade of CIS (without requiring an upgrade of the Utilities.)

Recursive Procedure Use

The AS Assets Utilities use a number of procedures that are recursive in nature (repeatedly call themselves until a condition is reached.) With that in mind, it's recommended that the "Maximum Request Depth" setting in CIS be updated from its default setting of 30 to 100. This setting can be found in the Studio's Configuration panel in "Server" > "Configuration" > "Transactions" > "Maximum Request Depth".

3 Top Level Utilities Procedures

Introduction

This section describes the procedures found directly under /shared/ASAssets/Utilities.

ExceptionDefinitions

Contains commonly used custom exceptions used throughout the /shared/ASAssets/Utilities folder.

getUtilitiesVersion (Custom Function)

Returns the current version of /shared/ASAssets/Utilities installed on the system. Can be used to enforce the minimum version of /shared/ASAssets/Utilities required for a script to function properly:

```
IF (getUtilitiesVersion() < 2017.4) THEN
    RAISE System.NotSupportedException
        VALUE '/shared/ASAssets/Utilities must be version 2017.4';
END IF;
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	DOUBLE

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	2016.1

reintrospectCJPs

This procedure walks the folder tree of the Utilities distribution and performs a reintrospection on any CJP data source that it finds. This is usually only needs to be executed once if/when a Utilities distribution is relocated within CIS. It expects that the Utilities are installed in /shared/ASAssets/Utilities. If not installed there, update the UTILITIES_HOME constant at the beginning of the procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	success	1

TypeDefinitions

Contains commonly used custom data types used throughout the /shared/ASAssets/Utilities folder.

4 How To Use ‘Active Directory’ Procedures

Introduction

This section will show how to use the Active Directory procedures.

ActiveDirectoryInt8ToDate (Custom Function)

Active Directory represents some Date values internally as an Integer8, which translates to a BIGINT. The number represents the number of 100-nanosecond intervals since 12:00 AM January 1, 1601. In addition, the fact that the CIS function UTC_TO_TIMESTAMP() uses 1/1/1970 as its base date must also be accounted for in the conversion.

Please note, Active Directory stores dates using Greenwich Mean Time (GMT), or a GMT offset of 0. The function supports converting the date value according to the number of hours that the local time zone is offset from GMT.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	ADInt8	BIGINT
IN	GMTOffsetInHours	INT
OUT	ADDate	DATE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	ADInt8	129296016000000000
IN	GMTOffsetInHours	6
OUT	ADDate	‘2010-09-21’

2.2. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	ADInt8	129296016000000000
IN	GMTOffsetInHours	0
OUT	ADDate	‘2010-09-22’

ActiveDirectoryTSToSQLTimeStamp (Custom Function)

Accept an Active Directory-formatted string as input and returns a Composite Timestamp. Supports a GMT offset in the same manner as `activedirectory/ActiveDirectoryInt8ToDate` (see previous function.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	TimeStampString	STRING
IN	GMTOffsetInHours	INT
OUT	TimeStampOut	TIMESTAMP

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	TimeStampString	'20100922123423'
IN	GMTOffsetInHours	0
OUT	TimeStampOut	'2010-09-22 12:34:23'

SimpleBinaryAND (Custom Function)

Accepts a BIGINT and a power of 2 (e.g. 2, 4, 8, etc.) and ANDs the two numbers to indicate whether the bit in the BIGINT at the position of the power of 2 is 1 or 0. It is a simple function originally created for determining whether or not an Active Directory user is enabled or disabled, which is stored in a bit mask in Active Directory. The function works for numbers 2 ^ 31 and smaller.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Num	BIGINT
IN	PowerOfTwo	BIGINT
OUT	Result	BIGINT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Num	3567

Direction	Parameter Name	Parameter Value
IN	PowerOfTwo	16
OUT	Result	0

5 How To Use ‘Archive’ Procedures

Introduction

This section will show how to use the ‘Archive’ procedures.

backup_export

This procedure performs a full server backup of the local CIS instance. It cannot be used to back up remote CIS instances.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	archiveFilePath	VARCHAR(4096)
OUT	success	BIT
OUT	responseXML	XML
OUT	faultXML	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	archiveFilePath	'C:\my_fsb.car'
OUT	success	1
OUT	responseXML	<xml …>
OUT	faultXML	NULL

importArchiveFile

This procedure performs an import of an archive CAR file in the Data Virtualization (DV) server. It gets the archive CAR file from the local DV file system where the DV server is running. This procedure uses two lower level procedures:

createImportArchiveXQuery – This procedure performs a transformation on the VARBINARY car file archive data into the request XML required for the procedure /services/webservices/system/admin/archive/operations/createImportArchive.

getArchiveFile – This procedure retrieves the car from the Data Virtualization server file system.

Examples of using this procedure can be found here: /shared/ASAssets/Utilities/examples/archive/test_importArchiveFile

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug – Y or N	CHAR(1)
IN	car_file_os_full_path – full path to the archive file on the DV server.	LONGVARCHAR
IN	excludeResources - optional - A list resources that should not be imported.	VECTOR(ROW(resourcePath LONGVARCHAR, resourceType VARCHAR))
IN	relocateResources - optional - A mapping of resources from their location in the archive to where they should be imported.	VECTOR(ROW(fromResourcePath LONGVARCHAR, fromResourceType VARCHAR, toResourcePath LONGVARCHAR, toResourceType VARCHAR))
IN	rebindResources - optional - A mapping of resources references within the archive to where they should refer to. All resources containing the fromResourcePath and fromResourceType are rebound to the toResourcePath and toResourceType.	VECTOR(ROW(fromResourcePath LONGVARCHAR, fromResourceType VARCHAR, toResourcePath LONGVARCHAR, toResourceType VARCHAR))
IN	rebindUsers - optional - A mapping of users within the archive to whom they should be. (change ownership). All resources with the current ownership in fromDomain and fromUser and changed to the owner toDomain and toUser.	VECTOR(ROW(fromDomain VARCHAR, fromUser VARCHAR, toDomain VARCHAR, toUser VARCHAR))
IN	remapAttributes - optional - A list of resource attribute settings that should be applied on import.	VECTOR(ROW(resourcePath LONGVARCHAR, resourceType VARCHAR, attributes VECTOR(/shared/ASAssets/Utilities/repository/definitions/RepositoryDefinitions.AttributeCompleteType)

Direction	Parameter Name	Parameter Type
		attrName VARCHAR(255),-- mandatory attrType VARCHAR(255),-- mandatory attrValue LONGVARCHAR,-- optional (1 of attrValue, valueList, valueMap or valueArray must be supplied) valueList VECTOR(itemType), -- optional "type" VARCHAR(255), "value" LONGVARCHAR valueMap VECTOR(entryType),-- optional "key" VECTOR(itemType),-- this vector always only contains 1 key "type" VARCHAR(255), "value" LONGVARCHAR "value" VECTOR(itemType)-- this vector always only contains 1 value "type" VARCHAR(255), "value" LONGVARCHAR valueArray VECTOR(LONGVARCHAR), --optional unset BIT-- optional - 1 is unset, 0 is set or null to ignore))
IN	importOptions - optional - A list of archive options indicating what additional features should be imported. By default, the same options used for export will be used for import.	VECTOR(LONGVARCHAR)
OUT	success	BIT
OUT	message	LONGVARCHAR

2. Examples:

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	car_file_os_full_path	'C:\my_fsb.car'
IN	excludeResources - optional - A list	VECTOR[

Direction	Parameter Name	Parameter Value
	resources that should not be imported.	('/shared/a/x2','TABLE'), ('/shared/a/x1','PROCEDURE')]
IN	relocateResources - optional - A mapping of resources from their location in the archive to where they should be imported.	VECTOR[('/shared/a/x3','TABLE', '/shared/a/f1/x3','TABLE')];
IN	rebindResources - optional - A mapping of resources references within the archive to where they should refer to. All resources containing the fromResourcePath and fromResourceType are rebound to the toResourcePath and toResourceType.	VECTOR[('/shared/a/x2','TABLE', '/shared/a/x3','TABLE'),];
IN	rebindUsers - optional - A mapping of users within the archive to whom they should be. (change ownership). All resources with the current ownership in fromDomain and fromUser and changed to the owner toDomain and toUser.	VECTOR[('composite','admin', 'composite','user1')];
IN	remapAttributes - optional - A list of resource attribute settings that should be applied on import.	VECTOR[('/shared/a/ds_orders','DATA_SOURCE', VECTOR[('url', 'STRING', 'jdbc:postresql://localhost:9408/orders', null,null,null,null)])];
IN	importOptions - optional - A list of archive options indicating what additional features should be imported. By default, the same options used for export will be used for import. OVERWRITE: Overwrite the existing resources if they exist. INCLUDE_CACHING: Include caching configurations for resources. INCLUDE_CUSTOM_JAVA_JARS: Include custom Java JARs in the export. (ADMIN ONLY) INCLUDE_STATISTICS: Include any resources statistics known about the table	VECTOR[('OVERWRITE'), ('INCLUDE_PHYSICAL_SOURCE_INFO'), ('INCLUDE_CACHING'), ('INCLUDE_SECURITY')];

Direction	Parameter Name	Parameter Value
	boundaries, and column boundaries. INCLUDE_DEPENDENCY: Gather and include all dependent resources for the resources you choose to export. INCLUDE_PHYSICAL_SOURCE_INFO: Include sensitive connection information for included physical sources. (OWNER ONLY) INCLUDE_REQUIRED_USERS: Include the information about the required users in the export file. INCLUDE_SECURITY: Include resource privilege settings. (OWNER ONLY)	
OUT	success	1
OUT	message	Successfully imported archive file.

6 How To Use ‘Calculation’ Procedures

Introduction

This section will show how to use the ‘Calculation’ procedures.

calculateAge (Custom Function)

This function is used to calculate a person’s age given their birthday timestamp and the current timestamp at the time of calculation.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	personBirthdayTimestamp	TIMESTAMP
OUT	age	INTEGER

2. **Examples:**

   2.1. **Assumptions:**

      2.1.1. **`CURRENT_TIMESTAMP` is used at the time of invocation. For example, the format would be something like this: ‘2010-07-27 10:30:00’**

Direction	Parameter Name	Parameter Value
IN	personBirthdayTimestamp	‘1990-01-01 00:00:00’
OUT	age	20

medianFromQuery (Custom Function)

This function calculates the median value from a single column query result. The query must be ordered for the median function to work properly (the query must include an ORDER BY clause or it will throw an exception.) If the number of rows in the result is odd, the function will return the middle value. If the number of rows in the result is even, the function will return the average of the two middle values. If there are no rows in the result, then the function will return NULL.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	query	/lib/util/System.Text (VARCHAR (2147483647))
OUT	result	DOUBLE

2. **Examples:**

Direction	**Parameter Name**	**Parameter Value**
IN	query	‘SELECT FreightCharge FROM /shared/examples/ds_orders/orders ORDER BY FreightCharge’
OUT	result	26.0

7 How To Use ‘Conversion’ Procedures

Introduction

This section will show how to use the ‘Conversion’ procedures.

convertBit (Custom Function)

Convert a string (T, F, 1, 0, Y, N, yes, no, true, or false) into a BIT response (1 or 0.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	request	VARCHAR(255)
OUT	response	BIT

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	request	‘Yes’
OUT	response	1

convertBoolean (Custom Function)

Convert a string (T, F, 1, 0, Y, N, yes, no, true, or false) into a Boolean response so that it makes it easier to test conditions.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	request	VARCHAR(255)
OUT	response	VARCHAR(255)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	request	‘Yes’
OUT	response	true

convertDoubleToInteger (Custom Function)

Convert a double into an integer so as to remove trailing '.00000…' values. It is useful for Oracle ID fields that were defined as NUMBER with no qualifying .0 decimal place. E.g. NUMBER(38.0).

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	d	DOUBLE
OUT	i	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	d	1650.000000000
OUT	i	1650

convertTemperatureUnit (Custom Function)

This procedure is used to convert passed in temperatures from one unit of measurement to another. Temperatures can be converted from / to degrees in Fahrenheit, Celsius and Kelvin.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputTemperature	FLOAT
IN	inputUnits	VARCHAR(1)
IN	targetUnits	VARCHAR(1)
OUT	convertedTemperature	FLOAT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputTemperature	98.6
IN	inputUnits	'F'
IN	targetUnits	'C'
OUT	convertedTemperature	37.002959999999995

convertYN (Custom Function)

Convert a string (T,F,1,0,Y,N,yes,no,true,false) into a Y or N response.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	request	VARCHAR(255)
OUT	response	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	request	‘0’
OUT	response	‘N’

8 How To Use ‘Documentation’ Procedures

Introduction

This section describes the routines using the “documentation” procedures.

getDocumentationDriver

This is the documentation driver procedure. It is used to generate documentation for composite resources and save to a file. The “/shared/ASAssets/Utilities/documentation/constants” procedure sets the defaults for a number of parameters. Because there is no output for this procedure it could be used as a trigger procedure if a customer wanted to generate documentation on a scheduled basis.

The best practice for the developer is to copy the constants() procedure to the project directory and modify the constants there. Pass in the location of that procedure to this driver. This allows you to customize the default values once.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**in_resourcePath** – The starting CIS path for which to introspect resources and generate documentation.	pathType
IN	**in_resourceType** – The starting CIS resource type for the resource path.	VARCHAR
IN	**in_filePath** – The full file system path to generate the documentation file to.	pathType
IN	**in_docPreambleImpl** – The CIS path to the procedure that represents the preamble of the documentation. If left null, the default “geDocPreambleImpl1” is used.	pathType
IN	**in_docResourceFormatImpl** – The CIS path to the procedure that performs the documentation formatting. If left null, the default “getDocResourceFormatImpl1” is used.	pathType
IN	**in_constantPath** – This is the path to the constants file	pathType
IN	**in_switches** – Provides guidance on what documentation to print. The format is print_switch=[option1|{default_option2}|option3] This is a space separate list with no spaces before or after the equal sign. Print_containers=[{none}|all] – print the resource container (folder) print_annotations=[none|{all}|nonblank|blank] – print all	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	annotations whether they are blank or not print_resource_projections=[none|{all}] – print the resource projections print_resources_used=[none|{all}] – print the immediate child resources used by the parent resource print_datasource_accessed=[none|{all}] – print the data source accessed list print_datasource_lineage=[none|{all}] – print the data source lineage print_time=[{no}|yes] – print the time it takes to retrieve the full documentation for each resource and the final time save_file=[{no}|yes] – save the results to a file save_file_intermediate=[{no}|yes] – save the file intermediately after each resource is completed Example: 1) switches: when left blank or null then the defaults are taken result: all documentation modules are printed 2) switches: print_annotations=nonblank print_resource_projections=none print_resources_used=none print_datasource_lineage=none result: only non-blank annotations are printed and nothing else	
IN	**in_excludeKeywordsInPathList** – Exclude keywords in path, case insensitive. Comma separated list. These are whole words and not wild cards. A word is defined by what exists between folder separators "/". i.e. 'analysis,archive'	LONGVARCHAR
IN	**in_excludePathsList** – Exclude actual paths. Double quotes are not required. Comma separated list. The exclude path list simply has to be present in any part of the resource path. This means that it can be a partial path.	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	in_resourcePath	/shared/examples

Direction	Parameter Name	Parameter Value
IN	in_resourceType	CONTAINER
IN	in_filePath	/temp/cis_resource_docs.txt
IN	in_docPreambleImpl	/shared/ASAssets/Utilities/documentation/implementations/geDocPreambleImpl1
IN	in_docResourceFormatImpl	/shared/ASAssets/Utilities/documentation/implementations/getDocResourceFormatImpl1
IN	in_constantPath	/shared/ASAssets/Utilities/documentation/constants()
IN	in_switches	print_containers=no print_time=yes save_file=yes save_file_intermediate=yes
IN	in_excludeKeywordsInPathList	analysis,archive,test,validation
IN	in_excludePathsList	/shared/ASAssets/Utilities,/shared/BestPractices,/lib

getAllDocumentationAPI

This procedure serves as an "API" procedure. It can be invoked by other application procedures to return the documentation. If the file path is left null, then it does not attempt to write to a file.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath** – The starting CIS path for which to introspect resources and generate documentation.	pathType
IN	**resourceType** – The starting CIS resource type for the resource path.	VARCHAR
IN	**docPreambleImpl** – The CIS path to the procedure that represents the preamble of the documentation. If left null, the default "geDocPreambleImpl1" is used.	pathType
IN	**docResourceFormatImpl** – The CIS path to the procedure that performs the documentation formatting. If left null, the default "getDocResourceFormatImpl1" is used.	pathType
IN	**constantPath** – This is the path to the constants file	pathType
IN	**switches** – Provides guidance on what documentation to print. The format is print_switch=[option1|{default_option2}|option3] This is a space separate list with no spaces before or after the equal sign. print_containers=[{none}|all] - print the resource	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	container (folder) print_annotations=[none|{all}|nonblank|blank] - print all annotations whether they are blank or not print_resource_projections=[none|{all}] - print the resource projections print_resources_used=[none|{all}] - print the immediate child resources used by the parent resource print_datasource_accessed=[none|{all}] - print the data source accessed list print_datasource_lineage=[none|{all}] - print the data source lineage print_time=[{no}|yes] - print the time it takes to retrieve the full documentation for each resource and the final time save_file=[{no}|yes] - save the results to a file save_file_intermediate=[{no}|yes] - save the file intermediately after each resource is completed Example: 3) switches: when left blank or null then the defaults are taken result: all documentation modules are printed 2) switches: print_annotations=nonblank print_resource_projections=none print_resources_used=none print_datasource_lineage=none result: only non-blank annotations are printed and nothing else	
IN	**excludeKeywordsInPathList** – Exclude keywords in path, case insensitive. Comma separated list. These are whole words and not wild cards. A word is defined by what exists between folder separators "/". i.e. 'analysis,archive'	LONGVARCHAR
IN	**excludePathsList** – Exclude actual paths. Double quotes are not required. Comma separated list. The exclude path list simply has to be present in any part of the resource path. This means that it can be a partial path.	LONGVARCHAR
IN	**filePath** – The full file system path to generate the documentation file to.	pathType

Direction	Parameter Name	Parameter Type
OUT	**formattedText** – The formatted text is returned for all documentation	/lib/util/System.Text

2. Examples:

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples
IN	resourceType	CONTAINER
IN	docPreambleImpl	/shared/ASAssets/Utilities/documentation/implementations/geDocPreambleImpl1
IN	docResourceFormatImpl	/shared/ASAssets/Utilities/documentation/implementations/getDocResourceFormatImpl1
IN	constantPath	/shared/myproject/documentation/constants
IN	switches	print_containers=no print_time=yes save_file=yes save_file_intermediate=yes
IN	excludeKeywordsInPathList	analysis,archive,test,validation
IN	excludePathsList	/shared/ASAssets/Utilities,/shared/BestPractices,/lib
IN	filePath	/temp/cis_resource_docs.txt
OUT	formattedText	See below:

```
Composite Software Documentation

Generated on 2012-11-11 08:10:34.614


=================================================================================
Resource Name: CompositeView
Resource Path: /shared/examples/CompositeView
Resource Type: TABLE
        SubType: SQL_TABLE

Description:
------------
None

Resource Column Projection:
---------------------------
   Column Name                Column Type         Native Base Type    Native Type
   ------------------------   --------------------   --------------------   --------------------
   OrderID                    INTEGER             N/A                 N/A
   ProductID                  INTEGER             N/A                 N/A
   Discount                   DOUBLE              N/A                 N/A
   OrderDate                  DATE                N/A                 N/A
   CompanyName                VARCHAR(50)         N/A                 N/A
```

```
  CustomerContactFirstName VARCHAR(30)       N/A                  N/A
  CustomerContactLastName VARCHAR(50)        N/A                  N/A
  CustomerContactPhone      VARCHAR(30)      N/A                  N/A
  ProductName               VARCHAR(32768)   N/A                  N/A
  TransactionID             INTEGER          N/A                  N/A
  DateRequired              DATE             N/A                  N/A
  DatePromised              DATE             N/A                  N/A
  ShipDate                  DATE             N/A                  N/A
  SupplierID                INTEGER          N/A                  N/A
  SupplierName              VARCHAR(50)      N/A                  N/A
  SupplierContactName       VARCHAR(50)      N/A                  N/A
  SupplierPhoneNumber       VARCHAR(30)      N/A                  N/A

Resources Used:
---------------
  Resource Name  Resource Type   Subtype       Resource Path
  -------------- --------------  ----------    -----------------------------
  ViewOrder      TABLE           SQL_TABLE     /shared/examples/ViewOrder
  ViewSales      TABLE           SQL_TABLE     /shared/examples/ViewSales
  ViewSupplier   TABLE           SQL_TABLE     /shared/examples/ViewSupplier

Data Source Accessed List:
---------------------------
  Datasource Name     Enabled    Type          Subtype                       Datasource Path
  ----------------------------- -------  -----------   ------------------------     -------------------------------------------
  ds_orders            1         DATA_SOURCE   RELATIONAL_DATA_SOURCE      /shared/examples/ds_orders
  ds_XML               1         DATA_SOURCE   XML_FILE_DATA_SOURCE        /shared/examples/ds_XML
  ds_inventory         1         DATA_SOURCE   RELATIONAL_DATA_SOURCE      /shared/examples/ds_inventory

Data Source Lineage:
--------------------
seqnum   id     pid    depth  resource path
   1 - 20587            0  /shared/examples/CompositeView

   2 - 20658  20587      1    /shared/examples/ViewOrder
   3 - 20741  20658      2      [CS] /shared/examples/ds_orders/customers  [TABLE.DATABASE_TABLE]
                          [DS] /shared/examples/ds_orders
   4 - 20679  20658      2      [CS] /shared/examples/ds_orders/orderdetails  [TABLE.DATABASE_TABLE]
                          [DS] /shared/examples/ds_orders
   5 - 20711  20658      2      [CS] /shared/examples/ds_orders/orders  [TABLE.DATABASE_TABLE]
                          [DS] /shared/examples/ds_orders
   6 - 20670  20711      3       /shared/examples/ds_orders
   7 - 20729  20670      4         [CS] /shared/examples/ds_orders/cache_status  [TABLE.DATABASE_TABLE]
                            [DS] /shared/examples/ds_orders
   8 - 20671  20670      4         [CS] /shared/examples/ds_orders/cache_tracking  [TABLE.DATABASE_TABLE]
                            [DS] /shared/examples/ds_orders
   9 - 20689  20711      3       [CS] /shared/examples/ds_orders/orders_cache  [TABLE.DATABASE_TABLE]
                           [DS] /shared/examples/ds_orders

  10 - 20774  20587      1    /shared/examples/ViewSales
  11 - 20679  20774      2      [CS] /shared/examples/ds_orders/orderdetails  [TABLE.DATABASE_TABLE]
                          [DS] /shared/examples/ds_orders
  12 - 20786  20774      2      /shared/examples/productCatalog_Transformation
  13 - 20757  20786      3       [CS] /shared/examples/ds_XML/productCatalog.xml  [TREE.XML_FILE_TREE]
                           [DS] /shared/examples/ds_XML

  14 - 20763  20587      1    /shared/examples/ViewSupplier
  15 - 20606  20763      2      [CS] /shared/examples/ds_inventory/inventorytransactions  [TABLE.DATABASE_TABLE]
                          [DS] /shared/examples/ds_inventory
  16 - 20619  20763     2     [CS] /shared/examples/ds_inventory/purchaseorders  [TABLE.DATABASE_TABLE]
                                [DS] /shared/examples/ds_inventory
  17 - 20644  20763     2     [CS] /shared/examples/ds_inventory/suppliers  [TABLE.DATABASE_TABLE]
                                [DS] /shared/examples/ds_inventory

Resource Documentation Generation Time=0 00:00:00.266
--------------------------------------------------------------------------------
...

================================================================================
---------------------
Documentation Summary
---------------------
Starting Root Path: /shared/examples
---------------------------------------------------------
Print Switches Input: save_file=yes save_file_intermediate=yes print_time=yes

print_containers=0              Key:[{none}=0|all=1]
print_annotations=1             Key:[none=0|{all}=1|nonblank=2|blank=3]
print_resource_projections=1  Key:[none=0|{all}=1]
print_resources_used=1          Key:[none=0|{all}=1]
print_datasource_accessed=1  Key:[none=0|{all}=1]
print_datasource_lineage=1    Key:[none=0|{all}=1]
print_time=1                    Key:[{no}=0|yes=1]
save_file=1                     Key:[{no}=0|yes=1]
save_file_intermediate=1        Key:[{no}=0|yes=1]

---------------------------------------------------------
Total Number of Resources:            23
---------------------------------------------------------
Number of Published    (LINK):                      0
Number of Folders      (CONTAINER):                 0
Number of Views        (TABLE):                     15
Number of Procedures   (PROCEDURE):                 3
Number of Data Sources (DATA_SOURCE):               3
Number of XML          (TREE):                      1
Number of Triggers     (TRIGGER):                   0
Number of Connectors   (CONNECTOR):                 0
Number of Def. Sets    (DEFINITION_SET):            1
Number of Other resource type:                      0
---------------------------------------------------------
Documentation Generation Time=0 00:00:01.7
--------------------------------------------------------------------------------
```

constants

These are default constants used by the documentation procedures.

The best practice for this procedure is to copy it and paste it into the project folder and configure the constant values as project specific rather than configuring /shared/ASAssets/Utilities/documentation/constants. If you were to upgrade the Utilities, you would lose the changes.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
CONSTANT	**docPreambleImpl** – Default location to the documentation preamble implementation procedure.	pathType
CONSTANT	**docResourceFormatImpl** – Default location to the resource formatting implementation procedure.	pathType
CONSTANT	**debug** – debug flag	CHAR(1)
CONSTANT	**debugTime** – first level of time display	CHAR(1)
CONSTANT	**debugTime2** – second level of time display for the resource vector loop	CHAR(1)
CONSTANT	**switches –** default switches	LONGVARCHAR
CONSTANT	**filePath** – the location for the output file	pathType
CONSTANT	**resourcePath** – default resource path to introspect	pathType
CONSTANT	**resourceType** – default resource type	VARCHAR
CONSTANT	**eol** – the end of line character	VARCHAR
CONSTANT	**indent2** – indent 2 spaces	VARCHAR
CONSTANT	**indent4** – indent 4 spaces	VARCHAR
CONSTANT	**padChar** – the padding characters for formatted printing	VARCHAR
CONSTANT	**beginSeparator** – the beginning separator for a resource grouping.	VARCHAR
CONSTANT	**endSeparator** – the ending separator for a resource grouping.	VARCHAR
CONSTANT	**minorSeparator** – the minor separator which may be used within a grouping.	VARCHAR
CONSTANT	**excludeKeywordsInPathList –** exclude keywords in path, case insensitive. Comma separated list. These are whole words and not wild cards.	LONGVARCHAR
CONSTANT	**excludePathsList –** exclude actual paths. Double quotes are not required. Comma separated list.	LONGVARCHAR
CONSTANT	**excludeDSPathsList** – exclude paths when finding matches for datasources. This is a comma separated list of paths to exclude from processing.	LONGVARCHAR

2. Examples:

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
CONSTANT	docPreambleImpl	/shared/ASAssets/Utilities/documentation/implementations/geDocPreambleImpl1

Direction	Parameter Name	Parameter Value
CONSTANT	docResourceFormatImpl	/shared/ASAssets/Utilities/documentation/implementations/getDocResourceFormatImpl1
CONSTANT	debug	Y or N
CONSTANT	debugTime	Y or N
CONSTANT	debugTime2	Y or N
CONSTANT	switches	
CONSTANT	filePath	/temp/cis_resource_docs.txt
CONSTANT	resourcePath	/shared
CONSTANT	resourceType	CONTAINER
CONSTANT	eol	CHR(13)
CONSTANT	indent2	‘ ‘
CONSTANT	indent4	‘ ‘
CONSTANT	padChar	‘ ‘
CONSTANT	beginSeparator	80 x ‘=’
CONSTANT	endSeparator	80 x ‘-‘
CONSTANT	minorSeparator	‘’
CONSTANT	excludeKeywordsInPathList	analysis,archive,test,validation
CONSTANT	excludePathsList	/shared/ASAssets/Utilities,/shared/BestPractices,/lib
CONSTANT	excludeDSPathsList	/shared/Common/COMPOSITE_CACHE

documentationTrigger

The documentation trigger provides a template for a developer to copy and configure to automatically wake up and generate the documentation for a project.

The best practice for this procedure is to copy it and paste it into the project folder and configure the parameters as project specific rather than configuring /shared/ASAssets/Utilities/documentation/documentationTrigger. If you were to upgrade the Utilities, you would lose the changes.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**Procedure Path** – The path to the documentation driver procedure.	pathType
IN	**Parameter Values** – refer to the procedure “getDocumentationDriver” for details on what to pass in.	VARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Procedure Path	/shared/ASAssets/Utilities/documentation/getDocumentationDriver
IN	Parameter Values	NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL, NULL Refer to the procedure "getDocumentationDriver" for details on what to pass in.

helpers

This section describes the auxiliary procedures for documentation.

helpers/getDocConstant (Custom Function)

This procedure gets a constant value from a dynamic constant path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**constantPath** – The path to the constants file ending with procedure parenthesis ().	pathType
IN	**constantName** – The name of the constant.	VARCHAR
OUT	**outValue** – The value of the constant	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
IN	constantName	switches
OUT	outValue	print_time=yes save_file=yes save_file_intermediate=yes

helpers/getDocCounts

This procedure is used to increment the counts for the various resource type counters.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	**resourceType** – The type of resource.	VARCHAR
INOUT	**numResources** – Total count for all resources.	ITNEGER
INOUT	**numContainers** – Count of containers (folders).	ITNEGER
INOUT	**numConnectors** – Count of connectors.	ITNEGER
INOUT	**numDefinitionSets** – Count of definition sets.	ITNEGER
INOUT	**numTriggers** – Count of triggers.	ITNEGER
INOUT	**numViews** – Count of views or database tables.	ITNEGER
INOUT	**numProcs** – Count of procedures.	ITNEGER
INOUT	**numTree** – Count of XML / Tree resources.	ITNEGER
INOUT	**numDatasources** – Count of data sources.	ITNEGER
INOUT	**numPublished** – Count of published (link) resources.	ITNEGER
INOUT	**numOtherType** – Count of other resources not covered above (catch-all).	ITNEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourceType	TABLE
INOUT	numResources	1
INOUT	numContainers	1
INOUT	numConnectors	1
INOUT	numDefinitionSets	1
INOUT	numTriggers	1
INOUT	numViews	1
INOUT	numProcs	1
INOUT	numTree	1
INOUT	numDatasources	1
INOUT	numPublished	1
INOUT	numOtherType	1

helpers/parseDocSwitches (Custom Function)

This procedure parses the switches that are passed in to determine what behavior should be taken.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**switches** – A space separated list of switches with no embedded spaces. Definitions: print_containers= [{none}|all] - print the resource container (folder) print_annotations= [none|{all}|nonblank|blank] - print all annotations whether they are blank or not print_resource_projections= [none|{all}] - print the resource projections print_resources_used= [none|{all}] - print the immediate child resources used by the parent resource print_datasource_accessed= [none|{all}] - print the data source accessed list print_datasource_lineage= [none|{all}] - print the data source lineage print_time= [{no}|yes] - print the time it takes to retrieve the full documentation for each resource and the final time save_file= [{no}|yes] - save the results to a file save_file_intermediate= [{no}|yes] - save the file intermediately after each resource is completed	LONGVARCHAR
IN	**command** – One of the following: print_containers | print_annotations | print_resource_projections | print_resources_used | print_datasource_accessed | print_datasource_lineage | print_time | save_file | save_file_intermediate	VARCHAR
OUT	**commandOptionValue** – The result command value [0,1,2,3] – none|no=0, all|yes=1, nonblank=2, blank=3	INTEGER

2. **Examples:**

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	switches	print_time=yes save_file=yes save_file_intermediate=yes
IN	command	print_time
OUT	commandOptionValue	1

Default values

```
case
        when command = 'print_containers'                  then set commandOptionValue = 0; -- none
        when command = 'print_annotations'         then set commandOptionValue = 1; -- all
        when command = 'print_resource_projections'    then set commandOptionValue = 1; -- all
        when command = 'print_resources_used'              then set commandOptionValue = 1; -- all
        when command = 'print_datasource_accessed'   then set commandOptionValue = 1; -- all
        when command = 'print_datasource_lineage'    then set commandOptionValue = 1; -- all
        when command = 'print_time'                then set commandOptionValue = 0; -- no
        when command = 'save_file'                 then set commandOptionValue = 0; -- no
        when command = 'save_file_initialize'      then set commandOptionValue = 0; -- no
        else                                               set commandOptionValue = 0;
end case;
```

implementations

This folder contains the different preamble and formatting implementations.

implementations/getDocPreambleImpl1

This procedure provides a default implementation for retrieving the preamble to the documentation. The preamble is the text that occurs at the beginning of the documentation prior to the repeatable formatted resource text.

This procedure is invoked by the `getAllDocumentationAPI` in a loop. The output of this procedure provides the formatting for the preamble of the documentation. A user may wish to customize the preamble text for their specific project. The idea behind this procedure is that it provides a template for an implementation. A user of the documentation utilities may choose to copy and create a new implementation and then customize it for their project. Any new implementation "must" follow the input and output interface definitions as shown below.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath** – (optional) Full resource path which includes the path and the resource name .	pathType
IN	**resourceType** – The starting CIS resource type for the resource path.	VARCHAR
IN	**constantPath** – This is the path to the constants file.	pathType
OUT	**formattedText** – formatted text is out output complete with a separator at the beginning of the resource.	PIPE (formattedText /lib/util/System.Tex)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples
IN	resourceType	CONTAINER
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
OUT	formattedText	See below:

```
Composite Software Documentation

Generated on 2012-07-29 00:25:56.510
```

implementations/getDocResourceFormatImpl1

This procedure provides an implementation to retrieve and format the documentation for all resources located in the passed in starting folder.

This procedure is invoked by the getAllDocumentationAPI. This procedure recursively loops through all of the resources found within the given starting folder (CONTAINER). The output of this procedure provides the formatting for all CIS resource. The idea behind this procedure is that it provides a template for an implementation. A user of the documentation utilities may choose to copy and create a new implementation and then customize it for their project. Any new implementation “must” follow the input and output interface definitions as shown below..

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**startingResourcePath** – The starting folder path from which to start formatting the documentation. It is always of type CONTAINER.	pathType
IN	**startingResourceType** – The starting CIS resource type for the resource path.	VARCHAR
IN	**constantPath** – This is the path to the constants file.	pathType
IN	**switches** – Provides guidance on what to print for the documentation and save files.	LONGVARCHAR
IN	**excludeKeywordsInPathList** – Provides guidance on what to print documentation for	LONGVARCHAR
IN	**excludePathsList** – Exclude actual paths. Double quotes are not required. Comma separated list.	LONGVARCHAR

Direction	Parameter Name	Parameter Type
IN	**filePath** – The full file system path to generate the documentation file to.	pathType
OUT	**formattedText** – formatted text is out output complete with a separator at the beginning of the resource.	PIPE (formattedText /lib/util/System.Text)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	startingResourcePath	/shared/examples
IN	startingResourceType	CONTAINER
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
IN	switches	print_time=yes save_file=yes save_file_intermediate=yes
IN	excludeKeywordsInPathList	analysis,archive,test,validation
IN	excludePathsList	/shared/ASAssets/Utilities,/shared/BestPractices,/lib
IN	filePath	/temp/cis_resource_docs.txt
OUT	formattedText	See below:

implementations/getDocResourceFormatImpl1_resource

This procedure provides an implementation to retrieve and format the documentation for a single CIS resource.

This procedure is invoked by the getDocResourceFormatImpl1 which controls the loop. The output of this procedure provides the formatting for a single CIS resource. The idea behind this procedure is that it provides a template for an implementation. A user of the documentation utilities may choose to copy and create a new implementation and then customize it for their project. Any new implementation “must” follow the input and output interface definitions as shown below.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath** – The the path to the resource.	pathType
IN	**resourceType** – The type of resource.	VARCHAR

Direction	Parameter Name	Parameter Type
IN	**constantPath** – This is the path to the constants file.	pathType
IN	**switches** – Provides guidance on what to print for documentation and save files.	LONGVARCHAR
OUT	**formattedText** – formatted text is out output complete with a separator at the beginning of the resource.	PIPE (formattedText /lib/util/System.Text)

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples/CompositeView
IN	resourceType	TABLE
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
IN	switches	print_time=yes save_file=yes save_file_intermediate=yes
OUT	formattedText	See below:

```
================================================================================
Resource Name: CompositeView
Resource Path: /shared/examples/CompositeView
Resource Type: TABLE
      Subtype: SQL_TABLE

Description:
------------
None

Resource Column Projection:
---------------------------
    Column Name              Column Type          Native Base Type     Native Type
    ------------------------ -------------------- -------------------- --------------------
    OrderID                  INTEGER              N/A                  N/A
    ProductID                INTEGER              N/A                  N/A
    Discount                 DOUBLE               N/A                  N/A
    OrderDate                DATE                 N/A                  N/A
    CompanyName              VARCHAR(50)          N/A                  N/A
    CustomerContactFirstName VARCHAR(30)          N/A                  N/A
    CustomerContactLastName  VARCHAR(50)          N/A                  N/A
    CustomerContactPhone     VARCHAR(30)          N/A                  N/A
    ProductName              VARCHAR(32768)       N/A                  N/A
    TransactionID            INTEGER              N/A                  N/A
    DateRequired             DATE                 N/A                  N/A
    DatePromised             DATE                 N/A                  N/A
    ShipDate                 DATE                 N/A                  N/A
    SupplierID               INTEGER              N/A                  N/A
    SupplierName             VARCHAR(50)          N/A                  N/A
    SupplierContactName      VARCHAR(50)          N/A                  N/A
    SupplierPhoneNumber      VARCHAR(30)          N/A                  N/A

Resources Used:
---------------
    Resource Name  Resource Type  Subtype    Resource Path
    -------------- -------------- ---------- -----------------------------
    ViewOrder      TABLE          SQL_TABLE  /shared/examples/ViewOrder
    ViewSales      TABLE          SQL_TABLE  /shared/examples/ViewSales
    ViewSupplier   TABLE          SQL_TABLE  /shared/examples/ViewSupplier

Data Source Accessed List:
---------------------------
    Datasource Name  Enabled Type        Subtype                  Datasource Path
    ---------------- ------- ----------- ------------------------ ------------------------------
    ds_orders        1       DATA_SOURCE RELATIONAL_DATA_SOURCE   /shared/examples/ds_orders
    ds_XML           1       DATA_SOURCE XML_FILE_DATA_SOURCE     /shared/examples/ds_XML
    ds_inventory     1       DATA_SOURCE RELATIONAL_DATA_SOURCE   /shared/examples/ds_inventory


Data Source Lineage:
--------------------
seqnum   id     pid    depth  resource path
     1 - 20587            0  /shared/examples/CompositeView

     2 - 20658  20587      1    /shared/examples/ViewOrder
     3 - 20741  20658      2      [CS] /shared/examples/ds_orders/customers  [TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     4 - 20679  20658      2      [CS] /shared/examples/ds_orders/orderdetails
[TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     5 - 20711  20658      2      [CS] /shared/examples/ds_orders/orders  [TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     6 - 20670  20711      3        /shared/examples/ds_orders
     7 - 20729  20670      4          [CS] /shared/examples/ds_orders/cache_status
[TABLE.DATABASE_TABLE]
                                      [DS] /shared/examples/ds_orders
     8 - 20671  20670      4          [CS] /shared/examples/ds_orders/cache_tracking
[TABLE.DATABASE_TABLE]
                                      [DS] /shared/examples/ds_orders
     9 - 20689  20711      3        [CS] /shared/examples/ds_orders/orders_cache
[TABLE.DATABASE_TABLE]
                                    [DS] /shared/examples/ds_orders

    10 - 20774  20587      1    /shared/examples/ViewSales
    11 - 20679  20774      2      [CS] /shared/examples/ds_orders/orderdetails
[TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
    12 - 20786  20774      2      /shared/examples/productCatalog_Transformation
    13 - 20757  20786      3        [CS] /shared/examples/ds_XML/productCatalog.xml
[TREE.XML_FILE_TREE]
                                    [DS] /shared/examples/ds_XML

    14 - 20763  20587      1    /shared/examples/ViewSupplier
    15 - 20606  20763      2      [CS] /shared/examples/ds_inventory/inventorytransactions
[TABLE.DATABASE_TABLE]
```

```
                                  [DS] /shared/examples/ds_inventory
    16 - 20619  20763      2      [CS] /shared/examples/ds_inventory/purchaseorders
[TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_inventory
    17 - 20644  20763      2      [CS] /shared/examples/ds_inventory/suppliers
[TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_inventory
--------------------------------------------------------------------------------
```

modules

This section describes the modules used in creating documentation.

modules/getDocDataSourceLineage

This procedure returns all the DATA_SOURCE type resources found under the starting path. It returns the formatted text for two sections: "Data Sources Accessed List" and "Data Sources Lineage". The data sources accessed is a distinct list of data sources along with their type, path and whether they are enabled or not.

An example output is shown below:

```
Data Source Accessed List:
---------------------------
    Datasource Name  Enabled Type        Subtype                   Datasource Path
    ---------------- ------- ----------- ------------------------  ------------------------------
    ds_orders        1       DATA_SOURCE RELATIONAL_DATA_SOURCE    /shared/examples/ds_orders
    ds_XML           1       DATA_SOURCE XML_FILE_DATA_SOURCE      /shared/examples/ds_XML
    ds_inventory     1       DATA_SOURCE RELATIONAL_DATA_SOURCE    /shared/examples/ds_inventory
```

The data source lineage provides a top to bottom lineage starting with the "resources used" list. The format of the lineage shows indenting and a depth counter when the depth of the resource changes. Additionally, when a child source is found an indicator of [CS] is placed in front of the resource.   The type of that resource is placed at the end of the child resource path in the format of [TYPE.SUBTYPE]. The parent data source path is placed under neath the child with a [DS] indicator in front of it.

An example output is shown below:

```
Data Source Lineage:
--------------------
seqnum   id     pid    depth  resource path
     1 - 20587             0  /shared/examples/CompositeView

     2 - 20658  20587      1    /shared/examples/ViewOrder
     3 - 20741  20658      2      [CS] /shared/examples/ds_orders/customers [TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     4 - 20679  20658      2      [CS] /shared/examples/ds_orders/orderdetails
[TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     5 - 20711  20658      2      [CS] /shared/examples/ds_orders/orders  [TABLE.DATABASE_TABLE]
                                  [DS] /shared/examples/ds_orders
     6 - 20670  20711      3        /shared/examples/ds_orders
     7 - 20729  20670      4          [CS] /shared/examples/ds_orders/cache_status
[TABLE.DATABASE_TABLE]
```

```
                                   [DS] /shared/examples/ds_orders
     8 - 20671  20670      4         [CS] /shared/examples/ds_orders/cache_tracking
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_orders
     9 - 20689  20711      3       [CS] /shared/examples/ds_orders/orders_cache
[TABLE.DATABASE_TABLE]
                                 [DS] /shared/examples/ds_orders
```

Parameters:

Direction	Parameter Name	Parameter Type
IN	**resourcePath –** Full resource path which includes the path and the resource name .	pathType
IN	**resourceType –** The type of resource.	VARCHAR
IN	**constantPath** – This is the path to the constants file.	pathType
IN	**commandOptionValueDsAccessed** – The result command ('print_datasource_accessed') value [0,1] - none=0, all=1	INTEGER
IN	**commandOptionValueDsLineage** – The result command ('print_datasource_lineage') value [0,1] - none=0, all=1	INTEGER
OUT	**formattedText –** formatted text is out output complete with a separator at the beginning of the resource.	/lib/util/System.Text

1. **Examples:**

1.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples/CompositeView
IN	resourceType	TABLE
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
IN	commandOptionValueDsAccessed	1
IN	commandOptionValueDsLineage	1
OUT	formattedText	See below:

```
Data Source Accessed List:
---------------------------
    Datasource Name  Enabled Type        Subtype                   Datasource Path
    ---------------- ------- ----------- ------------------------- ------------------------------
    ds_orders        1       DATA_SOURCE RELATIONAL_DATA_SOURCE    /shared/examples/ds_orders
    ds_XML           1       DATA_SOURCE XML_FILE_DATA_SOURCE      /shared/examples/ds_XML
    ds_inventory     1       DATA_SOURCE RELATIONAL_DATA_SOURCE    /shared/examples/ds_inventory
```

```
Data Source Lineage:
--------------------
seqnum   id     pid     depth  resource path
     1 - 20587              0  /shared/examples/CompositeView

     2 - 20658  20587       1    /shared/examples/ViewOrder
     3 - 20741  20658       2      [CS] /shared/examples/ds_orders/customers  [TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_orders
     4 - 20679  20658       2      [CS] /shared/examples/ds_orders/orderdetails
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_orders
     5 - 20711  20658       2      [CS] /shared/examples/ds_orders/orders  [TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_orders
     6 - 20670  20711       3        /shared/examples/ds_orders
     7 - 20729  20670       4          [CS] /shared/examples/ds_orders/cache_status
[TABLE.DATABASE_TABLE]
                                       [DS] /shared/examples/ds_orders
     8 - 20671  20670       4          [CS] /shared/examples/ds_orders/cache_tracking
[TABLE.DATABASE_TABLE]
                                       [DS] /shared/examples/ds_orders
     9 - 20689  20711       3        [CS] /shared/examples/ds_orders/orders_cache
[TABLE.DATABASE_TABLE]
                                     [DS] /shared/examples/ds_orders

    10 - 20774  20587       1    /shared/examples/ViewSales
    11 - 20679  20774       2      [CS] /shared/examples/ds_orders/orderdetails
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_orders
    12 - 20786  20774       2      /shared/examples/productCatalog_Transformation
    13 - 20757  20786       3        [CS] /shared/examples/ds_XML/productCatalog.xml
[TREE.XML_FILE_TREE]
                                     [DS] /shared/examples/ds_XML

    14 - 20763  20587       1    /shared/examples/ViewSupplier
    15 - 20606  20763       2      [CS] /shared/examples/ds_inventory/inventorytransactions
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_inventory
    16 - 20619  20763       2      [CS] /shared/examples/ds_inventory/purchaseorders
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_inventory
    17 - 20644  20763       2      [CS] /shared/examples/ds_inventory/suppliers
[TABLE.DATABASE_TABLE]
                                   [DS] /shared/examples/ds_inventory
```

modules/getDocResourceProjection

This procedure returns the column projection for TABLES and PROCEDURES.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath –** Full resource path which includes the path and the resource name .	pathType

Direction	Parameter Name	Parameter Type
IN	**resourceType –** Type of CIS resource to be created	VARCHAR
IN	**constantPath** – This is the path to the constants file.	pathType
OUT	**formattedText –** formatted text is out output complete with a separator at the beginning of the resource.	/lib/util/System.Text

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples/CompositeView
IN	resourceType	TABLE
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
OUT	formattedText	See below:

```
Resource Column Projection:
---------------------------
    Column Name              Column Type          Native Base Type     Native Type
    ------------------------ -------------------- -------------------- -----------------
    OrderID                  INTEGER              N/A                  N/A
    ProductID                INTEGER              N/A                  N/A
    Discount                 DOUBLE               N/A                  N/A
    OrderDate                DATE                 N/A                  N/A
    CompanyName              VARCHAR(50)          N/A                  N/A
    CustomerContactFirstName VARCHAR(30)          N/A                  N/A
    CustomerContactLastName  VARCHAR(50)          N/A                  N/A
    CustomerContactPhone     VARCHAR(30)          N/A                  N/A
    ProductName              VARCHAR(32768)       N/A                  N/A
    TransactionID            INTEGER              N/A                  N/A
    DateRequired             DATE                 N/A                  N/A
    DatePromised             DATE                 N/A                  N/A
    ShipDate                 DATE                 N/A                  N/A
    SupplierID               INTEGER              N/A                  N/A
    SupplierName             VARCHAR(50)          N/A                  N/A
    SupplierContactName      VARCHAR(50)          N/A                  N/A
    SupplierPhoneNumber      VARCHAR(30)          N/A                  N/A
```

modules/getDocResourcesUsed

This procedure returns the list of resources (level 1) used by this resources. The immediate resource list is the list of resources that are directly invoked by the current resource being formatted. In the example below the resource being formatted is `/shared/examples/CompositeView`. The `CompositeView` has three resources that it uses for immediate invocation which include `ViewOrder`, `ViewSales` and `ViewSupplier`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath –** Full resource path which includes the path and the resource name .	pathType
IN	**resourceType –** Type of CIS resource to be created	VARCHAR
IN	**constantPath** – This is the path to the constants file.	pathType
OUT	**formattedText –** formatted text is out output complete with a separator at the beginning of the resource.	/lib/util/System.Text

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples/CompositeView
IN	resourceType	TABLE
IN	constantPath	/shared/ASAssets/Utilities/documentation/constants()
OUT	formattedText	See below:

```
Resources Used:
---------------
    Resource Name  Resource Type  Subtype    Resource Path
    -------------- -------------- ---------- -----------------------------
    ViewOrder      TABLE          SQL_TABLE  /shared/examples/ViewOrder
    ViewSales      TABLE          SQL_TABLE  /shared/examples/ViewSales
    ViewSupplier   TABLE          SQL_TABLE  /shared/examples/ViewSupplier
```

9 How To Use ‘Encoding’ Procedures

Introduction

This section describes the routines for encoding, decoding, and encrypting text.

CIS_JCE_PROVIDERS_VIEW

A wrapper view for `encoding/EncodingCJP/CISSecurityProviders()`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	results	CURSOR (Provider LONGVARCHAR Algorithm LONGVARCHAR "Service Description" LONGVARCHAR)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	results	('SUN version 1.6', 'CaseExactJKS', 'SUN: KeyStore.CaseExactJKS -> sun.security.provider.JavaKeyStore$CaseExactJKS'), …

EncodingCJP

This section will show how to use the ‘Encoding’ CJP procedures.

EncodingCJP/Base64Decode (Custom Function)

Accepts a Base64 encoded string as input and returns the Base64 decoded value of the string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	base64EncodedString	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	base64EncodedString	‘Zm9v’
OUT	result	‘foo’

EncodingCJP/Base64Encode (Custom Function)

Accepts a string as input and returns the Base64 encoded value of the string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	‘foo’
OUT	result	‘Zm9v’

EncodingCJP/CISSecurityProviders

Lists all JCE providers, services and algorithms configured in the CIS JVM. A simple wrapper view, CIS_JCE_PROVIDERS_VIEW, on top of this procedure (see above) allows for the lookup of algorithms. This procedure can be used, for example, to track down the root cause of failures in a client certificate and/or mutual authentication schemes between CIS and secure data providers (REST and SOAP web-services, some DBMS with advanced security mechanisms) or clients (SOAP and REST service consumes, app servers, ESBs, etc.) when these failures are caused by unsupported security algorithms.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	results	CURSOR (Provider LONGVARCHAR Algorithm LONGVARCHAR "Service Description" LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	results	('SUN version 1.6', 'CaseExactJKS', 'SUN: KeyStore.CaseExactJKS -> sun.security.provider.JavaKeyStore$CaseExactJKS'), …

EncodingCJP/DecryptFrom3DES

Decrypts a symmetrical Triple DES encrypted string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"encrypted hex string"	LONGVARCHAR
IN	"digest seed"	LONGVARCHAR
OUT	"plain text"	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"encrypted hex string"	'975d580dc4134fefbf290d8841f66dac'
IN	"digest seed"	'the text used as the digest seed'
OUT	"plain text"	'this is a test'

EncodingCJP/DecryptFromAES

Decrypts an AES-encrypted string.

This procedure provides an implementation of symmetrical AES decryption. This can be used to decode sensitive data in CIS views when such a security requirement exists.

With OOTB installation of CIS, the procedure will be capable of using a 128-bit encryption key. Stronger encryption (192-bit and 256-bit) are also supported, but you must replace the policy files `local_policy.jar` and `US_export_policy.jar` files under `jre/lib/security` with the unlimited strength policies downloadable from the Oracle web site, and restart CIS.

Unlike the 3DES implementation, this AES decryption procedure allows the caller to specify both the encryption key and the initialization vector (IV) to feed into the algorithm.

However, for simplicity, if the caller does not supply the IV, it will be auto-generated from the encryption key. Note that this approach weakens the encryption, as the IV remains constant for all messages encrypted with the same key. Hence this practice is not recommended (or rather recommended not to follow.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"encrypted hex string"	LONGVARCHAR
IN	"key"	LONGVARCHAR
IN	"IV"	LONGVARCHAR
OUT	"plain text"	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"encrypted hex string"	'Tvc7ThIwAyZ3EJ381gnIXQ=='
IN	"key"	'1234567812345678'
IN	"IV"	NULL
OUT	"plain text"	'this is a test'

EncodingCJP/DecryptWithCISPrivKey

Decrypts a string encrypted using CIS's built in SSL certificate.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"encrypted hex string"	LONGVARCHAR
IN	"keystore password"	LONGVARCHAR
OUT	"plain text"	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"encrypted hex string"	'1232F6C78E9CE0D3EEB9CE05463942458C7689F 19CF45664B9BA8FFADDDBFBE075880BAE9CE BA93FF560368D96896E28E100FC79604CC75175 E4286B3B3D4BFDA6EA5A32E0568ECFB4C745D 8FC8A58CFCBF3BBE1C00BC55A97E23C7571705 2BFE51131E38D8504FB35E8393C277E7BEF9E5E

Direction	Parameter Name	Parameter Value
		36CDEF5D19F6769CA673F2AD65EE'
IN	"keystore password"	'changeit'
OUT	"plain text"	'this is a test'

EncodingCJP/EncryptWith3DES

Encrypts a string using symmetrical Triple DES encryption.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"plain text"	LONGVARCHAR
IN	"digest seed"	LONGVARCHAR
OUT	"encrypted raw bytes"	LONGVARBINARY
OUT	"encrypted hex string"	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"plain text"	'this is a test'
IN	"digest seed"	'any text should work here'
OUT	"encrypted raw bytes"	b1bca4647b3b859f6f4736c595596fc0
OUT	"encrypted hex string"	'B1BCA4647B3B859F6F4736C595596FC0'

EncodingCJP/EncryptWithAES

Encrypts a string using AES encryption (in CBC mode.)

This procedure provides an implementation of symmetrical AES encryption. This can be used to encode sensitive data in CIS views when such a security requirement exists.

With OOTB installation of CIS, the procedure will be capable of using a 128-bit encryption key. Stronger encryption (192-bit and 256-bit) are also supported, but you must replace the policy files `local_policy.jar` and `US_export_policy.jar` files under `jre/lib/security` with the unlimited strength policies downloadable from the Oracle web site, and restart CIS.

Unlike the 3DES implementation, this AES encryption procedure allows the caller to specify both the encryption key and the initialization vector (IV) to feed into the algorithm.

However, for simplicity, if the caller does not supply the IV, it will be auto-generated from the encryption key. Note that this approach weakens the encryption, as the IV remains constant for all messages encrypted with the same key. Hence this practice is not recommended (or rather recommended not to follow.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"plain text"	LONGVARCHAR
IN	"key"	LONGVARCHAR
IN	"IV"	LONGVARCHAR
OUT	"encrypted raw bytes"	LONGVARBINARY
OUT	"encrypted string (base-64)"	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"plain text"	'this is a test'
IN	"key"	'1234567812345678'
IN	"IV"	NULL
OUT	"encrypted raw bytes"	4ef73b4e1230032677109dfcd609c85d
OUT	"encrypted string (base-64)"	'Tvc7ThIwAyZ3EJ381gnIXQ=='

EncodingCJP/EncryptWithCISPubKey

Encrypts a string using CIS's built in SSL certificate.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	"encrypted hex string"	LONGVARCHAR
IN	"keystore password"	LONGVARCHAR
OUT	"plain text"	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	"plain text"	'this is a test'
IN	"keystore password"	'changeit'

Direction	Parameter Name	Parameter Value
OUT	"encrypted raw bytes"	1232f6c78e9ce0d3eeb9ce05463942458c7689f1...128
OUT	"encrypted hex string"	'1232F6C78E9CE0D3EEB9CE05463942458C7689F1 9CF45664B9BA8FFADDDBFBE075880BAE9CEBA 93FF560368D96896E28E100FC79604CC75175E428 6B3B3D4BFDA6EA5A32E0568ECFB4C745D8FC8 A58CFCBF3BBE1C00BC55A97E23C75717052BFE 51131E38D8504FB35E8393C277E7BEF9E5E36CD EF5D19F6769CA673F2AD65EE'

EncodingCJP/MD5Hash (Custom Function)

Accepts a string as input and returns the MD5 hash value of the string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	‘foo’
OUT	result	‘acbd18db4cc2f85cedef654fccc4a4d8’

EncodingCJP/SHA1Hash (Custom Function)

Accepts a string as input and returns the SHA1 hash value of the string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	‘foo’
OUT	result	‘0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33’

10 How To Use ‘Environment’ Procedures

Introduction

This section will show how to use the ‘Environment’ procedures.

NOTE: The environment procedures provide a way to get data virtualization environment settings.

getEnvName (Custom Function)

This procedure returns the environment name for this specific server. It is useful when sending emails to be able to identify the environment type in the subject line giving the recipient a quick way of determining which environment is having issues. It may also be used by procedural logic to perform different logic based on the environment type.

It is intended to be modified for each environment. For a cluster it will have the same value by default because the DV code is duplicated on each server. The value can be whatever the DV development team wants up to 255 characters.

Some examples are shown below:

DEV=TDV Development environment

TEST=TDV Test environment

PROD=TDV Production environment

DR=TDV Disaster Recovery environment

** WARNING **: Be aware that when you update the Utilities, you must modify the environment setting again as it will be overwritten.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	envName	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	envName	DEV

11 How To Use 'File' Procedures

Introduction

This section will show how to use the 'File' procedures.

NOTE: When accessing a CIS instance remotely, the 'File' procedures act on the file system of the CIS instance's host and not the file system of the host running the Composite Studio client.

copyAll

This procedure is used to copy all of the files from one source location to target destination.

In this example, the end point folder name in the sourcePath will be used as the starting point and concatenated onto the end point of the target folder. Therefore, the result will be all folders under /Temp/vcs1/Utilities will be recursively copied to /Temp/vcs2 resulting in /Temp/vcs2/Utilities....

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	newFilePath	VARCHAR(2147483647)
INOUT	mkdirCount (null initially)	INTEGER
INOUT	copyCount (null initially)	INTEGER

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourcePath	'/Temp/vcs1/Utilities'
IN	targetpath	'/Temp/vcs2'
INOUT	mkdirCount	350
INOUT	copyCount	451

getCisHome (Custom Function)

This function returns the folder on the CIS host where CIS is installed.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
OUT	result	VARCHAR(4096)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	‘C:\Program Files\Composite Software\CIS 6.0.0’

getFileSeparator (Custom Function)

This function returns the character used to separate folders/files in a path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	CHAR(1)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	‘\’ (when run on a Windows host.)

removeAllFilter

This procedure is used to recursively remove all of the designated filter directories starting at a given source Path. For example, to remove the .svn directory from all levels of a source path, set the directoryFilter='.svn'.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourcePath	VARCHAR(2147483647)
IN	directoryFilter	VARCHAR
INOUT	removeCount (null initially)	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourcePath	‘/Temp/vcs2/Utilities’

Direction	Parameter Name	Parameter Value
IN	directoryFilter	'.svn'
INOUT	removeCount	39

FileProcessingCJP

This section will show how to use the 'File' CJP procedures.

FileProcessingCJP/archiveFile

Archive a file by moving it from a source directory to a target directory.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	archivalDirectoryPath	VARCHAR(2147483647)
OUT	None: Throws exception upon failure	

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/file1.txt'
OUT	archivalDirectoryPath	'/files/archive'

FileProcessingCJP/archiveFileTimestamp

Archive a file by moving it from a source directory to a target directory. Also renames the file to include a timestamp.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	archivalDirectoryPath	VARCHAR(2147483647)
OUT	None: Throws exception upon failure	

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/file1.txt'

Direction	Parameter Name	Parameter Value
OUT	archivalDirectoryPath	'/files/archive'

FileProcessingCJP/copyFile

Copy a file from a source directory to a target directory.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	newFilePath	VARCHAR(2147483647)
OUT	None: Throws exception upon failure	

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/file1.txt'
OUT	newFilePath	'/files/copydir'

FileProcessingCJP/createFileASCII

Create an ASCII text file in a target directory. Provides an option to append which is useful for logging type files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	Append 0=do not append file, 1=append file	SMALLINT
IN	fileContent	LONGVARCHAR
OUT	None: Throws exception upon failure	

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/newfile.txt'
IN	append	0

Direction	Parameter Name	Parameter Value
IN	fileContent	'This is new file text.'

FileProcessingCJP/createFileBinary

Create a binary file in a target directory.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
IN	Append 0=do not append file, 1=append file	SMALLINT
IN	fileContent	LONGVARBINARY
OUT	None: Throws exception upon failure	

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/newfile.dat'
IN	append	0
IN	fileContent	--binary content is sent--

FileProcessingCJP/existsDir (Custom Function)

Check to see if the requested director exists.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dirPath	VARCHAR(2147483647)
OUT	success	BOOLEAN

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dirPath	'/files/incoming'
OUT	success (1=true, 0=false)	1

FileProcessingCJP/existsFile (Custom Function)

Check to see if the requested file exists.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
OUT	success	BOOLEAN

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/newfile.txt'
OUT	success (1=true, 0=false)	1

FileProcessingCJP/getFileContentsAscii (Custom Function)

Get the contents of an ASCII file from the requested file path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
OUT	fileContent	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	0
OUT	fileContent	20

FileProcessingCJP/getFileContentsBinary (Custom Function)

Get the contents of a binary file from the requested file path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
OUT	fileContent	LONGVARBINARY

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	0
OUT	fileContent	20

FileProcessingCJP/getFileInfo

Get the file metadata.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	directoryPath	VARCHAR(2147483647)
IN	includeDirs – include directories names in the response. Y=include, N=do not include	VARCHAR(255)
OUT	FileInfo	CURSOR
	filePath	VARCHAR(2147483647)
	fileName	VARCHAR(2147483647)
	fileTimestamp	TIMESTAMP
	fileSize	BIGINT
	isFile	SMALLINT
	isDir	SMALLINT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	directoryPath	'/files/incoming'
IN	includeDirs – include directories names in the response. Y=include, N=do not include	'Y'
OUT	FileInfo	CURSOR

filePath	filename	fileTimestamp	fileSize	isFile	isDir
\files\incoming	incoming	2010-07-16 13:00:28.65	0	0	1
\files\incoming\file1.txt	file1.txt	2010-07-16 13:28:17.915	512	1	0
\files\incoming\file2.txt	file2.txt	2010-07-16 13:29:01.001	1024	1	0

Direction	Parameter Name	Parameter Value			
\files\incoming\file3.txt	file3.txt	2010-07-16 13:30:43.873	58	1	0

FileProcessingCJP/getNewFiles

Get the new files that appear in the requested directory.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	directoryPath	VARCHAR(2147483647)
OUT	newFilenames	CURSOR
	filePath	VARCHAR(2147483647)
	fileName	VARCHAR(2147483647)
	fileTimestamp	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	directoryPath	'/files/incoming'
OUT	FileInfo	CURSOR
filePath	filename	fileTimestamp
\files\incoming\file1.txt	file1.txt	2010-07-16 13:28:17.915
\files\incoming\file2.txt	file2.txt	2010-07-16 13:29:01.001
\files\incoming\file3.txt	file3.txt	2010-07-16 13:30:43.873

FileProcessingCJP/gunzipFile (Custom Function)

Gunzip a gzip file for the requested file path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
OUT	success (1=true, 0=false)	BOOLEAN

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/files.gzip'
OUT	success (1=true, 0=false)	1

FileProcessingCJP/makeDirs (Custom Function)

Make the all the directories for the requested directory path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dirPath	VARCHAR(2147483647)
OUT	success (1=true, 0=false)	BOOLEAN

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dirPath	'/files/incoming/newdir'
OUT	success (1=true, 0=false)	1

FileProcessingCJP/removeAll (Custom Function)

Remove all files and folders in the specific file path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dirPath	VARCHAR(2147483647)
IN	removeDirs Y=remove all files and directories. N=remove only files and leave the directory structure in place.	VARCHAR
OUT	success (1=true, 0=false)	BOOLEAN

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dirPath	'/files/incoming/newdir'
IN	removeDirs	'Y'

Direction	Parameter Name	Parameter Value
OUT	success (1=true, 0=false)	1

FileProcessingCJP/remove (Custom Function)

Remove a specific file or directory from the file system. For directories, it only removes a single directory at the end of a path and not the entire path of directories.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dirPath	VARCHAR(2147483647)
OUT	success (1=true, 0=false)	BOOLEAN

2. **Examples:**

    2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dirPath	'/files/incoming/newdir'
OUT	success (1=true, 0=false)	1

FileProcessingCJP/unzipFile (Custom Function)

Unzip a zip file for the requested file path.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	filePath	VARCHAR(2147483647)
OUT	success (1=true, 0=false)	BOOLEAN

4. **Examples:**

    4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	filePath	'/files/incoming/files.zip'
OUT	success (1=true, 0=false)	1

12 How To Use ‘Generate’ Procedures

Introduction

This section will show how to use the ‘Generate’ procedures.

NOTE: The generation procedures provide a framework to generate views from a relational data source into the Data Abstraction Best Practices layers. In addition to generation there are also procedures that can be used to destroy resources from the Data Abstraction Best Practices layers.

generateViews

This script is used to provide a framework for introspection of a relational data source and generating views to the various Data Abstraction Best Practices layers. It provides a simple view generation into the four main layers described by the Data Abstraction Best Practices. This procedure copies privileges from the parent folder so it is important that proper privileges be assigned ahead of time on the project level folders and recursively pushed down to the layer folders. This is a simplified version of the open source Data Abstraction Best Practices in that it does not allow a formatting layer logical names to be derived from a spreadsheet.

This script will also copy both table and column annotations if they exist at the data source level. This script will copy SQL indexes and foreign keys from the data source to the layers. The published layer will inherit indexes and foreign keys directly from the view/table it was published from. Published views will have their own annotation but they will inherit column annotations from the view they were published from.

Published

/services/databases/Published_Database/[Catalog]/Schema - mandatory

Application

/Application/Views/folder - optional layer generation

Business

/Business/Logical/folder - optional layer generation

Physical

/Physical/Formatting/folder - optional layer generation

/Physical/Metadata/[catalog]/schema - mandatory

The top [published] and bottom [metadata] layers are mandatory. The middle layers are optional and the input can be left null to indicate that no generation is required in those layers. Below is a representation of the Data Abstraction Layers and how the input variables map to those layers.

```
/services/databases
        /TEST                                   <-- exists ----^
                /CAT1                           <-- create ----^
                        /SCH1                   <-- create ----^ = Published_DB_Layer_Path
                                /V_T1           <-- create ----^
/shared                                                        |
        /TEST                                                  |
                /Application                                   |
                        /Views/SCH1             <-- create     ^ = Application_Layer_Path
                                        /T1     <-- create     |
                /Business                                      |
                        /Logical/SCH1           <-- create     ^ = Business_Layer_Path
                                        /T1     <-- create     |
                /Physical                                      |
                        /Formatting/SCH1        <-- create     ^ = Formating_Layer_Path
                                        /T1     <-- create     |
                        /Metadata/Oracle/DS1                   | = datasourcePath
                                        /SCH1   <-- exists     | = Source_Phyisical_Path [Schema Path] and just the schemaName
                                            /T1 <-- exists   - | [Table exists]
                                            /T2
```

5. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Source_Phyisical_Path	VARCHAR(32768)
IN	datasourcePath	VARCHAR(4096)
IN	catalogName	VARCHAR
IN	schemaName	VARCHAR
IN	schemaTablePatterns	VARCHAR(4096)
IN	tableNames	LONGVARCHAR
IN	schemaProcedurePatterns	VARCHAR(4096)
IN	procedureNames	LONGVARCHAR
IN	separator	VARCHAR
IN	Formating_Layer_Path	VARCHAR(32768)
IN	Business_Layer_Path	VARCHAR(32768)
IN	Application_Layer_Path	VARCHAR(32768)
IN	Published_DB_Layer_Path	VARCHAR(32768)

Direction	Parameter Name	Parameter Type
IN	prefix	VARCHAR(255)
IN	overwrite	SMALLINT
IN	copyAnnotation	SMALLINT
IN	copySqlIndexes	SMALLINT
IN	copyForeignKeys	SMALLINT
IN	debug	CHAR(1)
OUT	errStatus	VARCHAR
OUT	errMessage	LONGVARCHAR
OUT	introspectionResult	LONGVARCHAR
OUT	numResourcesIntrospectAdd	INTEGER
OUT	numTablesIntrospectAdd	INTEGER
OUT	tablesIntrospectAdd	LONGVARCHAR
OUT	numResourcesIntrospectSkip	INTEGER
OUT	numTablesIntrospectSkip	INTEGER
OUT	tablesIntrospectSkip	LONGVARCHAR

6. Examples:

6.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	Source_Phyisical_Path	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Physical/Metadata/postgres/ds_inventory/tutorial
IN	datasourcePath	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Physical/Metadata/postgres/ds_inventory
IN	catalogName	null
IN	schemaName	tutorial
IN	schemaTablePatterns	null
IN	tableNames	categories,products
IN	schemaProcedurePatterns	null
IN	procedureNames	null
IN	separator	,
IN	Formating_Layer_Path	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Physical/Formatting/tutorial
IN	Business_Layer_Path	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Business/Logical/tutorial

Direction	Parameter Name	Parameter Value
IN	Application_Layer_Path	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Application/Views/tutorial
IN	Published_DB_Layer_Path	/services/databases/ExampleProject
IN	prefix	v_
IN	overwrite	1
IN	copyAnnotation	1
IN	copySqlIndexes	1
IN	copyForeignKeys	1
IN	debug	Y
OUT	errStatus	SUCCESS
OUT	errMessage	null
OUT	introspectionResult	<output from introspection>
OUT	numResourcesIntrospectAdd	2
OUT	numTablesIntrospectAdd	2
OUT	tablesIntrospectAdd	inventorytransactions,purchaseorders
OUT	numResourcesIntrospectSkip	5
OUT	numTablesIntrospectSkip	4
OUT	tablesIntrospectSkip	categories,employees,products,suppliers

destroyDependentLineage [CONTAINER/TABLE/LINK only]

This procedure recursively destroys all of the "dependent" resources for a given starting folder (container), table or link resource. The general use case is to destroy a metadata, data source, schema folder or table which in turn destroys their dependent lineage from bottom to top with respect to the data abstraction layers. The starting path may be at any layer within the Data Abstraction Best Practices layers. In essence, the table/view lineage traverses the layers as shown in the graphic below. The term "dependent" refers to any table/view that selects another view. In reality give the layered model the deleting happens to the table/view that was selected and table/views that are considered above in the layers all the way to the published resource. It is important to note that this procedure operates from the bottom up in terms of the lineage. In other words, given a folder of resources, it will destroy the dependent lineage for each table resource found in that folder and the resources that select that table.

Caution: If multiple resources (tables) are selecting a single resource then **ALL** resources in that lineage will be destroyed no matter where those resources exist in the data virtualization server. Think of this deletion as a tree. It will start at the root level, move up the trunk and fan out to all branches that are connected via the lineage and destroy all leaves of the tree that are connected.

A cursor of metadata is returned that shows which resources were destroyed. For example, if the invoker of this procedure wants to destroy the Physical/Metadata datasource schema and all views that the resource is dependent upon throughout the layers, then this procedure will accomplish that goal.

Values: ex1 (view): /shared/TEST/Physical/Metadata/Oracle/DS1/T1

In this scenario, only the data source table "T1" and its lineage is removed.

The database table T1 will be destroyed if the input destroyInitialResource=1.

If a folder is empty the folder will be removed if the input destroyEmptyParentFolder=1.

```
/services/databases
            /TEST
                  /CAT1
                        /SCH1
                              /V_T1              <-- destroy        ^
/shared                                                             |
            /TEST                                                   |
                  /Application                                      |
                        /Published/V_T1          <-- destroy        ^
                        /Views/V_T1              <-- destroy        ^
                  /Business                                         |
                        /Business/V_T1           <-- destroy        ^
                        /Logical/V_T1            <-- destroy        ^
                  /Physical                                         |
                        /Formatting/T1           <-- destroy        ^
                        /Metadata/Oracle/DS1                        |
                                      /T1        <-- start here     | - destroy table and lineage
                                      /T2
```

Values: ex2 (folder): /shared/TEST/Physical/Metadata/Oracle/DS1

In this scenario, the data source schema "DS1" and its table lineage is removed.

The database tables will be destroyed if the input destroyInitialResource=1.

If a folder is empty the folder will be removed if the input destroyEmptyParentFolder=1.

```
/services/databases
            /TEST
                  /CAT1
                        /SCH1
                              /V_T1              <-- destroy        ^
                              /V_T2              <-- destroy        ^
```

```
/shared                                                    |
          /TEST                                            |
               /Application                                |
                    /Published/V_T1     <-- destroy        ^
                          /V_T2         <-- destroy        ^
                    /Views/V_T1         <-- destroy        ^
                        /V_T2           <-- destroy        ^
               /Business                                   |
                    /Business/V_T1      <-- destroy        ^
                         /V_T2          <-- destroy        ^
                    /Logical/V_T1       <-- destroy        ^
                        /V_T2           <-- destroy        ^
               /Physical                                   |
                    /Formatting/T1      <-- destroy        ^
                          /T2           <-- destroy        ^
                    /Metadata/Oracle/DS1 <-- start here    | - destroy starting folder of resources
                                    /T1 <-- destroy        |
                                    /T2 <-- destroy        |
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	destroyInitialResource	SMALLINT
IN	destroyEmptyParentFolder	SMALLINT
IN	fullResourcePath	LONGVARCHAR
IN	inDebug	CHAR(1)
OUT	result PIPE (actionType id resourceName resourcePath resourceType subtype ownerDomain ownerName impactLevel impactMessage enabled)	 VARCHAR(255), INTEGER, VARCHAR(255), VARCHAR(4096), VARCHAR(40), VARCHAR(40), VARCHAR(255), VARCHAR(255), VARCHAR(255), VARCHAR(32768), BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	destroyInitialResource	1
IN	destroyEmptyParentFolder	1
IN	fullResourcePath	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Physical/Metadata/postgres/ds_inventory/tutorial
IN	inDebug	Y
OUT	result	Cursor of resources removed

destroyUsedLineage [CONTAINER/LINK/TABLE only]

This procedure recursively destroys the "used" resource for a given starting published resource, shared resource or folder. The general use case is to destroy a published folder or view which in turn destroys their used lineage from top to bottom with respect to the data abstraction layers. The resource "may" be a published link resource or folder container. If it is a published link then the "used" lineage is destroyed except any related foreign key and cache resources. If it is a published folder then all of the link/table resources are destroyed along with their "used" resource lineage. Related foreign key and cache are not destroyed.

The resource "may" be a /shared table/view resource or folder container. The same rules apply as above. However, an additional rule pertains to the dependent resources of a table resource. If a shared table is removed and it has a dependent resource, the dependent resource is left orphaned and will show up as red/impacted in Studio.

The valid resource types are CONTAINER (a.k.a FOLDER), LINK (published) or TABLE (shared). Procedures and non-TABLE/LINK resources are not supported and will be bypassed.

A cursor of metadata is returned that shows which resources were destroyed. For example, if the invoker of this procedure wants to destroy the published schema table and all views that the resource is using throughout the layers, then this procedure will accomplish that goal.

A specific view (link) resource may be deleted with its lineage or an entire schema of resources and their lineage may be deleted. The following is an example of the fullResourcePath:

Values: ex1 (view): /services/databases/TEST/CAT1/SCH1/V_T1

In this scenario, only the published view "V_T1" and its lineage is removed.

The database table T1 will be destroyed if the input destroyDatasourceResource=1.

If a folder is empty the folder will be removed if the input destroyEmptyParentFolder=1.

```
/services/databases
                  /TEST
                        /CAT1
```

```
                                        /SCH1
                                                  /V_T1            <-- start here    v
/shared                                                                              |
                    /TEST                                                            |
                              /Application                                           |
                                        /Published/V_T1            <-- destroy       v
                                        /Views/V_T1                <-- destroy       v
                              /Business                                              |
                                        /Business/V_T1             <-- destroy       v
                                        /Logical/V_T1              <-- destroy       v
                              /Physical                                              |
                                        /Formatting/T1             <-- destroy       v
                                        /Metadata/Oracle/DS1                         |
                                                        /T1        <-- destroy       |
                                                        /T2
```

Values: ex2 (folder): /services/databases/TEST/CAT1/SCH1

In this scenario, the entire schema "SCH1" and all the views and their lineage is removed.

This includes all layers and the metadata schema as long as the metadata schema is empty and the input destroyEmptyParentFolder=1.

```
/services/databases
                    /TEST
                              /CAT1
                                        /SCH1              <-- start here | - destroy the schema, all views and lineage
                                                  /V_T1    <-- destroy    v - destroy V_T1 and its lineage only
                                                  /V_T2    <-- destroy    v - destroy V_T2 and its lineage only
/shared                                                                   |
          /TEST                                                           |
                    /Application                                          |
                              /Published/SCH1/V_T1         <-- destroy    v - destroy V_T1 and SCH1 if no objects remain
                                             /V_T2         <-- destroy    v - destroy V_T2
                              /Views/SCH1/V_T1             <-- destroy    v - destroy V_T1 and SCH1 if no objects remain
                                         /V_T2             <-- destroy    v - destroy V_T2
                    /Business                                             |
                              /Business/SCH1/V_T1          <-- destroy    v - destroy V_T1 and SCH1 if no objects remain
                                            /V_T2          <-- destroy    v - destroy V_T2
                              /Logical/SCH1/V_T1           <-- destroy    v - destroy V_T1 and SCH1 if no objects remain
                                           /V_T2           <-- destroy    v - destroy V_T2
                    /Physical                                             |
                              /Formatting/SCH1/T1          <-- destroy    v - destroy T1 and SCH1 if no objects remain
                                              /T2          <-- destroy    v - destroy T2
```

```
/Metadata/Oracle/DS1                    |
                    /SCH1  <-- destroy  | - destroy SCH1 if no objects remain
                      /T1  <-- destroy  | - destroy T1
                      /T2  <-- destroy  | - destroy T2
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	destroyDatasourceResource	SMALLINT
IN	destroyEmptyParentFolder	SMALLINT
IN	fullResourcePath	LONGVARCHAR
IN	inDebug	CHAR(1)
OUT	result PIPE (actionType seqNum id parentID resourceDepth treeType resourceName resourcePath resourceType subtype enabled)	 VARCHAR(255), INTEGER, INTEGER, INTEGER, INTEGER, VARCHAR(255), VARCHAR(255), VARCHAR(4096), VARCHAR(255), VARCHAR(255), BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	destroyDatasourceResource	1
IN	destroyEmptyParentFolder	/1
IN	fullResourcePath	/services/databases/ExampleProject/TestCatalog/tutorial
IN	inDebug	Y
OUT	result	Cursor of resources removed

/helpers/createResourceProcess

The is a helper procedure which is used to manage the creation of the folder and views within the Data Abstraction Best Practices layers. It is invoked by generateViews().

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Res_Source_Path	VARCHAR(32768)
IN	Res_Target_Path	VARCHAR(4096)
IN	tableNames	VARCHAR
IN	separator	VARCHAR
IN	prefix	VARCHAR(255)
IN	overwrite	SMALLINT
IN	copyAnnotation	SMALLINT
IN	copySqlIndexes	SMALLINT
IN	copyForeignKeys	SMALLINT
IN	inDebug	CHAR(1)
OUT	numCreated	INTEGER
OUT	numUpdated	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Res_Source_Path	/shared/ASAssets/Utilities/generate/examples/ExampleProject /Physical/Metadata/postgres/ds_inventory/tutorial
IN	Res_Target_Path	/services/databases/ExampleProject/TestCatalog/tutorial
IN	tableNames	products
IN	separator	,
IN	prefix	V_
IN	overwrite	1
IN	copyAnnotation	1
IN	copySqlIndexes	0
IN	copyForeignKeys	0
IN	inDebug	Y
OUT	numCreated	1
OUT	numUpdated	0

/examples/generate

The examples provide background information on various view generation and deletion strategies. A summary of information is provided below. Detailed instructions can be found in the "_readme" script in the /generate/examples folder.

Summary

1. *pEx0_create_published_ds_group_and_user* - Create a default group [group1], user [user1], published data source [/services/databases/ExampleProject,] and assign privileges to [group1].

2. *pEx1_create_single_table_no_layers* - Generate Views for a single table with no layers [products]

3. *pEx1_destroy_single_table_prefix* - Destroy Views for a single table [products]

4. *pEx2_create_entire_schema_no_layers* - Generate Views for all tables with no layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

5. *pEx2_destroy_entire_schema* - Destroy Views for all tables [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

6. *pEx3_create_entire_schema_all_layers* - Generate Views for all tables through all layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

7. *pEx4_create_entire_schema_all_layers_with_fk* - Generate Views for all tables through all layers with foreign keys and annotations [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

8. *pEx4_destroy_entire_schema* - Destroy Views for all tables in all layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

9. *pEx5_create_multiple_tables_all_layers_prefix* - Generate Views for multiple tables through all layers with a prefix [categories,products]

10. *pEx4_destroy_entire_schema* - Destroy Views for all tables in all layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

11. *pEx6_create_entire_schema_all_layers_no_idx_fk* - Generate Views for all table through all layers with no annotations, indexe or foreign keys [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

12. *pEx4_destroy_entire_schema* - Destroy Views for all tables in all layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

13. *pEx7_create_multiple_tables_all_layers_no_prefix* - Generate Views for multiple tables through all layers with no prefix [categories,products]

14. *pEx4_destroy_entire_schema* - Destroy Views for all tables in all layers [categories, employees, inventorytransactions, products, purchaseorders, suppliers]

13 How To Use ‘Logging’ Procedures

Introduction

This section will show how to use the ‘Logging’ procedures.

auditLogger

Provides the developer with a way in which to programmatically send error messages to a File, Email, Database and/or Studio console.

The "attributeVect" parameter governs how the procedure will deal with messages. An example of declaring a variable to contain the attribute vector:

```
DECLARE attributeVect
      VECTOR(/shared/ASAssets/Utilities/log/auditLogger.AttributeType);
```

There are a number of attributes that can be specified:

Attribute	Description
debug	Y=debug or N=do not debug. Debugging within the auditLogger procedure only.
loggingType	The type of logging to perform. One or more of this list (comma or space separated): • LOG - Write to the <CIS_HOME>/logs/cs_server.log • EMAIL - Send an email (Email settings must be configured on the CIS instance in the Configuration panel.) • DB - Insert the message into the AUDIT_LOG table. (created and introspected by developer during initialization) • PRINT - Print to the Studio console tab
notificationType	The type of notification that is being logged. One and only one of: • ERROR - Output error message with severity level ERROR. • AUDIT - Output audit message with severity level INFO. • INFO - Output info message with severity level INFO. • DEBUG - Output debug message with severity level INFO.
auditTablePath	The CIS path to the AUDIT_LOG table. e.g. `/shared/Cache_DB/Cache_Repo/CACHE1/AUDIT_LOG`
sequenceNum	A sequence number (cast as a VARCHAR) used to correlate multiple messages across different log messages.

Attribute	Description
organizationName	The name of the organization which can be used to filter messages. e.g. Mortgage, Operations, CustomerSatisfaction.
applicationName	The application name that is using Composite within the organization which can be used as an additional filter. e.g. HomeLoans, Bankruptcy, etc.
origUserName	The original user name from the application: format=username@domain. e.g. user1@ldap or user2@composite
resourceName	The name of the resource being acted upon such as VIEW_NAME_INCR.
moduleName	The name of the module or procedure that is invoking the auditLoger which provides context for the code such as RefreshCache.
cachekey	The cachekey being used to refresh the cache or 0 if not applicable. This is especially helpful when invoked from incremental cache refresh scripts.
emailFrom	The address the message is from. NULL causes use of the server's configured "from" address. Only NULL is supported in this release.
emailReplyTo	The address to place in the replyTo field of the message.
emailTo	A comma separated list of e-mail addresses
emailCC	A comma separated list of e-mail addresses
emailBCC	A comma separated list of e-mail addresses
emailSubject	The message subject
emailContentType	This can be 'TEXT_PLAIN' or 'TEXT_HTML'

Parameters:

Direction	Parameter Name	Parameter Type
IN	messageText	LONGVARCHAR
IN	attributeVect	VECTOR[(VARCHAR(255), LONGVARCHAR)]

1. **Examples:**

1.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	messageText	'This is a test message'
IN	attributeVect	[('loggingType', 'LOG, DATABASE'),

Direction	Parameter Name	Parameter Value
		('notificationType', 'DEBUG'), ('auditTablePath', '/shared/myDB/audit_tab'), ('moduleName', '/shared/myTestProcedure') }

logDebugMessage

Provides the developer with a built in mechanism to log a message to the CIS log file whereby a single parameter can turn debugging on or off globally. Additionally the log message will contain the CIS module (procedure) where the message was registered. This allows a developer to locate the problem area more quickly.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	moduleName The name of the invoking procedure	/shared/ASAssets/Utilities/TypeDefinitions.moduleNameType
IN	debug Defines the debugging options for this procedure. Values: Y or T = debugging turned on, N or F = debugging turned off	CHAR(1)
IN	messageText The message to be to be sent or stored	/shared/ASAssets/Utilities/TypeDefinitions.messageType

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	moduleName	Any procedure name
IN	debug	Y/N or T/F
IN	messageText	Message text

LogUtils

This section will show how to use the 'Log' CJP procedures.

LogUtils/GetServerMetadataLog

Parses the local server's metadata change log files into rows and columns. Reads all metadata log files in the server's logs folder, oldest to newest. The result set is therefore naturally sorted in ascending order by change time.

NOTE: By default, CIS will only keep around 100Mb of change metadata logs (older log files will be deleted.) However, the file $CIS_HOME/conf/server/log4j.properties can be updated to store more data (requires CIS restart.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR (change_time TIMESTAMP, cid INTEGER, domain VARCHAR, user VARCHAR, userid INTEGER, hostname VARCHAR, operation VARCHAR, resource_id INTEGER, resource_path VARCHAR, resource_type VARCHAR, message VARCHAR)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	<result too large to display>

14 How To Use ‘Net’ Procedures

Introduction

This section will show how to use the ‘Net’ procedures.

NetUtils

This section will show how to use the ‘Net’ CJP procedures.

ftpFile

Connects to an FTP server and retrieves a file and places it into a folder on the CIS host filesystem.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fileName	LONGVARCHAR
IN	hostIp	LONGVARCHAR
IN	userId	LONGVARCHAR
IN	userPass	LONGVARCHAR
IN	ftpDirName	LONGVARCHAR
IN	dirName	LONGVARCHAR
OUT	success	BOOLEAN

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fileName	'test.xml'
IN	hostIp	'ftp.company.com'
IN	userId	'anonymous'
IN	userPass	'cgoodric@company.com'
IN	ftpDirName	'pub'
IN	dirName	'C:\Users\cgoodric\Desktop'
OUT	success	true

15 How To Use ‘PDTool’ Procedures

Introduction

This section describes the routines using the “PDTool” procedures.

generatePDToolDeployableResourcePlanByDate

This procedure forms the basis by which to generate PDTool incremental deployment plans based on a resource date. It provides the developers the control to create a deployment plan for only those resources that have changed based resources >= the passed in resource date. This procedure is only valid for CIS 6.2.2 and higher due to the fact that this release introduced a "lastModifiedDate" and "creationDate" in the metadata. Prior to this release their no date values to compare with.

This procedure will automatically insert "createFolder" actions to insure that the necessary folder path will exist when the resource is deployed from VCS. If the folder does not exist on the target server, an error would be thrown.

Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result. Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	planFilePath	VARCHAR(1024)
IN	bufferSize	INTEGER
IN	PDToolPlanTemplateVCS	LONGVARCHAR
IN	PDToolPlanTemplateFolders	LONGVARCHAR
IN	resourcePathList	LONGVARCHAR
IN	resourceTimestamp	TIMESTAMP
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	debug	CHAR(1)
OUT	success	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	planFilePath	'D:/PDTool62/resources/plans/examples.dp'
IN	bufferSize	100
IN	PDToolPlanTemplateVCS	'PASS TRUE ExecuteAction vcsCheckout2 $SERVERID $VCONN "$RESOURCE_PATH" "$RESOURCE_TYPE" HEAD "$MODULE_HOME/VCSModule.xml" "$MODULE_HOME/servers.xml"'
IN	PDToolPlanTemplateFolders	'PASS TRUE ExecuteAction createFolder $SERVERID "$RESOURCE_PATH" "$MODULE_HOME/servers.xml" true'
IN	resourcePathList	'/shared/examples, /services/databases/examples'
IN	resourceTimestamp	'2013-08-16 00:00:00'
IN	includeBasePath	NULL
IN	excludePathsList	NULL
IN	Debug	0
OUT	Success	1 (plan file is written to the path specified by planFilePath.)

generatePDToolDeployableResourcePlanByLineage

This procedure forms the basis by which to generate PDTool incremental deployment plans based on published resources. It provides the developers the control to create a deployment plan for only those resources that have changed based on a starting published resource. A published resource is any resource found in "/services/databases" or "/services/webservices".

This procedure will automatically insert "createFolder" actions to insure that the necessary folder path will exist when the resource is deployed from VCS. If the folder does not exist on the target server, an error would be thrown.

Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result. Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	planFilePath	VARCHAR(1024)
IN	bufferSize	INTEGER

Direction	Parameter Name	Parameter Type
IN	PDToolPlanTemplateVCS	LONGVARCHAR
IN	PDToolPlanTemplateFolders	LONGVARCHAR
IN	resourcePathList	LONGVARCHAR
IN	includeDependentTriggers	INTEGER
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	inIgnoreResourceDoesNotExist	INTEGER
IN	debug	CHAR(1)
OUT	success	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	planFilePath	'C:\PDTool62\resources\plans'
IN	bufferSize	100
IN	PDToolPlanTemplateVCS	'PASS TRUE ExecuteAction vcsCheckout2 $SERVERID $VCONN "$RESOURCE_PATH" "$RESOURCE_TYPE" HEAD "$MODULE_HOME/VCSModule.xml" "$MODULE_HOME/servers.xml"'
IN	PDToolPlanTemplateFolders	'PASS TRUE ExecuteAction createFolder $SERVERID "$RESOURCE_PATH" "$MODULE_HOME/servers.xml" true'
IN	resourcePathList	'/shared/examples, /services/databases/examples'
IN	includeDependentTriggers	1
IN	includeBasePath	NULL
IN	excludePathsList	NULL
IN	inIgnoreResourceDoesNotExist	1
IN	debug	0
OUT	success	1 (plan file is written to the path specified by planFilePath.)

template_generatePDToolDeployableResourcePlan

This procedure provides a template for invoking `generatePDToolDeployableResourcePlanByLineage` or `generatePDToolDeployableResourcePlanByDate`. This procedure should be copied to a working folder within the project and modified by the user for developers to use with ease.

Note that when using "generatePDToolDeployableResourcePlanByDate" to create a deployment plan based on a resource date can only be used with CIS 6.2.2 and higher. CIS 6.2.2 and higher introduced a "lastModifiedDate" and "creationDate" in the metadata. Prior to this release there are no date values to compare with.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	planOption	INTEGER
OUT	success	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	planOption	1
OUT	Success	1

helpers

This section describes the auxiliary procedures for the PDTool utilities.

helpers/getDeployableResourceListByDate

This procedure provides the logic for retrieving a complete list of paths based on the creation date or last modified date. If either the creation date or the last modified date is not null and is greater than the passed in resource timestamp then it appears in the result. Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result. Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePathList	LONGVARCHAR
IN	resourceTimestamp	TIMESTAMP

Direction	Parameter Name	Parameter Type
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	debug	INTEGER
OUT	resourceTreeList	CURSOR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePathList	'/shared/examples'
IN	resourceTimestamp	'2013-08-16 00:00:00'
IN	includeBasePath	NULL
IN	excludePathsList	NULL
IN	debug	0
OUT	resourceTreeList	<row set>

helpers/getDeployableResourceListByLineage

This procedure provides the logic for retrieving a complete list of paths based on the lineage of all the resources specified by the input parameter "resourcePathList". Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result. Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePathList	LONGVARCHAR
IN	includeDependentTriggers	INTEGER
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	debug	INTEGER
IN	inIgnoreResourceDoesNotExist	INTEGER
OUT	resourceTreeList	CURSOR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePathList	'/shared/examples'
IN	includeDependentTriggers	1
IN	includeBasePath	NULL
IN	excludePathsList	NULL
IN	debug	0
IN	inIgnoreResourceDoesNotExist	1
OUT	resourceTreeList	<row set>

helpers/getDistinctDeployableResourceListByDate

This procedure provides the logic for retrieving a distinct list of paths. A resource may be shared with other resources but it only needs to be deployed once. This retrieves deployable resources based on the creation date or last modified date. If either the creation date or the last modified date is not null and is greater than the passed in resource timestamp then it appears in the result. Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result. Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePathList	LONGVARCHAR
IN	resourceTimestamp	TIMESTAMP
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	debug	INTEGER
OUT	resourceTreeList	CURSOR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePathList	'/shared/examples'
IN	resourceTimestamp	'2013-08-16 00:00:00'
IN	includeBasePath	NULL
IN	excludePathsList	NULL

Direction	Parameter Name	Parameter Value
IN	debug	0
OUT	resourceTreeList	<row set>

helpers/getDistinctDeployableResourceListByLineage

This procedure provides the logic for retrieving a distinct list of paths. A resource may be shared with other resources but it only needs to be deployed once. This retrieves deployable resources based on lineage of the resource paths. If include triggers is set then any triggers associated associated with a resource are also returned. Any resource paths found in the exclude paths list "excludePathsList" will be excluded from the result.   Any resources that are not inclusive of the include base paths "includeBasePath" will be excluded from the result.

PDTool (Promotion and Deployment Tool) is used to execute generated plan files.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePathList	LONGVARCHAR
IN	includeDependentTriggers	INTEGER
IN	includeBasePath	LONGVARCHAR
IN	excludePathsList	LONGVARCHAR
IN	inIgnoreResourceDoesNotExist	INTEGER
IN	debug	INTEGER
OUT	resourceTreeList	CURSOR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePathList	'/shared/examples'
IN	includeDependentTriggers	1
IN	includeBasePath	NULL
IN	excludePathsList	NULL
IN	inIgnoreResourceDoesNotExist	1
IN	debug	0
OUT	resourceTreeList	<row set>

16 How To Use 'Repository' Procedures

Introduction

This section will show how to use the 'CIS Repository' API procedures. These numerous helper procedures are built around the CIS repository API to provide a higher-level of abstraction from the raw API's. This helps to increase developer productivity.

CIS Types and Subtypes listing

This section provides a list of CIS repository resource types and subtypes. Most of these can also be found as constants in the `/lib/resource/ResourceDefs` SQL definition set. These constants' names all start with either "RESOURCE_TYPE_" or "RESOURCE_SUBTYPE_".

Listing of CIS resource types and subtypes

The following resource types/subtypes are supported by several repository helper procedures. The format below is RESOURCE TYPE / SUB TYPE – Description.

(Basic CIS folder)

- CONTAINER / FOLDER_CONTAINER - A Composite folder. Cannot be created anywhere under /services except in another FOLDER under /services/webservices.

(Database)

- CONTAINER / CATALOG_CONTAINER - A Composite catalog folder under a data source. Can only be created within a data source.
- CONTAINER / SCHEMA_CONTAINER - A Composite schema container. Can only be created within a CATALOG_CONTAINER or data source.

(Web Services)

- CONTAINER / SERVICE_CONTAINER - A web service container for the service. Can only be created within a Composite Web Services data source that is under /services/webservices.
- CONTAINER / OPERATIONS_CONTAINER - A web service container for the operations
- CONTAINER / PORT_CONTAINER - A Composite web service container for port. Can only be created within a SERVICE under /services/webservices.

(Misc Containers)

- CONTAINER / CONNECTOR_CONTAINER - A Composite container for connectors.
- CONTAINER / DIRECTORY_CONTAINER - A folder within an LDAP data source.

(Connectors)

- CONNECTOR / JMS - A Composite JMS Connector. Created with no connection information
- CONNECTOR / HTTP - A Composite HTTP Connector. Created with no connection information

(Data Sources)

- DATA_SOURCE / RELATIONAL_DATA_SOURCE - A relational database source.

- DATA_SOURCE / FILE_DATA_SOURCE - A comma separate file data source.
- DATA_SOURCE / XML_FILE_DATA_SOURCE - An XML file data source.
- DATA_SOURCE / COMPOSITE_WEB_SERVICE - A published web service.
- DATA_SOURCE / WSDL_DATA_SOURCE - A Composite web service data source or published legacy web service.
- DATA_SOURCE / XML_HTTP_DATA_SOURCE - A simple XML over HTTP data source.
- DATA_SOURCE / REST_DATA_SOURCE - A REST data source.
- DATA_SOURCE / NONE - A custom java procedure data source.

(Definition Sets)
- DEFINITION_SET / SQL_DEFINITION_SET - A Composite SQL Definition set.
- DEFINITION_SET / XML_SCHEMA_DEFINITION_SET - A Composite XML Schema Definition set.
- DEFINITION_SET / WSDL_DEFINITION_SET - A Composite WSDL Definition set.
- DEFINITION_SET / ABSTRACT_WSDL_DEFINITION_SET - A Composite Abstract WSDL Definition set such as the ones imported from Designer.
- DEFINITION_SET / SCDL_DEFINITION_SET - A Composite SCA composite Definition set imported from Designer.

(Published Resources)
- LINK / NONE – A resource published in /services.

(Model Resources)
- MODEL / NONE – A Discovery model.

(Policies)
- POLICY / CACHE_POLICY - A Composite cache refresh policy. Created disabled.
- POLICY / NONE - A custom web services security policy.

(CIS procedures)
- PROCEDURE / SQL_SCRIPT_PROCEDURE - A Composite SQL Procedure. Created with a simple default script body that is runnable.

(Custom java procedures)
- PROCEDURE / JAVA_PROCEDURE - A Composite java data source procedure. Created from a java data source (jar file).

(Database procedures)
- PROCEDURE / EXTERNAL_SQL_PROCEDURE - A Composite Packaged Query. Created with no SQL text, so it is not runnable.
- PROCEDURE / DATABASE_PROCEDURE - A database stored procedure.
- PROCEDURE / NATIVE_FUNCTION - A database function.

(XML procedures)

- PROCEDURE / BASIC_TRANSFORM_PROCEDURE - A Composite Basic XSLT Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
- PROCEDURE / TRANSFORM_PROCEDURE - A Composite Transformation Editor (Any to Any) procedure. Created with no target model or output definition so is not runnable.
- PROCEDURE / XSLT_ PROCEDURE - A Composite XSLT procedure. Created with no target procedure, so it is not runnable.
- PROCEDURE / XSLT_TRANSFORM_PROCEDURE - A Composite XSLT Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
- PROCEDURE / STREAM_TRANSFORM_PROCEDURE - A Composite XSLT Streaming Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
- PROCEDURE / XQUERY_ PROCEDURE - A Composite XQUERY procedure. Created with no target procedure, so it is not runnable.
- PROCEDURE / XQUERY_TRANSFORM_PROCEDURE - A Composite XQUERY Transformation Procedure. Created with no target schema and no model, so it is not runnable.

(Misc procedures)

- PROCEDURE / OPERATION_PROCEDURE - A Composite web service or HTTP procedure operation.

(Relationship Resources)

- RELATIONSHIP / NONE – A Discovery relationship.

(Tables or Views)

- TABLE / SQL_TABLE - A Composite View. Created with no SQL text or model, so it is not runnable.
- TABLE / DATABASE_TABLE - A data source table or view.
- TABLE / DELIMITED_FILE_TABLE - A delimited file data source "table".
- TABLE / EXCEL_NON_ODBC_POI - An Excel (non-ODBC) worksheet "table".
- TABLE / SYSTEM_TABLE - A Composite system table view. Cannot be created or modified.

(XML Structures)

- TREE / XML_FILE_TREE - The XML tree structure associated with a file-XML data source.

(Triggers)

- TRIGGER / NONE - A Composite trigger. Created disabled.

The ACCESS_TOOLS Right

Because the Repository Helper procedures make use of the published Admin web services API (see /services/webservices/system/admin), users that need to make use of the Repository Helper procedures will need to have the ACCESS_TOOLS right. A security exception will be thrown otherwise.

Note On Using Repository Helper Procedures With Triggers and Cache Procedures

CIS does not by default support **updating** the repository with a task session (i.e. a triggered procedure or pre/post cache procedure.) Any task session that attempts to update the repository (create a resource, update a resource’s attributes, move a resource, etc.) will cause an exception to be thrown. Triggered processes should be able to use Repository Helper procedures to **read** from the repository without any problems (assuming the task session's owner has the ACCESS_TOOLS right as mentioned above.)

In 6.2, the ability to update the repository from a task session was supported as a configuration setting that needs to be explicitly enabled. See Server -> Configuration -> Security -> Enable Task Session API Access in the Studio Configuration panel.

CIS Repository Helper Procedures

This section describes each 'CIS Repository' API helper procedure. It provides the name of the procedure and a description of what it does. It provides a chart of the parameters that define the direction of the parameter, the parameter name and the type. Finally, a chart with sample parameter is data is provided to give the user an idea of what is expected.

addRemoveDataSourceChildren

This procedure performs the same function as the "Add/Remove Resources ..." context menu item for a data source. It takes as input a data source path (i.e. '/shared/examples/ds_orders') and a list of child resources (the child path specification is an absolute path that begins at the data source container) and/or containers and their introspection state (one of 'SELF', 'SELF_AND_CHILDREN', or 'IGNORED'.) This procedure has been deprecated in CIS 6.2. See introspectResourcesResultCursor

This script is used to extract the introspection result text into a cursor of results. The input to this procedure is the output variable [introspectionResult LONGVARCHAR] from the procedure introspectResources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	introspectionResult	LONGVARCHAR
OUT	result	CURSOR (totalResults INTEGER, completed BIT, status VARCHAR(32768), introspectorVersion INTEGER, startTime TIMESTAMP, endTime TIMESTAMP, addedCount INTEGER, removedCount INTEGER, updatedCount INTEGER, skippedCount INTEGER, totalCompletedCount INTEGER, toBeAddedCount INTEGER, toBeRemovedCount INTEGER, toBeUpdatedCount INTEGER, totalToBeCompletedCount INTEGER, warningCount INTEGER,

Direction	Parameter Name	Parameter Type
		errorCount INTEGER, "path" VARCHAR(4096), "type" VARCHAR(40), subtype VARCHAR(40), "action" VARCHAR(32768), durationMs INTEGER, entryStatus VARCHAR(32768), code VARCHAR(32768), name VARCHAR(32768), message LONGVARCHAR, detail VARCHAR(32768), severity VARCHAR(32768))

lowerLevelProcedures/introspectResourcesResult

This procedure was moved to lowerLevelProcedures in release 2018Q1 as it is now a supportive procedure for “introspectResources”. This procedure gathers the results from a call to introspectResourcesTask. If the introspection task is still running, the procedure can be called in such a way as to block execution until the task completes before returning results.

The result output will contain a cursor of rows if the introspection is successful. The cursor of rows will provide information on catalogs, schemas and tables. Otherwise, the additional result rows will indicate what went wrong.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
IN	taskId	LONGVARCHAR
IN	block	BIT
IN	pageSize	INTEGER
IN	pageStart	INTEGER
OUT	result	CURSOR (totalResults INTEGER, completed BIT, status VARCHAR(32768), introspectorVersion INTEGER, startTime TIMESTAMP, endTime TIMESTAMP, addedCount INTEGER, removedCount INTEGER, updatedCount INTEGER, skippedCount INTEGER,

Direction	Parameter Name	Parameter Type
		totalCompletedCount INTEGER, toBeAddedCount INTEGER, toBeRemovedCount INTEGER, toBeUpdatedCount INTEGER, totalToBeCompletedCount INTEGER, warningCount INTEGER, errorCount INTEGER, "path" VARCHAR(4096), "type" VARCHAR(40), subtype VARCHAR(40), "action" VARCHAR(32768), durationMs INTEGER, entryStatus VARCHAR(32768), code VARCHAR(32768), name VARCHAR(32768), message LONGVARCHAR, detail VARCHAR(32768), severity VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
IN	taskId	'1000'
IN	block	1
IN	pageSize	NULL
IN	pageStart	NULL
OUT	result	(result too large to display)

lowerLevelProcedures/introspectResourcesTask below.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inDataSourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inChildInfos	/shared/ASAssets/Utilities/TypeDefinitions. ChildInfosVectorType
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inDataSourcePath	'/shared/examples/ds_orders'
IN	inChildInfos	VECTOR [('/customers', 'TABLE', 'IGNORED'), ('/orderdetails', 'TABLE', 'IGNORED')]
OUT	success	1
OUT	faultResponse	[NULL]

applyReservedListToPath (Custom Function)

This procedure has been deprecated. See RepoUtils/applyReservedListToPath below.

applyReservedListToWord (Custom Function)

This procedure has been deprecated. See RepoUtils/applyReservedListToWord below.

configureReservedList

This procedure has been deprecated. It has been functionally replaced with the properties file found on the CIS host filesystem at `$CIS_HOME/conf/customjars/RepoUtils.properties`.

cachedResources

This procedure is used to manipulate cached resources within a starting folder. This procedure can retrieve, enable, or disable cached resources within a designated folder. It operates recursively.

Input:

operation - R=retrieve, E=enable caches, D=disable caches. The operation acts upon all resources found in the path where caching is configured and the includePathList_ and excludePathList_ filters are applied.

startingPath - The path to recursively start searching

includePathList - A comma separated list of paths or partial paths to include as filters (only execute on these paths). A partial path only has to be present anywhere within the path, not just the beginning of the path.

e.g. startingPath=/shared/project, includePathList=/F2

Searched paths would include:

- /shared/project/F1/F2

- /shared/project/F2/F2

- /shared/project/F3/F2

but not:

- /shared/project/F4/F1

excludePathList - A comma separated list of paths or partial paths to exclude from the list (do not execute on these paths). This works in a similar manner to includePathList.

debug - Y=debug is on, N=do not debug

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	operation	VARCHAR(255)
IN	startingPath	/lib/resource/ResourceDefs.ResourcePath
IN	includePathList	LONGVARCHAR
IN	excludePathList	LONGVARCHAR
IN	debug	CHAR(1)
OUT	result	CURSOR (operation VARCHAR(255) prevStatus VARCHAR(255) currStatus VARCHAR(255) resourceType ResourceType resourcePath ResourcePath)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	operation	‘R’
IN	startingPath	‘/shared/examples’
IN	includePathList	NULL
IN	excludePathList	NULL
IN	debug	‘N’
OUT	result	(‘R’,

Direction	Parameter Name	Parameter Value
		‘ENABLED’, ‘ENABLED’, ‘TABLE’, ‘/shared/examples/ds_orders/orders’)

changePassword

If published, this procedure allows a “composite” domain user logged in from an external client to change his/her password programmatically. If a non-composite domain user attempts to use this procedure or if the newPassword and confirmNewPassword arguments don’t match, an IllegalArgumentException will be thrown.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	oldPassword	VARCHAR(255)
IN	newPassword	VARCHAR(255)
IN	confirmNewPassword	VARCHAR(255)
OUT	result	VARCHAR(255)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	oldPassword	‘0ld p4ssw0rd’
IN	newPassword	‘n3w p4ssw0rd’
IN	confirmNewPassword	‘n3w p4ssw0rd’
OUT	result	‘Password successfully updated.’

compareCisVersions (Custom Function)

This method compares two CIS version (baseline and current). It converts the version string to an integer and performs a comparison. The following is returned based on the comparison:

-1 - if the current version is less than the baseline version.

0 - if the current version is equal to the baseline version.

1 - if the current version is greater than the baseline version.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	baseCisVersion	VARCHAR
IN	currentCisVersion	VARCHAR
OUT	status	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	baseCisVersion	'6.1.0.01.09'
IN	currentCisVersion	'6.1.0.01.14'
OUT	status	1

copyResources

This procedure is used to copy all of the CIS resources from a source folder to a target folder. If the target folder does not exist, then it is created. An exception is thrown if the source does not exist.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourceFolderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	targetFolderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourceFolderPath	'/shared/ASAssets/Utilities/repository/examples/source'
IN	targetFolderPath	'/shared/ASAssets/Utilities/repository/examples/target'
OUT	success	1
OUT	faultResponse	null

copyResourceAnnotations

This procedure is used to copy all of the annotations of one resource to another. If both resources are of type "TABLE", the column annotations are copied as well (where the column names are the same, ignoring case.) This procedure is NOT recursive when resources of type "CONTAINER" are specified as input. There are a couple of Utilities that can walk a resource or dependency tree and can be used in conjunction with this procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inSourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inSourceType	/lib/resource/ResourceDefs.ResourceType
IN	inDestPath	/lib/resource/ResourceDefs.ResourcePath
IN	inDestType	/lib/resource/ResourceDefs.ResourceType

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSourcePath	'/shared/examples/ds_orders/orders'
IN	inSourceType	'TABLE'
IN	inDestPath	'/shared/examples/CompositeView'
IN	inDestType	'TABLE'

copyResourcesPrivileges

This is a procedure is used to copy resource privileges from one resource to another.

This procedure enables changes to resource privileges for users and groups, by copying privileges from other resources. Changes can be made to one or more resources with different source resource for one or many users and groups. Resource privileges can be set for a specified set of users and groups without modifying any existing privileges for other users and groups, or the procedure can set resource privileges restrictively to only privileges of source resource explicitly.

Only a user with GRANT privilege on a resource can modify the privileges for that resource. The owner of a resource always has GRANT privilege, as do users with the MODIFY_ALL_RESOURCES right.

When "mode" is "OVERWRITE_APPEND", or is not supplied, privileges are applied on a per-user or per-group basis, so that updating privileges for one user or group does not alter privileges from any other user or group. The privileges applied for a user or group replace the previous value for that user or group.

When "mode" is "SET_EXACTLY", all privileges on the resource are made to look exactly like the privileges of source resource.

When "updateRecursively" is "false", the privileges are applied only to the specified resources. When it is "true", the privileges are recursively applied into any CONTAINER or DATA_SOURCE resource specified. When recursively applying privileges, the privilege change is ignored for any resource the user lacks owner privileges for.

Privileges that are not applicable for a given resource type are automatically stripped down to the set that is legal for each resource. TABLE resources support NONE, READ, WRITE, SELECT, INSERT, UPDATE, and DELETE. PROCEDURE resources support NONE, READ, WRITE, and EXECUTE. All other resource types only support NONE, READ, and WRITE.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	updateRecursively 0 (false) or null - only update the given resource and not the children. 1 (true) - update children recursively	BIT
IN	copyPrivilegeMode null (default) - do not set any privileges at all 0 - set mode to "OVERWRITE_APPEND" - merges and does not update privileges for users or groups not mentioned. 1 - set the mode to "SET_EXACTLY" - makes privileges look exactly like those provided in the call.	INTEGER
IN	copyPrivilegeEntry	RepositoryDefinitions.copyPrivilegeEntryVector
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	updateRecursively	1
IN	copyPrivilegeMode	0

Direction	Parameter Name	Parameter Value
IN	copyPrivilegeEntry	VECTOR[{(‘/shared/test00’,’CONTAINER’)}, {(‘/shared/test01’,’CONTAINER’)}]
OUT	success	1
OUT	faultResponse	null

createAllFolders

This procedure is used to create all the CIS folders designated by the incoming folder path variable.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourceFolderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourceFolderPath	‘/shared/ASAssets/Utilities/repository/examples/target/newfolder’
OUT	success	1
OUT	faultResponse	null

createAllFoldersPrivileges

This procedure is used to create all the CIS folders designated by the incoming folder path variable and copy privileges from the parent folder. The parent folder is determined to be the last folder encountered in the path that exists. All other folders beyond the existing folder ar considered children and must be created.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourceFolderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	updatePrivilegesRecursively 0 (false) or null - only update privileges for the given resource and not the children.	BIT

Direction	Parameter Name	Parameter Type
	1 (true) - update privileges for children recursively	
IN	copyPrivilegeMode null (default) - do not set any privileges at all 0 - set mode to "OVERWRITE_APPEND" - merges and does not update privileges for users or groups not mentioned. 1 - set the mode to "SET_EXACTLY" - makes privileges look exactly like those provided in the call.	BIT
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourceFolderPath	'/shared/ASAssets/Utilities/repository/examples/target/newfolder'
IN	updatePrivilegesRecursively	0
IN	copyPrivilegeMode	0
OUT	success	1
OUT	faultResponse	null

createConnector

This procedure creates a JMS connector

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	name	VARCHAR(100)
IN	groupName	VARCHAR(100)

Direction	Parameter Name	Parameter Type
IN	jmsClientID	VARCHAR(1024)
IN	annotation	VARCHAR(1024)
IN	jndiContextFactory	VARCHAR(1024)
IN	jndiProperties	LONGVARCHAR
IN	jndiProviderUrl	VARCHAR(1024)
IN	jndiUser	VARCHAR(50)
IN	jndiPassword	VARCHAR(50)
IN	queueConnectionFactory	VARCHAR(1024)
IN	minPool	INTEGER
IN	maxPool	INTEGER
IN	poolTimeout	INTEGER

2. Examples:

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	name	'myMQ'
IN	groupName	'<Group Name>'
IN	jmsClientID	'<JMS Client ID>'
IN	annotation	'This is a JMS message queue'
IN	jndiContextFactory	'<JNDI context factory>'
IN	jndiProperties	'<JNDI Properties XML>'
IN	jndiProviderUrl	'<JNDI Provider URL>'
IN	jndiUser	'myMQuser'
IN	jndiPassword	'myMQpassword'
IN	queueConnectionFactory	'<Queue Connection Factory>'
IN	minPool	1
IN	maxPool	10
IN	poolTimeout	300

createDataSource

This procedure creates an empty data source. See chart below for valid data source types and the folders where they can be created. As of 5.2.0.02.27 some data source types had issues when attempting to create data sources for them. They are also noted in the chart below.

After creation, `repository/updateResourceDataSource` must be used to add connectivity information. Then `repository/addRemoveDataSourceChildren` must be used to introspect resources.

Data Source Type	Valid Location	Notes
'COMPOSITE_DATABASE'	/services/Databases	
'COMPOSITE_SERVICE'	/services/Web Services	
'Custom Java Procedure'	/shared/...	
'DB2 v7 (Type 2)'	/shared/...	
'DB2 v7 (Type 4)'	/shared/...	
'DB2 v8 (Type 2)'	/shared/...	
'DB2 v8 (Type 4)'	/shared/...	
'DB2 v9 (Type 2)'	/shared/...	
'DB2 v9 (Type 4)'	/shared/...	
'DB2 z/OS v8 (Type 4)'	/shared/...	
'DataDirect Mainframe'	/shared/...	Requires DataDirect Mainframe connector software from DataDirect
'Essbase'	/shared/...	Requires appropriate CAC software and license
'File-Cache'	/shared/...	
'File-Delimited'	/shared/...	
'File-XML'	/shared/...	
'Greenplum 3.3'	/shared/...	
'Informix 9.x'	/shared/...	This one causes a problem when used: java.lang.String cannot be cast to java.lang.Integer
'LDAP'	/shared/...	
'Microsoft Access'	/shared/...	
'Microsoft Excel'	/shared/...	
'Microsoft Excel (non-ODBC)'	/shared/...	
'Microsoft SQL Server 2000'	/shared/...	

Data Source Type	Valid Location	Notes
'Microsoft SQL Server 2005'	/shared/...	
'Microsoft SQL Server 2008'	/shared/...	
'MySQL 4.0'	/shared/...	
'MySQL 5.0'	/shared/...	
'NeoView 2.3'	/shared/...	
'NeoView 2.4'	/shared/...	
'Netezza 3.0'	/shared/...	This one causes a problem when used: java.lang.String cannot be cast to java.lang.Integer
'Netezza 4.5'	/shared/...	This one causes a problem when used: java.lang.String cannot be cast to java.lang.Integer
'Netezza 5.0'	/shared/...	This one causes a problem when used: java.lang.String cannot be cast to java.lang.Integer
'Oracle 10g (OCI Driver)'	/shared/...	
'Oracle 10g (Thin Driver)'	/shared/...	
'Oracle 11g (OCI Driver)'	/shared/...	
'Oracle 11g (Thin Driver)'	/shared/...	
'Oracle 8i (OCI Driver)'	/shared/...	
'Oracle 8i (Thin Driver)'	/shared/...	
'Oracle 9i (OCI Driver)'	/shared/...	
'Oracle 9i (Thin Driver)'	/shared/...	
'Oracle E-Business Suite on 10g'	/shared/...	Requires appropriate CAC software and license
'Oracle E-Business Suite on 8i'	/shared/...	Requires appropriate CAC software and license
'Oracle E-Business Suite on 9i'	/shared/...	Requires appropriate CAC software and license
'Salesforce.com'	/shared/...	Requires appropriate CAC software and license
'SAP'	/shared/...	Requires appropriate CAC software and license
'SAP BW'	/shared/...	Requires appropriate CAC software and license

Data Source Type	Valid Location	Notes
'Siebel'	/shared/...	Requires appropriate CAC software and license
'Sybase'	/shared/...	
'Sybase IQ'	/shared/...	
'System'	/shared/...	
'Teradata'	/shared/...	
'Teradata 12'	/shared/...	
'Teradata 13'	/shared/...	
'XML/HTTP'	/shared/...	

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dataSourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	dataSourceName	/lib/resource/ResourceDefs.ResourceName
IN	dataSourceType	/lib/resource/ResourceDefs.ResourceType
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dataSourcePath	'/shared'
IN	dataSourceName	'DS_Oracle11g'
IN	dataSourceType	'Oracle 11g (Thin Driver)'
OUT	success	1
OUT	createResponse	XML
OUT	faultResponse	null

createFolder

This procedure creates a single folder at the end of the folder path. All other intermediate folders must exist. For example if `/shared/intermediate/folder1` is to be created, then `/shared` and `/shared/intermediate` must exist for this procedure to work properly. Use the `repository/createAllFolders` procedure to create all of the entire folder path structure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourceFolderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourceFolderPath	'/shared/ASAssets/Utilities/repository/examples/source'
OUT	success	1
OUT	faultResponse	null

createOrUpdateConnector

This procedure creates or updates a JMS connector

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	forceNullUpdate	SMALLINT
IN	name	VARCHAR(100)
IN	groupName	VARCHAR(100)
IN	jmsClientID	VARCHAR(1024)
IN	annotation	VARCHAR(1024)
IN	jndiContextFactory	VARCHAR(1024)
IN	jndiProperties	LONGVARCHAR
IN	jndiProviderUrl	VARCHAR(1024)
IN	jndiUser	VARCHAR(50)
IN	jndiPassword	VARCHAR(50)
IN	queueConnectionFactory	VARCHAR(1024)

Direction	Parameter Name	Parameter Type
IN	minPool	INTEGER
IN	maxPool	INTEGER
IN	poolTimeout	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	forceNullUpdate	0
IN	name	'myMQ'
IN	groupName	'<Group Name>'
IN	jmsClientID	'<JMS Client ID>'
IN	annotation	'This is a JMS message queue'
IN	jndiContextFactory	'<JNDI context factory>'
IN	jndiProperties	'<JNDI Properties XML>'
IN	jndiProviderUrl	'<JNDI Provider URL>'
IN	jndiUser	'myMQuser'
IN	jndiPassword	'myMQpassword'
IN	queueConnectionFactory	'<Queue Connection Factory>'
IN	minPool	1
IN	maxPool	10
IN	poolTimeout	300

createResource

Create a resource in a default state. The actual content of the resource is not provided here. This procedure only handles the initial creation. There are other procedures to update the resource with its content such as the following:

- `repository/updateDefSetDef`
- `repository/updateResourceCacheConfig`
- `repository/updateResourceCacheConfiguration`
- `repository/updateResourceCacheEnabled`

- `repository/updateResourceDataSource`
- `repository/updateResourcesSqlTable`
- `repository/updateSqlScript`
- `repository/updateSqlTable`
- `repository/updateTrigger`

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceName	VARCHAR(255)
IN	resourceType	VARCHAR(255)
IN	resourceSubType	VARCHAR(255)
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/ASAssets/Utilities/repository/examples/target'
IN	resourceName	'PRODUCT_VIEW'
IN	resourceType	'TABLE'
IN	resourceSubType	'SQL_TABLE'
OUT	success	1
OUT	createResponse	Create Response XML: <resource>
OUT	faultResponse	null

createResourceCopy

This is a procedure is used to create a copy of a resource by adding _Copy_1 and etc. It looks for existing copies and increments the number. The resource can be any resource including folders.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	mode C=Copy resource and leave original, R=Rename resource to the "copied" name. If left null, the default is copy and leave original in place	CHAR(1)
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR
OUT	success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	mode	C
IN	resourcePath	'/shared/ASAssets/Utilities/repository/examples/target/newfolder'
IN	resourceType	'CONTAINER'
OUT	success	1

createUnionView

This procedure creates a union view composed of the columns of two other views. The two views do not necessarily need to have the same columns (though logically, there should at least be one column in common.) The procedure will substitute NULLs for columns that don't exist in a particular view (cast to the data type of the other view's column.) If a column is common across both views but either the case of the column name or the data types differ, a flag is used to indicate which view's column definition to give preference to. The "AllIndicator" input specifies whether the view should be created as UNION (0) or UNION ALL (1).

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	UnionViewPath	/lib/resource/ResourceDefs.ResourcePath
IN	View1Path	/lib/resource/ResourceDefs.ResourcePath
IN	View2Path	/lib/resource/ResourceDefs.ResourcePath
IN	NameTypeConflictPreference	INTEGER

Direction	Parameter Name	Parameter Type
IN	AllIndicator	BIT
OUT	success	BIT
OUT	responseXML	XML
OUT	faultResponse	XML

2. **Examples:**

   2.1. **Assumptions: A view has been wrapped around /shared/examples/productCatalog_Transformation called "productCatalog_wrapper"**

Direction	Parameter Name	Parameter Value
IN	UnionViewPath	'/shared/examples/products_union'
IN	View1Path	'/shared/examples/ds_inventory/products'
IN	View2Path	'/shared/examples/productCatalog_wrapper'
IN	NameTypeConflictPreference	2
IN	AllIndicator	1
OUT	success	1
OUT	responseXML	(response XML)
OUT	faultResponse	NULL

deleteAllConnectors

This procedure deletes all the configured JMS connectors.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'

deleteConnector

This procedure deletes a configured JMS connector.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	CONN_NAME	VARCHAR(100)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	CONN_NAME	'myMQ'

destroyResource

This procedure is used to destroy/delete a resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceName	VARCHAR(255)
IN	resourceType	VARCHAR(255)
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	‘/shared/ASAssets/Utilities/repository/examples/target’
IN	resourceName	‘PRODUCT_VIEW’
IN	resourceType	‘TABLE’
OUT	success	1
OUT	createResponse	Create Response XML: <resource>
OUT	faultResponse	null

expireProcCacheEntryByName

The expireProcCacheEntryByName procedure can be used for any procedural cache to clear a single entry per call for a given set of parameters. This allows the selective expiration of part of a procedural cache's contents without completely expiring the entire cache. After expiring the results associated with a set of parameters, the Composite server will re-execute the procedure for the given set of parameters and will cache the newly calculated result.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	cachedResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	cacheStatusPath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	params	VARCHAR(255)
OUT	isSuccessful	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	cachedResourcePath	'/shared/examples/LookupProduct'
IN	cacheStatusPath	'/shared/examples/ds_orders/cache_status'
IN	params	"1" (single quoted 1)
OUT	isSuccessful	1

exportResourceDefinitions

This procedure generates a complete export of the definitions for all resources defined in CIS under shared and its subdirectories.

A complete export of the definitions for all view and procedure resources contained under the folder `/shared` are written under the starting folder indicated by the input parameter `outputDirectory` in a folder structure on the CIS host that matches the structure of the namespace tree.

Resource definitions are exported to files with a name matching the exported resource. The file extension of the output file is set by the input parameter `outputFileExtension`.

Directories and files are written using the credentials of the account that the CIS server is running under. Attempts to export resource definitions to directories that the CIS app account does not have sufficient rights to read and write to will fail.

Please note that the underlying CIS system tables used by this procedure require the ACCESS_TOOLS right to query, so this procedure will not work successfully if the invoking user does not have ACCESS_TOOLS. Only resources that the user has the READ privilege on will be exported.

Also note that this view will not expose the java code for CJP procedures. The author must provide Java source code.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	outputDirectory	LONGVARCHAR
IN	outputFileExtension	VARCHAR(10)
OUT	result	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	outputDirectory	'C:\Tmp'
IN	outputFileExtension	'.sql'
OUT	result	'SUCCESS'

exportResourcePrivileges

This procedure generates a complete export of the privileges for all the specified resources. An optional filter can be specified so that only those privileges that were explicitly set (and not inherited through a global right or group membership) are exported.

The generated export is written to the CIS host filesystem in XML format. This file can be modified (to apply new privileges to existing resources) or left alone (to re-apply existing privilege settings) and used by the `resources/importResourcePrivileges()` procedure.

The **in_path_list** parameter specifies the resources paths and types of the resource privileges to export. It is composed of a comma separated list of colon separated resource path and type pairs. For example:

```
/shared/examples:CONTAINER,
/services/databases/examples:DATA_SOURCE
```

The **Constants** section of the resource `/lib/resources/ResourceDefs` lists the possible values of resource types.

The **in_filter** input parameter can be used to specify whether all privileges should be exported (use either NULL or an empty string) or just those that were explicitly set (use the string 'ALL_EXPLICIT'.)

If column level security is being used on any views whose privileges are being exported, set the **in_includeColumns** parameter to 1 to preserve column level privilege settings. Otherwise set the parameter to 0 to conserve space.

The procedure does not output any parameters. If the procedure exits without throwing an error, then the export operation was a success.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	in_path_list	LONGVARCHAR
IN	in_filter	VARCHAR(255)
IN	in_includeColumns	BIT
IN	in_filename	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	in_path_list	'/shared/examples:CONTAINER, /services/databases/examples:DATA_SOURCE'
IN	in_filter	'ALL_EXPLICIT'
IN	in_includeColumns	0
IN	in_filename	'C:\cis_examples_privileges.xml'

fixLeadingCharactersInFolderPath (Custom Function)

This procedure is used to fix the leading characters in a folder path. Any path that contains a leading underscore '_' or a number '0123456789' must have double quotes inserted around that portion of the folder. This procedure would be called in conjunction with other procedures. For example, when generating a view based off of another view, the SELECT statement's FROM clause would require that the path to the underlying view be fixed with double quotes if it finds.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	fixedResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/ASAssets/Utilities/repository/examples/1folder/_folder'
OUT	fixedResourcePath	'/shared/ASAssets/Utilities/repository/examples/"1folder"/"_folder"'

getAllDataSourceChildren

This procedure returns all the children for a given DATA_SOURCE type resources found under the starting path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	includeColumns	BIT
OUT	childResCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(1024), resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, isNullable VARCHAR(255), columnName VARCHAR(255), columnType VARCHAR(255), nativeBaseType VARCHAR(255), nativeType VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
IN	includeColumns	1
OUT	childResCursor	(result too large to display)

getAllDataSources

This procedure returns all the DATA_SOURCE type resources found under the starting path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	resourceTreeList	CURSOR (name VARCHAR, resPath TypeDefinitions.pathType, resType VARCHAR, subType VARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
OUT	resourceTreeList	See chart below

2.2. Chart showing example output for resourceTreeList:

name	resPath	resType	subType
ds_inventory	/shared/examples/ds_inventory	DATA_SOURCE	RELATIONAL_DATA_SOURCE
ds_orders	/shared/examples/ds_orders	DATA_SOURCE	RELATIONAL_DATA_SOURCE
ds_XML	/shared/examples/ds_XML	DATA_SOURCE	XML_FILE_DATA_SOURCE

getAncestorResources

Get all of the ancestors of the specified resource up to and including the root resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath** - CIS source path to an actual resource	pathType
IN	**resourceType** - Type of CIS resource to be created. It is null on the first invocation.	VARCHAR
OUT	result	CURSOR lineageTreeType (resourceName VARCHAR(255), resourcePath pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, id INTEGER, changeid INTEGER, ownerDomain VARCHAR(255),

Direction	Parameter Name	Parameter Type
		ownerName VARCHAR(255), impactLevel VARCHAR(255), childCount INTEGER);

2. Examples:

2.1. Assumptions: none

Direction	Parameter Name	Parameter Value
IN	resourcePath	/shared/examples/ds_orders/customers
IN	resourceType	TABLE
OUT	result	See charts below

2.2. Chart 1: Columns (1-4)

resourceName	resourcePath	resourceType	subtype
[NULL]	/	CONTAINER	NONE
Shared	/shared	CONTAINER	FOLDER_CONTAINER
Examples	/shared/examples	CONTAINER	FOLDER_CONTAINER
ds_orders	/shared/examples/ds_orders	DATA_SOURCE	RELATIONAL_DATA_SOURCE

2.3. Chart 2: Columns (5-11)

enabled	id	changeid	ownerDomain	ownerName	impactLevel	childCount
1	1	202	composite	system	NONE	6
1	10104	202	composite	admin	NONE	9
1	20586	205	composite	admin	NONE	11
1	20670	205	composite	admin	NONE	7

2.4. Chart 3: Columns (11-15)

dsID	dsResName	dsResPath	dsResType	dsResSubType
[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
20605	ds_inventory	/shared/examples/ds_inventory	DATA_SOURCE	RELATIONAL_DATA_SOURCE

2.5. Chart 4: Columns (16-17)

dsEnabled	dsChildCount
[NULL]	[NULL]
1	4

GetAnsi2NativeMapping

Given a Composite (ANSI) data type and a path to a data source, this procedure returns the data source's data type equivalent. This is a wrapper script that automatically detects which version of CIS is running and calls the appropriate CJP.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datasourcePath	VARCHAR(2147483647)
IN	cisType	VARCHAR(2147483647)
OUT	result	CURSOR(cisType VARCHAR(2147483647), cisNormalizedType VARCHAR(2147483647), cisBaseType VARCHAR(2147483647), cisScale INTEGER, cisPrecision INTEGER, dataTypeId INTEGER, dataTypeName VARCHAR(2147483647), nativeType VARCHAR(2147483647), nativeBaseType VARCHAR(2147483647), nativeScale INTEGER, nativePrecision INTEGER)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	datasourcePath	'/shared/examples/ds_orders'
IN	cisType	'LONGVARCHAR'
OUT	result	('LONGVARCHAR', 'LONGVARCHAR', 'LONGVARCHAR', -1, -1, -983, 'LONGVARCHAR', 'varchar(2147483647)', 'varchar', '2147483647', '-1')

getBasicResourceCursor

This procedure retrieves the basic resource metadata for a given resource. A cursor of metadata is returned. The results being returned will contain values or be null based on the type of resource as described below:

- If a resource is impacted (showing red in Studio), then the impactLevel and impactMessage will contain the level and the reason for being impacted otherwise impactLevel=NONE.
- If the resource is a data source or a container (folder, catalog, schema) then childCount will contain the number of children.
- If the resource is a data source then the dataSourceType will contain what type it is such as "File-XML", "Oracle 11g (Thin Driver)" or "PostgreSQL 9.1"
- If the resource is a procedure then the scriptText will contain the actual procedure definition.
- If the resource is a view then the tableType=VIEW and the sqlText contains the view definition.
- If the resource is a database table then the tableType=TABLE and sqlText is null.

This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (name VARCHAR(255), "path" VARCHAR(32768), "type" VARCHAR(32768), subtype VARCHAR(255), enabled BIT, id INTEGER, changeid INTEGER,

Direction	Parameter Name	Parameter Type
		version VARCHAR(255), introspectState VARCHAR(255), ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), impactMessage VARCHAR(32768), annotation LONGVARCHAR, explicitlyDesigned BIT, tableType VARCHAR(255), sqlText LONGVARCHAR, scriptText LONGVARCHAR, childCount INTEGER, dataSourceType. VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	/shared/ASAssets/Utilities/repository/examples/source/proc1
IN	resourceType	PROCEDURE
OUT	resourceCursor	name: ‘proc1’ path: ‘/shared/ASAssets/Utilities/repository/examples/source/proc1’ type: ‘PROCEDURE’ subtype: ‘SQL_SCRIPT_PROCEDURE’ enabled: 1 id 548377 changeid 16318 version null introspectState null ownerDomain composite ownerName admin impactLevel NONE impactMessage null annotation Proc1 description explicitlyDesigned 0 tableType null scriptText <too large to fit in this window>

Direction	Parameter Name	Parameter Value
		childCount null dataSourceType null

getBasicResourceCursor_All [DEPRECATED]

This procedure retrieves the "all" basic resource metadata for a given resource. A cursor of metadata is returned.

This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT` - this procedure takes the XML response from getBasicResourceXML() and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (name VARCHAR(255), "path" VARCHAR(32768), "type" VARCHAR(32768), subtype VARCHAR(255), enabled BIT, id INTEGER, changeId INTEGER, ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), annotation LONGVARCHAR, explicitlyDesigned BIT, tableType VARCHAR(255), sqlText LONGVARCHAR, scriptText LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/trigger1'
IN	resourceType	TRIGGER
OUT	resourceCursor	(Result too large to display)

getBasicResourceCursor_ActionAttributes

This procedure retrieves the resource metadata for a given resource that is contains Action Attributes such as a TRIGGER. A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_ActionAttributes` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (actionType VARCHAR(255), name VARCHAR(255), "type" VARCHAR(255), "value" VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/trigger1'
IN	resourceType	TRIGGER
OUT	resourceCursor	actionType name type value PROCEDURE PARAMETERS STRING 'string2'

Direction	Parameter Name	Parameter Value
		PROCEDURE PATH STRING '/shared/ASAssets/Utilities/repository/examples/source/proc1'

getBasicResourceCursor_PROCEDURE

This procedure retrieves the resource metadata for a given resource that is a PROCEDURE (SQL Script, DB store procedure, transformation, CJP, etc.) A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_PROCEDURE` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(1024), resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, annotation LONGVARCHAR, parameterName VARCHAR(255), parameterType VARCHAR(255), parameterDirection VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/LookupProduct'
IN	resourceType	'PROCEDURE'
OUT	resourceCursor	(See table below for resourceCursor example output)

2.2. resourceCursor example output:

resourceName	resourcePath	Resource Type	subtype	enabled	annotation	parameterName	parameterType	parameterI
LookupProduct	$1/ LookupProduct	PROCEDURE	SQL_SCRIPT_PROCEDURE	1		[NULL]	[NULL]	[NULL]
LookupProduct	$1/ LookupProduct	PROCEDURE	SQL_SCRIPT_PROCEDURE	1		ProductID	INTEGER	IN
LookupProduct	$1/ LookupProduct	PROCEDURE	SQL_SCRIPT_PROCEDURE	1		result	CURSOR(…	OUT

Note: Path $1 = /shared/examples

getBasicResourceCursor_PROCEDURE_CURSOR

This procedure retrieves the cursor output metadata for a given PROCEDURE (SQL Script, DB store procedure, transformation, CJP, etc.) The user must pass a number indicating which cursor input/ouput parameter to return. A cursor of metadata is then returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_PROCEDURE` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40))
IN	cursorNum	INTEGER
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(1024), resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, annotation LONGVARCHAR, tableType VARCHAR(255), sqlText VARCHAR(32768), columnName VARCHAR(255), columnType VARCHAR(255), nativeBaseType VARCHAR(255), nativeType VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/LookupProduct'
IN	resourceType	'PROCEDURE'
IN	cursorNum	1
OUT	resourceCursor	(Result too large to display here)

getBasicResourceCursor_ResourceAttributes

This procedure retrieves the resource metadata for a given resource that is a PROCEDURE (SQL Script, DB store procedure, transformation, CJP, etc.) A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_PROCEDURE` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (name VARCHAR(255), type VARCHAR(1024), value LONGVARCHAR, valueList LONGVARCHAR, valueMap LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders'
IN	resourceType	'DATA_SOURCE'
OUT	resourceCursor	('autoAddChildren', 'BOOLEAN', 'true', NULL, NULL),

Direction	Parameter Name	Parameter Value
		('login', 'STRING', 'tutorial', NULL, NULL), …

getBasicResourceCursor_SQL_TABLE

This procedure retrieves the resource metadata for a given resource that is a SQL TABLE/VIEW. A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_SQL_TABLE` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(1024), resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, tableType VARCHAR(255), sqlText VARCHAR(32768), columnName VARCHAR(255), columnType VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	/shared/ASAssets/Utilities/repository/examples/source/PRODUCT_VIEW
IN	resourceType	TABLE
OUT	resourceCursor	See table below for resourceCursor example output

2.2. resourceCursor example output:

2.2.1. Note: Path $1 = /shared/ASAssets/Utilities/repository/examples/source

resourceName	resourcePath	Resource Type	subtype	enabled	Table Type	SQL Text	columnName	columnType
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	Not shown	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	NULL	ProductID	INTEGER
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	ProductName	VARCHAR(50)
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	ProductDescription	VARCHAR(255)
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	CategoryID	INTEGER
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	SerialNumber	VARCHAR(50)
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	UnitPrice	DECIMAL(12,2)
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	ReorderLevel	INTEGER
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	VIEW	VIEW	LeadTime	VARCHAR(30)

Note: Path $1 = /shared/ASAssets/Utilities/repository/examples/source

getBasicResourceCursor_SQL_TABLE_SQLINDEXES

This procedure retrieves the resource metadata for a given resource that is a SQL TABLE/VIEW. A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_SQL_TABLE` - this procedure takes the XML response turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), sqlIndexName VARCHAR(255), sqlIndexType VARCHAR(255), sqlIndexUnique BIT, sqlIndexColName VARCHAR(255), sqlIndexColOrder VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/PRODUCT_VIEW'
IN	resourceType	'TABLE'
OUT	resourceCursor	See table below for resourceCursor example output

2.2. resourceCursor example output:

2.2.1. Note: Path $1 = /shared/ASAssets/Utilities/repository/examples/source

sqlIndexName	sqlIndexType	sqlIndexUnique	sqlIndexColName	sqlIndexColOrder
productPK	PRIMARY_KEY	1	ProductID	ASCENDING

Note: Path $1 = /shared/ASAssets/Utilities/repository/examples/source

getBasicResourceCursor_XSLT_TEXT

This procedure retrieves the resource metadata for a given resource that is an XSLT procedure. A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getBasicResourceXML` - this performs the actual invocation to the CIS repository API and returns XML
- `repository/lowerLevelProcedures/getBasicResourceXSLT_XSLT_TEXT` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (name VARCHAR(255), "path" VARCHAR(32768), "type" VARCHAR(255), subtype VARCHAR(255), enabled BIT, explicitlyDesigned BIT, transformSourcePath VARCHAR(32768), transformSourceType VARCHAR(255), xsltText VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/lowerLevelProcedures/resourceExistsXSLT'
IN	resourceType	'PROCEDURE'
OUT	resourceCursor	See table below for resourceCursor example output

2.2. resourceCursor example output:

name:	resourceExistsXSLT
path:	/shared/ASAssets/Utilities/repository/lowerLevelProcedures/resourceExistsXSLT
type:	PROCEDURE
subtype:	XSLT_TRANSFORM_PROCEDURE
enabled:	1
explicitlyDesigned:	0
transformSourcePath:	/services/webservices/system/admin/resource/operations/resourceExists
transformSourceType:	PROCEDURE
xsltText:	<xslt:stylesheet version="1.0" xmlns:csw-xform="http://www.compositesw.com/2003/xform" xmlns:ns1="http://www.compositesw.com/services/system/admin/resource" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xslt="http://www.w3.org/1999/XSL/Transform"> <xslt:template match="/"> <xslt:variable name="_resourceExists"/> <xslt:element name="results"> <xslt:for-each select="ns1:resourceExistsResponse"> <xslt:variable name="_resourceExists" select="ns1:exists"/> <xslt:element name="result"> <xslt:element name="resourceExists"> <xslt:value-of select="$_resourceExists"/> </xslt:element> </xslt:element> </xslt:for-each> </xslt:element> </xslt:template> </xslt:stylesheet>

getChildResourcesCursor

This procedure retrieves the child metadata for a given resource. A cursor of metadata is returned. This procedure invokes 2 lower level API procedures:

- `repository/lowerLevelProcedures/getChildResourcesXML` - this performs the actual invocation to the CIS repository API and returns XML

- `repository/lowerLevelProcedures/getChildResourcesXSLT` - this procedure takes the XML response and turns it into a cursor which is more usable by other CIS procedures..

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(1024), resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, isNullable VARCHAR(255), columnName VARCHAR(255), columnType VARCHAR(255), nativeBaseType VARCHAR(255), nativeType VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/definitions'
IN	resourceType	'CONTAINER'

resourceName	resourcePath	resource Type	subtype	enabled	isNullable	columnName	columnType	ntive BaseType	naive Type
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	ProductID	INTEGER	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	ProductName	VARCHAR(50)	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	ProductDescription	VARCHAR(255)	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	CategoryID	INTEGER	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	SerialNumber	VARCHAR(50)	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	UnitPrice	DECIMAL(12,2)	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	ReorderLevel	INTEGER	[NULL]	[NULL]
PRODUCT_VIEW	$1/PRODUCT_VIEW	TABLE	SQL_TABLE	1	[NULL]	LeadTime	VARCHAR(30)	[NULL]	[NULL]
foder1	$1/PRODUCT_VIEW	CONTAINER	FOLDER_CONTAINER	1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
folder2	$1/PRODUCT_VIEW	CONTAINER	FOLDER_CONTAINER	1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
poc1	$1/PRODUCT_VIEW	PROCEDURE	SQL_SCRIPT_PROCEDURE	1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]

Note: Path $1 = /shared/ASAssets/Utilities/repository/examples/target

getCisVersion (Custom Function)

This function returns version (including patch/hotfix) that the current instance of CIS is running.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	VARCHAR(15)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	‘6.0.0.01.05’

getConnectors

This procedure retrieves metadata for all the configured JMS connectors.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	name	VARCHAR(100)
OUT	resourceCursor	CURSOR (name VARCHAR(1024), annotation VARCHAR(1024), connectorType VARCHAR(1024), groupName VARCHAR(1024), jmsClientID VARCHAR(1024), jndiContextFactory VARCHAR(1024), jndiProperties LONGVARCHAR, jndiProviderUrl VARCHAR(1024), jndiUser VARCHAR(50), jndiPassword VARCHAR(50), maxPool INTEGER, minPool INTEGER, poolTimeout INTEGER, queueConnectionFactory VARCHAR(1024), useJNDI VARCHAR(50))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	name	'myMQ'
OUT	resourceCursor	(Result too large to display here.)

getContainer

This procedure retrieves information about the container of the specified resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inResourceType	/lib/resource/ResourceDefs.ResourceType
OUT	parentPath	/lib/resource/ResourceDefs.ResourcePath
OUT	parentType	/lib/resource/ResourceDefs.ResourceType
OUT	parentSubtype	/lib/resource/ResourceDefs.ResourceType

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/CompositeView'
IN	inResourceType	'TABLE'
OUT	parentPath	'/shared/examples'
OUT	parentType	'CONTAINER'
OUT	parentSubtype	'FOLDER_CONTAINER'

getDataSourceCacheConfig

Returns the cache status and cache tracking table paths of a data source. Returns NULL for both output values if the data source is not configured for caching.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inDataSourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
OUT	StatusTablePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
OUT	TrackingTablePath	/lib/resource/ResourceDefs.ResourcePath

Direction	Parameter Name	Parameter Type
		(VARCHAR(4096) as of CIS 5.1)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inDataSourcePath	'/shared/examples/ds_orders'
OUT	StatusTablePath	'/shared/examples/ds_orders/cache_status'
OUT	TrackingTablePath	'/shared/examples/ds_orders/cache_tracking'

getDataSourceRootPath

This procedure returns the file root/url path for a given file datasource path. The root/url path is the actual file system path when no file system security is being used. When file system security is in place, it is the root name mapping.

Output: fileRootPath - The "root" path to the file. If file system security is in place then this will be the root name mapping value. Example values:

No file system security:	C:\files\myfile.txt
With file system security:	MY_FILES_ROOT_NAME

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	debug	CHAR(1)
OUT	fileRootPath	LONGVARCHAR

4. **Examples:**

   4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/myproject/ds_files'
IN	debug	'Y'
OUT	fileRootPath	'C:\files\myfile.txt'

getDataSourceStatsConfig

Retrieve the statistics configuration for a given data source.

Usage Note: The calling user must have:
- The ACCESS_TOOLS right
- Read permission on the data source set
- Read permission on any of the data source's parent folders

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	VARCHAR(4096)
OUT	curs	CURSOR (configured BIT, useEnabled BIT, tableGatherDefault VARCHAR, numThreads INTEGER, maxTime INTEGER, refreshMode VARCHAR, scheduleMode VARCHAR, startTime TIMESTAMP, fromTimeInADay BIGINT, endTimeInADay BIGINT, recurringDay INTEGER, interval INTEGER, period VARCHAR, count INTEGER, isCluster BIT,)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
OUT	result	(0, NULL, …)

getDefSetDefs

Dumps the contents of an SQL definition set to a cursor.

Usage Note: The calling user must have:
- The ACCESS_TOOLS right
- Read permission on the definition set
- Read permission on any of the definition set's parent folders

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	defSetPath	VARCHAR(4096)
OUT	result	CURSOR (defName VARCHAR(32768), defType VARCHAR(32768), dataType VARCHAR(32768), defValue VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	defSetPath	'/lib/utils/System'
OUT	result	('CannotExecuteSelectException', 'EXCEPTION_DEFINITION', NULL, NULL), … ('CANCELED', 'CONSTANT_DEFINITION', 'VARCHAR(255)', 'CANCELED'), … ('CONTENT', 'TYPE_DEFINITION', 'VARCHAR(65536)', NULL), …

getDependentResourcesCursor

This procedure retrieves the immediately dependent metadata for a given resource. A cursor of metadata is returned. This procedure invokes `repository/lowerLevelProcedures/getDependentResourcesXSLT`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(4096), resourceType VARCHAR(40),

Direction	Parameter Name	Parameter Type
		subtype VARCHAR(40), id INTEGER, ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), impactMessage VARCHAR(32768), enabled BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
OUT	resourceCursor	('ViewOrder', '/shared/examples/ViewOrder', 'TABLE', 'SQL_TABLE', 1003 'composite', 'admin', 'NONE', NULL, 1)

getDependentResourcesRecurseCursor

This procedure is similar to `repository/getDependentResourcesCursor()` (and in fact uses it to generate its results), however it recursively returns all the resource's dependents instead of just the immediate dependents.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	resourceType	/lib/resource/ResourceDefs.ResourceType

Direction	Parameter Name	Parameter Type
		(VARCHAR(40) as of CIS 5.1)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(4096), resourceType VARCHAR(40), subtype VARCHAR(40), id INTEGER, ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), impactMessage VARCHAR(32768), enabled BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
OUT	resourceCursor	('ViewOrder', '/shared/examples/ViewOrder', 'TABLE', 'SQL_TABLE', 1003 'composite', 'admin', 'NONE', NULL, 1), …

getDependentResourcesDirectCursor

This procedure retrieves the "direct" dependent resource metadata for a given resource. If a dependent resource is a foreign key reference to another view it is not returned. If a dependent resource is a data source reference, it is not returned. A cursor of metadata is returned. This procedure invokes `repository/lowerLevelProcedures/getDependentResourcesXSLT`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(4096), resourceType VARCHAR(40), subtype VARCHAR(40), id INTEGER, ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), impactMessage VARCHAR(32768), enabled BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
OUT	resourceCursor	('ViewOrder', '/shared/examples/ViewOrder', 'TABLE', 'SQL_TABLE', 1003 'composite', 'admin', 'NONE', NULL, 1)

getDependentResourcesDirectRecurseCursor

This procedure recursively retrieves the "direct" dependent resource metadata for a given resource. If a dependent resource is a foreign key reference to another view it is not returned. If a dependent resource is a data source reference, it is not returned. A cursor of metadata is returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
IN	includeInitialPath	SMALLINT
IN	inLineageResourceIdList	LONGVARCHAR
OUT	resourceCursor	CURSOR (resourceName VARCHAR(255), resourcePath VARCHAR(4096), resourceType VARCHAR(40), subtype VARCHAR(40), id INTEGER, ownerDomain VARCHAR(255), ownerName VARCHAR(255), impactLevel VARCHAR(255), impactMessage VARCHAR(32768), enabled BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
IN	includeInitialPath	1
IN	inLineageResourceIdList	null
OUT	resourceCursor	('ViewOrder', '/shared/examples/ViewOrder', 'TABLE',

Direction	Parameter Name	Parameter Value
		'SQL_TABLE', 1003 'composite', 'admin', 'NONE', NULL, 1), …

getImpactedResources

This procedure takes a folder path as input, walks the tree of resources in the input folder, and reports on any impacted resources. If no resources are marked as impacted, an empty result set will be returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startingFolder	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
OUT	result	CURSOR (ResourcePath VARCHAR(4096), ResourceType VARCHAR(40), ImpactLevel VARCHAR(32768), ImpactMessage VARCHAR(32768))

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples'
OUT	resourceCursor	('/shared/examples/NewView', 'TABLE', 'UNKNOWN', 'View is newly created and has not been saved.'), …

getIntrospectableResourceIdsResult

This procedure gathers the results from a call to `repository/getIntrospectableResourceIdsTask()`. If the introspection task is still running, the procedure can be called in such a way as to block execution until the task completes before returning results.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	taskId	LONGVARCHAR
IN	block	BIT
IN	pageSize	INTEGER
IN	pageStart	INTEGER
OUT	result	CURSOR (totalResults INTEGER, completed BIT, lastUpdate TIMESTAMP, "path" VARCHAR(4096), "type" VARCHAR(40), subtype VARCHAR(40))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	taskId	'1000'
IN	block	1
IN	pageSize	NULL
IN	pageStart	NULL
OUT	result	(8, 1, '2012-11-01 10:45:00', 'cache_status', 'TABLE', 'DATABASE_TABLE'), …

getIntrospectableResourceIdsTask

This procedure begins an asynchronous thread that scans a data source for introspectable objects. This asynchronous thread will survive server restarts. This is used to populate the introspection cache and may take some time to run. Use `repository/getIntrospectableResourceIdsResult()` to retrieve the results.

Input:

dsPath - The path to the data source.
Values: Any valid data source path.

dsContainerPath - The relative path to the data source container to begin introspection.
Values: Any relative path (i.e. 'mySchema' or 'myCatalog/mySchema'). May be NULL to indicate the entire data source should be scanned.

dsContainerType - The type of the data source container to begin introspection.
Values: This will nearly always be 'CONTAINER' (see /lib/resource/ResourceDefs for other types.)

dsContainerSubType - The subtype of the data source container to begin introspection.
Values: Any container sub-type (see /lib/resource/ResourceDefs.)

recurse - indicates whether introspection should be recursive or not.
Values: 1 or 0

Output:

taskId - The introspection task ID. Use this with `repository/getIntrospectableResourceIdsResult()`.
Values: A task ID

totalResults - Total size of the result set (if known.)
Values: A positive integer or NULL.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
IN	dsContainerPath	/lib/resource/ResourceDefs.ResourcePath
IN	dsContainerType	/lib/resource/ResourceDefs.ResourceType
IN	dsContainerSubType	/lib/resource/ResourceDefs.ResourceType
IN	recurse	BIT
OUT	taskId	VARCHAR(32768)

Direction	Parameter Name	Parameter Type
OUT	totalResults	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
IN	dsContainerPath	NULL
IN	dsContainerType	NULL
IN	dsContainerSubType	NULL
IN	recurse	0
OUT	taskId	'1000'
OUT	totalResults	NULL

getIntrospectedResourceIdsResult

This procedure gathers the results from a call to `repository/getIntrospectedResourceIdsTask()`. If the introspection task is still running, the procedure can be called in such a way as to block execution until the task completes before returning results.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	taskId	LONGVARCHAR
IN	block	BIT
IN	pageSize	INTEGER
IN	pageStart	INTEGER
OUT	result	CURSOR (totalResults INTEGER, completed BIT, lastUpdate TIMESTAMP, "path" VARCHAR(4096), "type" VARCHAR(40))

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	taskId	'1000'
IN	block	1
IN	pageSize	NULL
IN	pageStart	NULL
OUT	result	(8, 1, '2012-11-01 10:45:00', 'cache_status', 'TABLE'), …

getIntrospectedResourceIdsTask

This procedure begins an asynchronous thread that scans a data source for introspected objects. This asynchronous thread will survive server restarts. Use `repository/getIntrospectedResourceIdsResult()` to retrieve the results.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
OUT	taskId	VARCHAR(32768)
OUT	totalResults	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
OUT	taskId	'1000'
OUT	totalResults	NULL

getLockedResources

This procedure is used to generate a string containing the output columns returned by a view or procedure for easy definition of a custom row type. The procedure returns the full list of output colums for a table or view and the full list of out and inout columns for a procedure. The procedure can also strip out column definitions of any output cursors and include them in the list of returned columns.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	detail	VARCHAR(6)
IN	includeOnlyUnlockableResources	VARCHAR(5)
OUT	result	CURSOR (name VARCHAR(32768), "path" VARCHAR(32768), "type" VARCHAR(32768), subtype VARCHAR(32768), id VARCHAR(32768), changeId INTEGER, version VARCHAR(32768), introspectState VARCHAR(32768), ownerDomain VARCHAR(32768), ownerName VARCHAR(32768), impactLevel VARCHAR(32768), impactMessage VARCHAR(32768), enabled BIT, annotation VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	detail	‘FULL’
IN	includeOnlyUnlockableResources	‘FALSE’
OUT	result	(‘CompositeView’, ‘/shared/examples/CompositeView’, ‘TABLE’, ‘SQL_TABLE’, 21874, 14124, NULL, NULL, ‘composite’, ‘admin’,

Direction	Parameter Name	Parameter Value
		'NONE', NULL, 1, NULL)

getOutputColDefs

This procedure is used to generate a string containing the output columns returned by a view or procedure for easy definition of a custom row type. The procedure returns the full list of output colums for a table or view and the full list of out and inout columns for a procedure. The procedure can also strip out column definitions of any output cursors and include them in the list of returned columns.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
IN	convertCursorsToCols	BIT
OUT	rowDef	VARCHAR(2147483647)

3. **Examples:**

3.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResroucePath	'/shared/examples/ds_orders/customers'
IN	resourceType	'TABLE'
IN	convertCursorsToCols	0
OUT	rowDef	' CustomerID INTEGER, CompanyName VARCHAR(50), ContactFirstName VARCHAR(30), ContactLastName VARCHAR(50), BillingAddress VARCHAR(255), City VARCHAR(50), StateOrProvince VARCHAR(20), PostalCode VARCHAR(20), CountryRegion VARCHAR(50), ContactTitle VARCHAR(50), PhoneNumber VARCHAR(30), FaxNumber VARCHAR(30)'

getResourceAnnotations

This procedure returns the annotations of a resource and its columns (if any.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40))
OUT	result	CURSOR TypeDefinitions.ColumnAnnotationRow

2. **Examples:**

2.1. **Assumptions: none.**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
IN	resourceType	'CONTAINER'
OUT	result	columnName: NULL columnType: NULL annotation: 'This folder contains pre-created resources.'

getResourceByDate

This procedure is used to return a list of resources from a given starting folder based on the options described in in the "optionType" input parameter. Depending on the option type selected, it may use the creation date attribute or the modification date attribute when determining the result set. The starting resource is included in the output list if applicable for the option and parameters passed in.

Because the results are not ordered, the user of this method may wish to execute the procedure with an ORDER BY.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	optionType	INTEGER
IN	resourcePath	TypeDefinitions.pathType
IN	resourceType	VARCHAR
IN	resourceNum	INTEGER
IN	resourceDate	TIMESTAMP

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR

2. **Examples:**

    2.1. **Assumptions: none.**

Direction	Parameter Name	Parameter Value
IN	optionType	1
IN	resourcePath	'/shared/examples'
IN	resourceType	'CONTAINER'
IN	resourceNum	1
IN	resourceDate	'2013-08-16 00:00:00'
OUT	result	<row set>

getResourceCacheConfig

This procedure returns the metadata for a resource that has caching configured.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	cacheConfigured	VARCHAR(255)
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

    2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source'
IN	resourceType	'PROCEDURE'
OUT	cacheConfigured	'true' or 'false'
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

getResourceCacheConfigCursor

Expands on `repository/getResourceCacheConfig` by providing more detailed information about a resource's caching configuration.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	inType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
OUT	result	CURSOR (configured BIT, enabled BIT, storageMode VARCHAR(32768), storageDataSourcePath VARCHAR(32768), storageTargetName VARCHAR(32768), storagePath VARCHAR(32768), storageType VARCHAR(32768), refreshMode VARCHAR(32768), scheduleMode VARCHAR(32768), startTime TIMESTAMP, fromTimeInADay BIGINT, endTimeInADay BIGINT, recurringDay INTEGER, "interval" INTEGER, period VARCHAR(32768), "count" INTEGER, isCluster BIT, expirationPeriod BIGINT, clearRule VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/ds_orders/orders'
IN	inType	'TABLE'
OUT	result	configured: 1 enabled: 1 storageMode: DATA_SOURCE

Direction	Parameter Name	Parameter Value
		storageDataSourcePath: /shared/examples/ds_orders storageTargetName: result storagePath: /shared/examples/ds_orders/orders_cache storageType: TABLE refreshMode: MANUAL scheduleMode: [NULL] startTime: [NULL] fromTimeInADay: [NULL] endTimeInADay: [NULL] recurringDay: [NULL] interval: [NULL] period: [NULL] count: [NULL] isCluster: [NULL] expirationPeriod: 0 clearRule: NONE

getResourceCreated

This procedure returns the creation date of a resource. If no creation date is recorded, then NULL is returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inResourceType	/lib/resource/ResourceDefs.ResourceType
OUT	created	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/CompositeView'
IN	inResourceType	'TABLE'
OUT	created	NULL

getResourceImpactedCursor

Returns the impacted information for a resource (if any.) A resource may be impacted because of a syntax error of its code, a missing dependent resource, caching misconfiguration, etc.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	inType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
OUT	result	CURSOR (impactLevel VARCHAR(32768), impactMessage VARCHAR(32768))

2. **Examples:**

   2.1. **Assumptions: A new view created in /shared/examples/NewView that has not been edited or saved.**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/NewView'
IN	inType	'TABLE'
OUT	result	impactLevel: UNKNOWN impactMessage: View is newly created and has not been saved.

getResourceLastModified

This procedure returns the last modified date of a resource. If no last modified date is recorded then the creation date is returned. If no last modified or creation date is recorded, then NULL is returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inResourceType	/lib/resource/ResourceDefs.ResourceType
OUT	lastModified	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/CompositeView'
IN	inResourceType	'TABLE'
OUT	lastModified	2014-03-28 15:44:48.258

getResourceLineageDatasources

Return the list of Data Sources used by a given resource path.

1. Start with the resource path and recursively walk its lineage until no more resources are found.
2. For each resource found, determine if that resources has as associated resource of type=DATA_SOURCE.
3. Find the distinct list of data sources for this resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**resourcePath** – CIS source resource path to being assessing the parent data lineage	pathType
IN	**resourceType** – Type of CIS resource to be created.	VARCHAR
IN	**excludePathsList** – comma separated list of resource paths or partials paths to exclude	LONGVARCHAR
IN	**datasourceAncestor** – flag to get the data source ancestors. 1=yes, 0=no	INTEGER
IN	**inIgnoreResourceDoesNotExist –** flag to ignore missing resources. 1=yes, 0=no	INTEGER
OUT	**datasourceResource**	CURSOR (id INTEGER, resourceName VARCHAR(255), resourcePath pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, childCount INTEGER)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/ViewSales'
IN	resourceType	TABLE
IN	excludePathsList	null
IN	datasourceAncestor	1
IN	inIgnoreResourceDoesNotExist	1
OUT	datasourceResource	See charts below

2.2. Chart 1: Columns (1-5)

id	resourceName	resourcePath	resourceType	subtype
20670	ds_orders	/shared/examples/ds_orders	DATA_SOURCE	RELATIONAL_DATA_SOURCE
20756	ds_XML	/shared/examples/ds_XML	DATA_SOURCE	XML_FILE_DATA_SOURCE

2.3. Chart 2: Columns (6-7)

enabled	childCount
1	1
1	7

getResourceLineageRecursive

This procedure recursively walks the dependent tree to discover resource lineage.

This procedure uses the resource ID to show the lineage by returning the correlation of the resourceID and the parentID. The parentID refers back to the resourceID. The lineage is discovered as the procedure recursively walks the tree. Additionally, the for each child falling within a set of conditions, the ancestor of that child is returned to find out if it has any DATA_SOURCE types in its upstream lineage. That information is returned along with the resource record. This is a flattened way of returning the information all in one record rather than making a separate call to get this information outside of this procedure. However, the user of this procedure may determine that they do not want to calculate ancestors so there is an option to turn that off.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**inSeqNum** – sequence number starting with 1.	INTEGER
IN	**inParentID** – the id of the parent.	INTEGER
IN	**inResourceDepth** – the depth of the parent (number of levels	INTEGER

Direction	Parameter Name	Parameter Type
	which recursion has occurred).	
IN	**inLineageResourceIdList** - a list of space separated resource ids built up as the resources are traversed.	LONGVARCHAR
IN	**resourcePath** - CIS source path to an actual resource	pathType
IN	**resourceType** - Type of CIS resource to be created. It is null on the first invocation.	VARCHAR
IN	**excludePathsList** - comma separate list of resource paths or partials paths to exclude	LONGVARCHAR
IN	**datasourceAncestor** - flag to get the data source ancestor	INTEGER
IN	**inIgnoreResourceDoesNotExist** – flag to ignore missing resources	INTEGER
OUT	resourceTreeList	CURSOR lineageTreeType (seqNum INTEGER, -- generated sequence number resourceID INTEGER, -- resource id from CIS parentID INTEGER, -- how this row relates to resourceID resDepth INTEGER, -- depth as related to the start treeType VARCHAR(255), -- Parent, Child resName VARCHAR(255), -- resource name resPath pathType, -- resource path resType VARCHAR(255), -- resource type subType VARCHAR(255), -- resource sub type enabled BIT, -- enabled or not (1 or 0) dsID INTEGER, -- datasource ancestor id dsResName VARCHAR(255), -- datasource ancestor name

Direction	Parameter Name	Parameter Type
		dsResPath pathType, -- datasource ancestor path dsResType VARCHAR(255), -- datasource ancestor type dsResSubType VARCHAR(255), -- datasource ancestor sub type dsEnabled BIT, -- datasource ancestor enabled (1 or 0) dsChildCount INTEGER -- datasource ancestor number of children);

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSeqNum	null
IN	inParentID	null
IN	inResourceDepth	null
IN	resourcePath	/shared/examples/LookupProduct
IN	resourceType	PROCEDURE
IN	excludePathsList	null
IN	datasourceAncestor	1
IN	inIgnoreResourceDoesNotExist	1
OUT	resourceTreeList	See charts below

2.2. Chart 1: Columns (1-6)

seqnum	resourceID	parentID	resDepth	treeType	resName
1	20760	[NULL]	0	Parent	LookupProduct
2	20633	20760	1	Child	products

2.3. Chart 2: Columns (7-10)

resPath	resType	subType	enabled
/shared/examples/LookupProduct	PROCEDURE	SQL_SCRIPT_PROCEDURE	1
/shared/examples/ds_inventory/products	TABLE	DATABASE_TABLE	1

2.4. Chart 3: Columns (11-15)

dsID	dsResName	dsResPath	dsResType	dsResSubType

[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
20605	ds_inventory	/shared/examples/ds_inventory	DATA_SOURCE	RELATIONAL_DATA_SOURCE

2.5. Chart 4: Columns (16-17)

dsEnabled	dsChildCount
[NULL]	[NULL]
1	4

getResourceLineageDirectRecursive

This procedure recursively walks the "direct" descendent tree to discover resource lineage. It only returns "direct" descendants and not foreign key descendants and not cache related descendants.

This procedure uses the resource ID to show the lineage by returning the correlation of the resourceID and the parentID. The parentID refers back to the resourceID. The lineage is discovered as the procedure recursively walks the tree. Additionally, the for each child falling within a set of conditions, the ancestor of that child is returned to find out if it has any DATA_SOURCE types in its upstream lineage. That information is returned along with the resource record. This is a flattened way of returning the information all in one record rather than making a separate call to get this information outside of this procedure. However, the user of this procedure may determine that they do not want to calculate ancestors so there is an option to turn that off.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**inSeqNum** - sequence number starting with 1.	INTEGER
IN	**inParentID** - the id of the parent.	INTEGER
IN	**inResourceDepth** - the depth of the parent (number of levels which recursion has occurred).	INTEGER
IN	**inLineageResourceIdList** - null to start with. A list of space separated resource ids built up as the resources are traversed.	LONGVARCHAR
IN	**resourcePath** – CIS source path to an actual resource	pathType
IN	**resourceType** – Type of CIS resource to be created. It is null on the first invocation.	VARCHAR
IN	**excludePathsList** – comma separate list of resource paths or	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	partials paths to exclude	
IN	**datasourceAncestor** – flag to get the data source ancestor	INTEGER
IN	**inIgnoreResourceDoesNotExist** – flag to ignore missing resources	INTEGER
OUT	resourceTreeList	CURSOR lineageTreeType (seqNum INTEGER, -- generated sequence number resourceID INTEGER, -- resource id from CIS parentID INTEGER, -- how this row relates to resourceID resDepth INTEGER, -- depth as related to the start treeType VARCHAR(255), -- Parent, Child resName VARCHAR(255), -- resource name resPath pathType, -- resource path resType VARCHAR(255), -- resource type 170ubtype VARCHAR(255), -- resource sub type enabled BIT, -- enabled or not (1 or 0) dsID INTEGER, -- datasource ancestor id dsResName VARCHAR(255), -- datasource ancestor name dsResPath pathType, -- datasource ancestor path dsResType VARCHAR(255), -- datasource ancestor type dsResSubType VARCHAR(255), -- datasource ancestor sub type dsEnabled BIT, -- datasource ancestor enabled (1 or 0) dsChildCount INTEGER -- datasource ancestor number of children);

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSeqNum	null
IN	inParentID	null
IN	inResourceDepth	null
IN	resourcePath	/shared/examples/LookupProduct
IN	resourceType	PROCEDURE
IN	excludePathsList	null
IN	datasourceAncestor	1
IN	inIgnoreResourceDoesNotExist	1
OUT	resourceTreeList	See charts below

getResourceLineageRecursiveAncestors

This procedure was built as a building block with the intention to mimic the Composite Manager capability to show "Dependency Privileges".

This procedure provides the first step, which is to recursively walk the resource lineage depencecy tree to discover child resources for a given input resource. Additionally, the Ancestor folder structure is also retrieved. The reason for the ancestor folders is that privileges need to be assigned to each folder of the ancestry tree. This procedure provides the listing of those folders. The "driver" procedure "`getResourcePrivilegeDependencies`" sits on top of this procedure and will be responsible for retrieving privileges. Any procedure that invokes this procedure will most likely want to do a "SELECT DISTINCT" on the columns so that repeating resource paths are trimmed out.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**inSeqNum** - sequence number starting with 1.	INTEGER
IN	**inParentID** - the id of the parent.	INTEGER
IN	**inResourceDepth** - the depth of the parent (number of levels which recursion has occurred).	INTEGER
IN	**inLineageResourceIdList** - a list of space separated resource ids built up as the resources are traversed.	INTEGER
IN	**resourcePath** - CIS source path	pathType

Direction	Parameter Name	Parameter Type
	to an actual resource	
IN	**resourceType** - Type of CIS resource to be created. It is null on the first invocation.	VARCHAR
IN	**excludePathsList** - comma separate list of resource paths or partials paths to exclude	LONGVARCHAR
IN	**datasourceAncestor** - flag to get the data source ancestor	INTEGER
IN	**inIgnoreResourceDoesNotExist** – flag to ignore missing resources	INTEGER
OUT	resourceTreeList	CURSOR lineageTreeType (seqNum INTEGER, -- generated sequence number resourceID INTEGER, -- resource id from CIS parentID INTEGER, -- how this row relates to resourceID resDepth INTEGER, -- depth as related to the start treeType VARCHAR(255), -- Parent, Ancestor resName VARCHAR(255), -- resource name resPath pathType, -- resource path resType VARCHAR(255), -- resource type subType VARCHAR(255), -- resource sub type enabled BIT, -- enabled or not (1 or 0) dsID INTEGER, -- datasource ancestor id dsResName VARCHAR(255), -- datasource ancestor name dsResPath pathType, -- datasource ancestor path dsResType VARCHAR(255), -- datasource ancestor type dsResSubType VARCHAR(255), -- datasource ancestor sub type

Direction	Parameter Name	Parameter Type
		dsEnabled BIT, -- datasource ancestor enabled (1 or 0) dsChildCount INTEGER -- datasource ancestor number of children);

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSeqNum	null
IN	inParentID	null
IN	inResourceDepth	null
IN	inLineageResourceIdList	null
IN	resourcePath	/shared/examples/ViewSales
IN	resourceType	TABLE
IN	excludePathsList	null
IN	datasourceAncestor	1
IN	inIgnoreResourceDoesNotExist	1
OUT	resourceTreeList	See charts below

2.2. Chart 1: Columns (1-6)

```
seqnum  resourceID  parentID  resDepth  treeType  resName
1       22063       [NULL]    0         Parent    ViewSales
1       1           [NULL]    0         Ancestor  [NULL]
…
```

2.3. Chart 2: Columns (7-10)

```
resPath                      resType     subType     enabled
/shared/examples/ViewSales   TABLE       SQL_TABLE   1
/                            CONTAINER   NONE        1
…
```

2.4. Chart 3: Columns (11-15)

```
dsID    dsResName  dsResPath   dsResType  dsResSubType
[NULL]  [NULL]     [NULL]      [NULL]     [NULL]
[NULL]  [NULL]     [NULL]      [NULL]     [NULL]
```

…

2.5. Chart 4: Columns (16-17)

dsEnabled	dsChildCount
[NULL]	[NULL]
[NULL]	[NULL]

…

getResourceListChildren

Return a list of resources, resource path, type and subtype for a given folder path. The input to this is a starting folder and not an actual resource. Return all immediate children of the starting folder including sub-folders (containers) and non-folder resources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
OUT	resourceTreeList	CURSOR (name VARCHAR, resPath TypeDefinitions.pathType, resType VARCHAR, subType VARCHAR, creationDate TIMESTAMP, creationDateBigint BIGINT, creatorUserDomain VARCHAR(255), creatorUserName VARCHAR(255) lastModifiedDate TIMESTAMP, lastModifiedDateBigint BIGINT, lastModifiedUserDomain VARCHAR(255), lastModifiedUserName VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/ASAssets/Utilities'
IN	resourceType	'CONTAINER'
OUT	resourceTreeList	See chart below

2.2. Chart showing example output for resourceTreeList:

name	resPath	resType	subType
calculations	/shared/ASAssets/Utilities/calculations	CONTAINER	FOLDER_CONTAINER
conversions	/shared/ASAssets/Utilities/conversions	CONTAINER	FOLDER_CONTAINER
file	/shared/ASAssets/Utilities/file	CONTAINER	FOLDER_CONTAINER
log	/shared/ASAssets/Utilities/log	CONTAINER	FOLDER_CONTAINER
repository	/shared/ASAssets/Utilities/repository	CONTAINER	FOLDER_CONTAINER
Etc…			

getResourceListRecursive

Return a list of resources, resource path, type and subtype for a given folder path. The input to this is a starting folder and not an actual resource. Recursively walk down the tree to identify the list of resources. Return all resources including folders (containers) and non-folder resources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
OUT	resourceTreeList	CURSOR (name VARCHAR, resPath TypeDefinitions.pathType, resType VARCHAR, subType VARCHAR, creationDate TIMESTAMP, creationDateBigint BIGINT, creatorUserDomain VARCHAR(255), creatorUserName VARCHAR(255) lastModifiedDate TIMESTAMP, lastModifiedDateBigint BIGINT, lastModifiedUserDomain VARCHAR(255), lastModifiedUserName VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/ASAssets/Utilities'
IN	resourceType	'CONTAINER'
OUT	resourceTreeList	See chart below

2.2. Chart showing example output for resourceTreeList:

name	resPath	resType	subType
TypeDefinitions	/shared/ASAssets/Utilities/TypeDefinitions	PROCEDURE	SQL_SCRIPT_PROCEDURE
calculations	/shared/ASAssets/Utilities/calculations	CONTAINER	FOLDER_CONTAINER
calculateAge	/shared/ASAssets/Utilities/calculations/calculateAge	PROCEDURE	SQL_SCRIPT_PROCEDURE
conversions	/shared/ASAssets/Utilities/conversions	CONTAINER	FOLDER_CONTAINER
convertBoolean	/shared/ASAssets/Utilities/conversions/convertBoolean	PROCEDURE	SQL_SCRIPT_PROCEDURE
convertDoubleToInteger	/shared/ASAssets/Utilities/conversions/convertDoubleToInteger	PROCEDURE	SQL_SCRIPT_PROCEDURE
convertYN	/shared/ASAssets/Utilities/conversions/convertYN	PROCEDURE	SQL_SCRIPT_PROCEDURE
file	/shared/ASAssets/Utilities/file	CONTAINER	FOLDER_CONTAINER
FileProcessingCJP	/shared/ASAssets/Utilities/file/FileProcessingCJP	DATA_SOURCE	NONE
log	/shared/ASAssets/Utilities/log	CONTAINER	FOLDER_CONTAINER
errorNotification	/shared/ASAssets/Utilities/log/errorNotification	PROCEDURE	SQL_SCRIPT_PROCEDURE
logDebugMessage	/shared/ASAssets/Utilities/log/logDebugMessage	PROCEDURE	SQL_SCRIPT_PROCEDURE
repository	/shared/ASAssets/Utilities/repository	CONTAINER	FOLDER_CONTAINER
_debug	/shared/ASAssets/Utilities/repository/_debug	PROCEDURE	SQL_SCRIPT_PROCEDURE
Etc…			

getResourceListUnpublished

This procedure looks at the resources in a starting folder and reports on any that cannot be traced to a dependency that is either a published resource or trigger. In other words, it reports on any orphaned resources that cannot be accessed by an external user over one of the access protocols.

If **excludeTypes** is NULL, then the procedure will automatically exclude resources of type CONTAINER and TRIGGER in its search.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startingFolder	/lib/resource/ResourceDefs.ResourcePath
IN	excludeTypes	VARCHAR(1024)
IN	excludePathsList	LONGVARCHAR
OUT	result	CURSOR (name /lib/resource/ResourceDefs.ResourceName, resPath /lib/resource/ResourceDefs.ResourcePath, resType /lib/resource/ResourceDefs.ResourceType, subtype /lib/resource/ResourceDefs.ResourceType, creationDate TIMESTAMP, creationDateBigint BIGINT, creatorUserDomain VARCHAR(255), creatorUserName VARCHAR(255), lastModifiedDate TIMESTAMP,

Direction	Parameter Name	Parameter Type
		lastModifiedDateBigint BIGINT, lastModifiedUserDomain VARCHAR(255), lastModifiedUserName VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	startingFolder	'/shared/examples'
IN	excludeTypes	NULL
IN	excludePathsList	NULL
OUT	resourceTreeList	[output too large to display]

getResourcePrivilegeDependencies

This procedure was built with the intention to mimic the Composite Manager capability to show "Dependency Privileges". This procedure returns a list of user/group resource privileges for a specified resource path and its dependencies given various inclusion and exclusion filters. This procedure excludes the system paths shown here: /, /shared, /services, /services/databases, /services/webservices. This procedure returns a privilegeStatus= [PASS,FAIL] which indicates whether the privileges for a given dependent view meet the combined privilege criteria [PASS] or they do not [FAIL].

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(4096))
IN	excludePathsList	LONGVARCHAR
IN	inIgnoreResourceDoesNotExist	BIT
IN	nameTypeFilter	VARCHAR
IN	domainFilter	VARCHAR
IN	includeNameEqualFilter	LONGVARCHAR
IN	includeNameLikeFilter	LONGVARCHAR
IN	excludeNameNotEqualFilter	LONGVARCHAR
IN	excludeNameNotLikeFilter	LONGVARCHAR

Direction	Parameter Name	Parameter Type
IN	includePrivsEqualFilter	VARCHAR
IN	includePrivsLikeFilter	VARCHAR
IN	excludePrivsNotEqualFilter	VARCHAR
IN	excludePrivsNotLikeFilter	VARCHAR
IN	includeColumnPrivs	BIT
IN	debug	CHAR(1)
OUT	result	CURSOR (resourceID INTEGER, treeType VARCHAR(255), resName VARCHAR(255), path VARCHAR(4096), type VARCHAR(40), subtype VARCHAR(40), enabled BIT, name VARCHAR(255), nameType VARCHAR(255), domain VARCHAR(255), privilegeStatus VARCHAR(255), privilegeRunTimeAnalysis VARCHAR(255), privilegeDesignTimeAnalysis VARCHAR(255), privs VARCHAR(255), combinedPrivs VARCHAR(255), inheritedPrivs VARCHAR(255), p_N BIT, p_R BIT, p_W BIT, p_E BIT, p_S BIT, p_U BIT, p_I BIT, p_D BIT, p_G BIT, c_N BIT, c_R BIT, c_W BIT, c_E BIT, c_S BIT,

Direction	Parameter Name	Parameter Type
		c_U BIT, c_I BIT, c_D BIT, c_G BIT, i_N BIT, i_R BIT, i_W BIT, i_E BIT, i_S BIT, i_U BIT, i_I BIT, i_D BIT, i_G BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
IN	resourceType	'CONTAINER'
IN	excludePathsList	'/shared/examples/not_this_folder, /shared/examples/not_that_folder'
IN	inIgnoreResourceDoesNotExist	0
IN	nameTypeFilter	'GROUP'
IN	domainFilter	'composite'
IN	includeNameEqualFilter	'all'
IN	includeNameLikeFilter	NULL
IN	excludeNameNotEqualFilter	NULL
IN	excludeNameNotLikeFilter	NULL
IN	includePrivsEqualFilter	NULL
IN	includePrivsLikeFilter	NULL
IN	excludePrivsNotEqualFilter	NULL
IN	excludePrivsNotLikeFilter	NULL

Direction	Parameter Name	Parameter Value
IN	includeColumnPrivs	0
IN	debug	'N'
OUT	result	<Too large to display>

getResourcePrivileges

This procedure returns a list of user resource privileges for a specified resource path given various inclusion and exclusion filters.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(4096))
IN	nameTypeFilter	VARCHAR
IN	domainFilter	VARCHAR
IN	includeNameEqualFilter	LONGVARCHAR
IN	includeNameLikeFilter	LONGVARCHAR
IN	excludeNameNotEqualFilter	LONGVARCHAR
IN	excludeNameNotLikeFilter	LONGVARCHAR
IN	includePrivsEqualFilter	VARCHAR
IN	includePrivsLikeFilter	VARCHAR
IN	excludePrivsNotEqualFilter	VARCHAR
IN	excludePrivsNotLikeFilter	VARCHAR
IN	includeColumnPrivs	BIT
IN	debug	CHAR(1)
OUT	result	CURSOR (name VARCHAR(255), path VARCHAR(4096), type VARCHAR(40), nameType VARCHAR(255), domain VARCHAR(255), privs VARCHAR(255), combinedPrivs VARCHAR(255), inheritedPrivs VARCHAR(255),

Direction	Parameter Name	Parameter Type
		p_N BIT, p_R BIT, p_W BIT, p_E BIT, p_S BIT, p_U BIT, p_I BIT, p_D BIT, p_G BIT, c_N BIT, c_R BIT, c_W BIT, c_E BIT, c_S BIT, c_U BIT, c_I BIT, c_D BIT, c_G BIT, i_N BIT, i_R BIT, i_W BIT, i_E BIT, i_S BIT, i_U BIT, i_I BIT, i_D BIT, i_G BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
IN	resourceType	'CONTAINER'
IN	nameTypeFilter	'GROUP'
IN	domainFilter	'composite'

Direction	Parameter Name	Parameter Value
IN	includeNameEqualFilter	'all'
IN	includeNameLikeFilter	NULL
IN	excludeNameNotEqualFilter	NULL
IN	excludeNameNotLikeFilter	NULL
IN	includePrivsEqualFilter	NULL
IN	includePrivsLikeFilter	NULL
IN	excludePrivsNotEqualFilter	NULL
IN	excludePrivsNotLikeFilter	NULL
IN	includeColumnPrivs	0
IN	debug	'N'
OUT	result	<Too large to display>

getResourcePrivilegesByUser

Return a list of privileges for a resource. Shows explicit privileges, inherited privileges, and combined privileges for each user that has any kind of privileges on the resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(4096))
OUT	result	CURSOR (“path” VARCHAR(32768), “type” VARCHAR(32768), “name” VARCHAR(32768), “domain” VARCHAR(32768), privs VARCHAR(32768), combinedPrivs VARCHAR(32768), inheritedPrivs VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples'
IN	resourceType	'CONTAINER'
OUT	result	See chart below

2.2. Chart showing example output for result:

Path	Type	Name	Domain	Privs	CombinedPrivs	InheritedPrivs
/shared/examples	CONTAINER	admin	composite	READ WRITE EXE…	READ WRITE EXE…	READ WRITE EXE…
/shared/examples	CONTAINER	nobody	composite	NONE	NONE	NONE
Etc.						

getResourceSqlTable

This procedure is used to retrieve table or view metadata. This metadata can be used with `resources/updateResourcesSqlTable()`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath - Full resource path which includes the path and the resource name	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	scripttext - SQL Table text to be updated	LONGVARCHAR
OUT	columnList - a vector array of sql columns and definitions	childResourceType ROW (resourceName VARCHAR, resourcePath TypeDefinitions.pathType, resourceType VARCHAR, columnName VARCHAR, columnType VARCHAR);
OUT	sqlIndexList - a vector array of sql indexes	sqlIndexType ROW (sqlIndexName VARCHAR(255), sqlIndexType VARCHAR(255), sqlIndexUnique BIT, sqlIndexColName VARCHAR(255), sqlIndexColOrder VARCHAR(255));
OUT	foreignKeyList - a vector	foreignKeyType ROW (

Direction	Parameter Name	Parameter Type
	array of foreign keys	fkName VARCHAR(255), fkPrimaryKeyName VARCHAR(255), fkPrimaryKeyTable TypeDefinitions.pathType, fkForeignKeyColumnName VARCHAR(255), fkPrimaryKeyColumnName VARCHAR(255));

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/target/ PRODUCT_VIEW2'
OUT	scripttext	'SELECT * FROM /shared/examples/ds_inventory/products products'
OUT	columnList	[('products','$products','TABLE','ProductID','INTEGER'), ('products','$products','TABLE','ProductName','VARCHAR(50)'), ('products','$products','TABLE','ProductDescription','VARCHAR(255)'), ('products','$products','TABLE','CategoryID','INTEGER'), ('products','$products','TABLE','SerialNumber','VARCHAR(50)'), ('products','$products','TABLE','UnitPrice','DECIMAL(12,2)'), ('products','$products','TABLE','ReorderLevel','INTEGER'), ('products','$products','TABLE','LeadTime','VARCHAR(30)')]
OUT	sqlIndexList	[('productsPK','PRIMARY_KEY',1,'ProductID','ASCENDING')]
OUT	foreignKeyList	[('categoriesFK','categoriesPK','$categories','CategoryID','CategoryID')]

$products = /shared/ASAssets/Utilities/repository/examples/source/ds_inventory/products
$categories = /shared/ASAssets/Utilities/repository/examples/source/ds_inventory/categories

getScriptText (Custom Function)

This procedure returns the script text for a procedure. This allows a program to get the text modify the text and then use another procedure to update the procedure.

The following resource types and sub-types are supported:

resourceType = 'PROCEDURE'

subtype = 'SQL_SCRIPT_PROCEDURE' -- Get Regular Procedure

subtype = 'EXTERNAL_SQL_PROCEDURE' -- Get Packaged Query Procedure

subtype = 'XSLT_TRANSFORM_PROCEDURE' -- Get XSLT Transformation text

subtype = 'XQUERY_TRANSFORM_PROCEDURE' -- Get XQuery Transformation text

resourceType = 'TABLE'

subtype = 'SQL_TABLE' -- Get Regular View

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	subType	VARCHAR(255)
OUT	scriptText	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/proc2'
IN	resourceType	'PROCEDURE'
OUT	subType	'SQL_SCRIPT_PROCEDURE'
OUT	scriptText	'PROCEDURE proc2() BEGIN END'

getTableColumnStatisticsConfiguration

This procedure returns the statistics configuration and manual override information for the table specified in resourcePath, as well as its columns.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	cardinalityMin	INTEGER
OUT	cardinalityMax	INTEGER
OUT	cardinalityExpected	INTEGER

Direction	Parameter Name	Parameter Type
OUT	gatherEnabled	VARCHAR(20)
OUT	maxTime	INTEGER
OUT	columnSettings	CURSOR (name VARCHAR(1000), flags VARCHAR(1000), columnMin DOUBLE, columnMax DOUBLE, columnDistinct INTEGER, enableColumnOverride BIT)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/ds_orders/orders'
OUT	cardinalityMin	NULL
OUT	cardinalityMax	NULL
OUT	cardinalityExpected	NULL
OUT	gatherEnabled	'DEFAULT'
OUT	maxTime	-1
OUT	columnSettings	See table below:

2.2. Chart 1: Columns (1-6)

name	flags	columnMin	columnMax	columnDistinct	enableColumnOverride
'OrderID'	'NONE'	NULL	NULL	NULL	1
'CustomerID'	'NONE'	NULL	NULL	NULL	1
'EmployeeID'	'NONE'	NULL	NULL	NULL	1
…					

getUsedResourcesCursor

This procedure retrieves a cursor of metadata describing what resources are used by the input resource path. The full resource path and resource type must be provided.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType

Direction	Parameter Name	Parameter Type
IN	resourceType	VARCHAR(255)
OUT	usedResCursor	CURSOR (resourceName VARCHAR(255), resourcePath TypeDefinitions.pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, id INTEGER, tableType VARCHAR(255), explicitlyDesigned BIT, sqlText VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/CompositeView'
IN	resourceType	'TABLE'
OUT	usedResCursor	<See table below>

2.2. Chart 1: Columns (1-6)

resourceName	resourcePath	resourceType	subtype	Enabled	Id
VIewOrder	/shared/examples/ViewOrder	TABLE	SQL_TABLE	1	20658
ViewSales	/shared/examples/ViewSales	TABLE	SQL_TABLE	1	20774
ViewSupplier	/shared/examples/ViewSupplier	TABLE	SQL_TABLE	1	20763

2.3. Chart 2: Columns (7-9)

tableType	explicitlyDesigned	sqlText
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …

getUsedResourcesRecurseCursor

This procedure recursively retrieves a cursor of metadata describing what resources are "used" by the resource path provided. For each child dependency resource found for the parent, retrieve its "used" dependencies until the entire lineage has been discovered. The full resource path and resource type must be provided. Use the resource type "LINK" for any published database or web service resources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
IN	inParentID	INTEGER
IN	inLineageResourceIdList	LONGVARCHAR
IN	inIgnoreResourceDoesNotExist	BIT
OUT	result	CURSOR (resourceName VARCHAR(255), resourcePath TypeDefinitions.pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, id INTEGER, tableType VARCHAR(255), explicitlyDesigned BIT, sqlText VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/CompositeView'
IN	resourceType	'TABLE'
IN	inParentID	NULL
IN	inLineageResourceIdList	NULL
IN	inIgnoreResourceDoesNotExist	0
OUT	result	<See table below>

2.2. Chart 1: Columns (1-6)

resourceName	resourcePath	resourceType	subtype	Enabled	Id
VIewOrder	/shared/examples/ViewOrder	TABLE	SQL_TABLE	1	20658
ViewSales	/shared/examples/ViewSales	TABLE	SQL_TABLE	1	20774

```
ViewSupplier    /shared/examples/ViewSupplier    TABLE    SQL_TABLE    1    20763
…
```

2.3. Chart 2: Columns (7-9)

```
tableType        explicitlyDesigned    sqlText
UNKNOWN          0                     SELECT …
UNKNOWN          0                     SELECT …
UNKNOWN          0                     SELECT …
…
```

getUsedResourcesDirectCursor

This procedure retrieves a cursor of metadata describing what resources are "directly" "used" by the resource path provided. It only returns "direct" descendants and not foreign key descendants or cache table or data source references. The full resource path and resource type must be provided.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	usedResCursor	CURSOR (resourceName VARCHAR(255), resourcePath TypeDefinitions.pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, id INTEGER, tableType VARCHAR(255), explicitlyDesigned BIT, sqlText VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/CompositeView'
IN	resourceType	'TABLE'

Direction	Parameter Name	Parameter Value
OUT	usedResCursor	<See table below>

2.2. Chart 1: Columns (1-6)

resourceName	resourcePath	resourceType	subtype	Enabled	Id
VIewOrder	/shared/examples/ViewOrder	TABLE	SQL_TABLE	1	20658
ViewSales	/shared/examples/ViewSales	TABLE	SQL_TABLE	1	20774
ViewSupplier	/shared/examples/ViewSupplier	TABLE	SQL_TABLE	1	20763

2.3. Chart 2: Columns (7-9)

tableType	explicitlyDesigned	sqlText
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …

getUsedResourcesDirectRecurseCursor

This procedure recursively retrieves a cursor of metadata describing what resources are "directly" "used" by the resource path provided. It only returns "direct" descendants and not foreign key descendants or cache table or data source references.

For each child dependency resource found for the parent, retrieve its "used" dependencies until the entire lineage has been discovered. The full resource path and resource type must be provided. Use the resource type "LINK" for any published database or web service resources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
IN	inParentID	INTEGER
IN	inLineageResourceIdList	LONGVARCHAR
IN	inIgnoreResourceDoesNotExist	BIT

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR (resourceName VARCHAR(255), resourcePath TypeDefinitions.pathType, resourceType VARCHAR(255), subtype VARCHAR(255), enabled BIT, id INTEGER, tableType VARCHAR(255), explicitlyDesigned BIT, sqlText VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/CompositeView'
IN	resourceType	'TABLE'
IN	inParentID	NULL
IN	inLineageResourceIdList	NULL
IN	inIgnoreResourceDoesNotExist	0
OUT	result	<See table below>

2.2. Chart 1: Columns (1-6)

resourceName	resourcePath	resourceType	subtype	Enabled	Id
VIewOrder	/shared/examples/ViewOrder	TABLE	SQL_TABLE	1	20658
ViewSales	/shared/examples/ViewSales	TABLE	SQL_TABLE	1	20774
ViewSupplier	/shared/examples/ViewSupplier	TABLE	SQL_TABLE	1	20763
…					

2.3. Chart 2: Columns (7-9)

tableType	explicitlyDesigned	sqlText
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …
UNKNOWN	0	SELECT …
…		

getUserPermissionsRecursive

This procedure retrieves a cursor of metadata containing the privileges a user has for a given starting CONTAINER. The procedure cursively inspects and reports on the privileges for all the child resources.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	userName	/lib/users/UserDefs.UserName (VARCHAR(255))
IN	domainName	/lib/users/UserDefs.DomainName (VARCHAR(255))
IN	beginFolder	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
OUT	result	CURSOR (resPath /lib/resource/ResourceDefs.ResourcePath, privRead CHAR(1), privWrite CHAR(1), privExecute CHAR(1), privSelect CHAR(1), privDelete CHAR(1), privInsert CHAR(1), privDelete CHAR(1), privGrant CHAR(1))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	userName	'admin'
IN	domainName	'composite'
IN	beginFolder	'/shared/examples'
OUT	result	See chart below

2.2. Chart showing example output for result:

resPath	privRead	privWrite	privExecute	privSelect	privUpdate	privInsert	privDelete	privGrant
/shared/examples/CompositeView	Y	Y	Y	Y	Y	Y	Y	Y
/shared/examples/LookupProduct	Y	Y	Y	Y	Y	Y	Y	Y
Etc.								

impactedTargetsList

This procedure crawls through a starting folder and locates all of the resources that are impacted and produces an output cursor with the resource location, impact level and script text if possible.

inExcludePathsKeywords is a comma separated list of keywords used to exclude paths containing these any of the keywords (case insensitive.) Examples: Analysis, Archive, save, validation

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	inStartingFolders	LONGVARCHAR
IN	inExcludePathsKeywords	LONGVARCHAR
OUT	result	CURSOR (resourcePath /lib/resource/ResourceDefs.ResourcePath, resourceType /lib/resource/ResourceDefs.ResourceType, subType /lib/resource/ResourceDefs.ResourceType, impactLevel VARCHAR(1024), impactMessage VARCHAR(32768), scriptText LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	inStartingFolders	'/shared/examples'
IN	inExcludePathsKeywords	NULL
OUT	result	('/shared/examples/NewView', 'TABLE', 'SQL_TABLE', 'UNKNOWN', 'View is newly created and has not been saved.', 'SELECT * FROM')

importResourcePrivileges

This procedure imports the privileges specified in an XML file on the CIS host filesystem. See the `resources/importResourcePrivileges()` procedure.

The input parameter **updateRecursively** indicates whether to recursively apply the specified privileges to the target resources' children (if the target resource is a CONTAINER, DATA_SOURCE, or TABLE.)

The input parameter **updateDependenciesRecursively** indicates whether to recursively apply the target resources' privileges to the targets' dependencies (resources that are used by the target.)

The input parameter **updateDependentsRecursively** indicates whether to recursively apply the target resources' privileges to the targets' dependents (resources that use the target.)

The **mode** input parameter indicates whether the privileges settings should be applied without modifying any unreferenced privileges ('OVERWRITE_APPEND') or should the privileges be applied exactly as presented in the XML file ('SET_EXACTLY'). If this parameter is NULL then 'OVERWRITE_APPEND' will be used.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	updateRecursively	BIT
IN	updateDependenciesRecursively	BIT
IN	updateDependentsRecursively	BIT
IN	filename	LONGVARCHAR
IN	mode	VARCHAR
OUT	updateResourcePrivilegesResponse	XML
OUT	fault	XML

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	updateRecursively	1
IN	updateDependenciesRecursively	0
IN	updateDependentsRecursively	0
IN	filename	'C:\cis_examples_privileges.xml'
IN	mode	'SET_EXACTLY'

Direction	Parameter Name	Parameter Value
OUT	updateResourcePrivilegesResponse	<xml>
OUT	fault	NULL

introspectResources

This script is used to provide a consistent and generic interface for introspecting database tables. It creates a necessary transaction around introspectResourcesTask and introspectResourcesResult. Since the output variable "introspectionResult" is a string result, the invoker of this procedure may also invoke "/shared/ASAssets/Utilities/repository/introspectResourcesResultCursor" to extract the results into a cursor format. The errStatus may return 'SUCCESS' or 'FAILED'. The following section provides detailed on the input and output variables with a description of each field:

Input:

IN dsPath /lib/resource/ResourceDefs.ResourcePath, -- mandatory. Full path to datasource

IN catalogName VARCHAR, -- optional (null) -

Relational Database - database catalog Name if applicable.

CSV - leave this null as it is not applicable

Excel Non-ODBC - leave this null as it is not applicable

IN schemaNames LONGVARCHAR, -- optional (null) - NULL=no specified schema list OR one or more comma separated schemas to introspect. Note: If a list of schema names are provided and a list of table names are provided the list of table names will be applied in total to each schema name. It is not currently in scope to correlate a list of tables to a list of schemas.

Relational Database - schema name or list of schema names.

CSV - leave this null as it is not applicable

Excel Non-ODBC - the excel file name or list of file names.

IN schemaTablePatterns VARCHAR(4096), -- comma separated list of patterns such as D%, E%, F%. Introspect all tables starting with D, E and F or a bracketed correlated and comma-separated list of table patterns associated with each schema:

schemaNames= SCHEMA1,SCHEMA2 - This is the list of comma-separated schemas

schemaTablePatterns=[M%] [P%] - The 1st set of bracketed table patterns goes with the 1st schema. The 2nd bracketed set of table patterns goes with the 2nd schema

IN tableNames LONGVARCHAR, -- optional. NULL=no specified table list OR one or more comma separated table namess to introspect or a bracketed correlated and comma-separated list of table names associated with each schema:

schemaNames= SCHEMA1,SCHEMA2 - This is the list of comma-separated schemas

tableNames= [T1,T2] [T3,T4] - The 1st set of bracketed tables goes with the 1st schema. The 2nd bracketed set of tables goes with the 2nd schema.

Relational Database - When NULL, all tables for the schema are introspected, otherwise the list of tables provided are introspected.

CSV - When NULL, all CSV files are introspected, otherwise the list of files provided are introspected.

Excel Non-ODBC - When NULL, all sheets are introspected, otherwise the list of sheets provided are introspected.

Table names with spaces may be use double quotes to enclose the name but it is not necessary as spaces are preserved.

IN schemaProcedurePatterns VARCHAR(4096), -- comma separated list of patterns such as D%, E%, F%. Introspect all new procedure patterns starting with D, E and F or a bracketed correlated and comma-separated list of procedure patterns associated with each schema:

schemaNames= SCHEMA1,SCHEMA2 - This is the list of comma-separated schemas

schemaProcedurePatterns=[N%] [O%] - The 1st set of bracketed procedure patterns goes with the 1st schema. The 2nd bracketed set of procedure patterns goes with the 2nd schema

IN procedureNames LONGVARCHAR, -- optional. NULL=no specified procedure list OR 1 or more comma separated procedure names to introspect or a bracketed correlated and comma-separated list of procedure names associated with each schema:

schemaNames= SCHEMA1,SCHEMA2 - This is the list of comma-separated schemas

procedureNames= [P1][P2,P3,P4] - The 1st set of bracketed procedures goes with the 1st schema. The 2nd bracketed set of procedures goes with the 2nd schema

Relational Database - When NULL, all procedures are introspected, otherwise the list of procedures provided are introspected.

CSV - leave this null as it is not applicable

Excel Non-ODBC - leave this null as it is not applicable

Procedure names with spaces may be use double quotes to enclose the name but it is not necessary as spaces are preserved.

IN separator VARCHAR, -- Default=, The separator used to separate lists for the input parameters: tableNames, schemaTablePatterns, schemaProcedurePatterns and procedureNames. The separator value should not exist within the tableNames or procedureNames variable.

IN inDebug CHAR(1), -- Y=debug on, N=debug off.

Output:

OUT errStatus VARCHAR, -- SUCCESS or FAILED

OUT errMessage LONGVARCHAR, -- Error message if errStatus=FAILED, otherwise null

OUT introspectionResult LONGVARCHAR, -- A line (CHR(10)) delimited string of results. Use introspectResourcesResultCursor() to return a cursor.

OUT dataSourceType VARCHAR, -- The type of data source that was introspected.

OUT dataSourceSubtype VARCHAR -- The subtype of data source that was introspected.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
IN	catalogName	VARCHAR
IN	schemaNames	LONGVARCHAR
IN	schemaTablePatterns	VARCHAR(4096)
IN	tableNames	LONGVARCHAR
IN	schemaProcedurePatterns	VARCHAR(4096)
IN	procedureNames	LONGVARCHAR
IN	separator	VARCHAR
IN	inDebug	CHAR(1)
OUT	errStatus	VARCHAR
OUT	errMessage	LONGVARCHAR
OUT	introspectionResult	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_inventory'
IN	catalogName	NULL
IN	schemaName	'tutorial'
IN	schemaTablePatterns	NULL
IN	tableNames	'categories,employees,products'
IN	schemaProcedurePatterns	NULL
IN	procedureNames	NULL
IN	separator	','
OUT	errStatus	SUCCESS
OUT	errMessage	NULL
OUT	introspectionResult	(result too large to display)

introspectResourcesResultCursor

This script is used to extract the introspection result text into a cursor of results. The input to this procedure is the output variable [introspectionResult LONGVARCHAR] from the procedure introspectResources.

2. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	introspectionResult	LONGVARCHAR
OUT	result	CURSOR (totalResults INTEGER, completed BIT, status VARCHAR(32768), introspectorVersion INTEGER, startTime TIMESTAMP, endTime TIMESTAMP, addedCount INTEGER, removedCount INTEGER, updatedCount INTEGER, skippedCount INTEGER, totalCompletedCount INTEGER, toBeAddedCount INTEGER, toBeRemovedCount INTEGER, toBeUpdatedCount INTEGER, totalToBeCompletedCount INTEGER, warningCount INTEGER, errorCount INTEGER, "path" VARCHAR(4096), "type" VARCHAR(40), subtype VARCHAR(40), "action" VARCHAR(32768), durationMs INTEGER, entryStatus VARCHAR(32768), code VARCHAR(32768), name VARCHAR(32768), message LONGVARCHAR, detail VARCHAR(32768), severity VARCHAR(32768))

lowerLevelProcedures/introspectResourcesResult

This procedure was moved to lowerLevelProcedures in release 2018Q1 as it is now a supportive procedure for “introspectResources”. This procedure gathers the results from a call to introspectResourcesTask. If the introspection task is still running, the procedure can be called in such a way as to block execution until the task completes before returning results.

The result output will contain a cursor of rows if the introspection is successful. The cursor of rows will provide information on catalogs, schemas and tables. Otherwise, the additional result rows will indicate what went wrong.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
IN	taskId	LONGVARCHAR
IN	block	BIT
IN	pageSize	INTEGER
IN	pageStart	INTEGER
OUT	result	CURSOR (totalResults INTEGER, completed BIT, status VARCHAR(32768), introspectorVersion INTEGER, startTime TIMESTAMP, endTime TIMESTAMP, addedCount INTEGER, removedCount INTEGER, updatedCount INTEGER, skippedCount INTEGER, totalCompletedCount INTEGER, toBeAddedCount INTEGER, toBeRemovedCount INTEGER, toBeUpdatedCount INTEGER, totalToBeCompletedCount INTEGER, warningCount INTEGER, errorCount INTEGER, "path" VARCHAR(4096), "type" VARCHAR(40), subtype VARCHAR(40), "action" VARCHAR(32768), durationMs INTEGER, entryStatus VARCHAR(32768), code VARCHAR(32768), name VARCHAR(32768), message LONGVARCHAR, detail VARCHAR(32768), severity VARCHAR(32768))

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
IN	taskId	'1000'
IN	block	1
IN	pageSize	NULL
IN	pageStart	NULL
OUT	result	(result too large to display)

lowerLevelProcedures/introspectResourcesTask

This procedure was moved to lowerLevelProcedures in release 2018Q1 as it is now a supportive procedure for “introspectResources”. This procedure begins an asynchronous thread that introspects a data source and adds objects to the CIS repository. This asynchronous thread will survive server restarts. Use `repository/introspectResourcesResult()` to retrieve the results.

See `repository/getIntrospectableResourceIdsTask()` for the list of resources in a data source that CAN be introspected (whether they're in the CIS repository or not.)

See `repository/getIntrospectedResourceIdsTask()` for the list of resources in a data source that already ARE introspected.

Input:

dsPath - The path to the data source to be modified.
Values: Any path to a data source

updateAllIntrospectedResources - Indicates whether all currently introspected resources should be updated.
Values: 1 or 0 (NULL indicates FALSE)

failFast - Indicates whether the introspection task should halt on first error or continue on a best effort basis.
Values: 1 or 0 (NULL indicates FALSE)

commitOnFailure - Indicates whether the introspection should commit whatever it can. If fastFail is also TRUE, then only the successfully introspected resources, up to that point, will be committed.
Values: 1 or 0 (NULL indicates FALSE)

autoRollback - Indicates whether the introspection task will rollbackback rather than committing. This supersedes all commit options. This allows you to perform a dry run of resource introspection. The "introspectResourcesResult" procedure is usable if autoRollback is TRUE. If autoRollback is FALSE or NULL, then the introspection will not automatically be rolled

back.
Values: 1 or 0 (NULL indicates FALSE)

scanForNewResourcesToAutoAdd - Indicates whether the introspection task will scan for native resources that have been newly added to the data source. If newly added resources are found and their parent container has the "autoAddChildren" introspection attribute set, then that child will automatically be introspected.
Values: 1 or 0 (NULL indicates FALSE)

runInBackgroundTransaction - Indicates that the introspection task should run as a background transaction.
Values: 1 or 0 (NULL indicates FALSE)

entries - The list of resources to introspect and their respectiveintrospection actions. Optionally (rare) includes introspection attributes. (See /services/webservices/system/admin/resource/operations/getIntrospectionAttributeDefs for details.)
Values: A vector of type /shared/ASAssets/Utilities/TypeDefinitions.IntrospectionPlanVectorType. For example, when introspecting /shared/examples/ds_orders the entities vector might look like:

```
VECTOR [
   ('orders',    'TABLE', 'DATABASE_TABLE', 'ADD_OR_UPDATE', NULL),
   ('customers', 'TABLE', 'DATABASE_TABLE', 'REMOVE',        NULL)
]
```

For an Oracle database, the entities vector might look like (notice the attribute
vector on the SCOTT schema entry):

```
VECTOR [
    ('SCOTT',      'CONTAINER', 'SCHEMA_CONTAINER', 'ADD_OR_UPDATE', VECTOR
[('tablePatterns', 'STRING', 'E%, D%')]),
    ('SCOTT/EMP',  'TABLE',     'DATABASE_TABLE',   'ADD_OR_UPDATE', NULL),
    ('SCOTT/DEPT', 'TABLE',     'DATABASE_TABLE',   'ADD_OR_UPDATE', NULL)
]
```

dsAttributes - The list of introspection attributes to set at the data source level.
Values: A vector of type /shared/ASAssets/Utilities/TypeDefinitions.AttributesVectorType. See /services/webservices/system/admin/resource/operations/getIntrospectionAttributeDefs for detail.
For example, when introspecting /shared/examples/ds_orders the dsAttributes vector might look like:

```
VECTOR [
    ('autoAddChildren',  'BOOLEAN', 'true'),
    ('patternSeparator', 'STRING',  ',')
]
```

Output:

taskId - The introspection task ID. Use this with `repository/introspectResourcesResult()`.
Values: A task ID

totalResults - Total size of the result set (if known.)
Values: A positive integer or NULL.

completed - Indicates whether the introspection task has already finished (if known.)
Values: 1, 0, or NULL (indicating completion status unknown.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	dsPath	/lib/resource/ResourceDefs.ResourcePath
IN	updateAllIntrospectedResources	BIT
IN	failFast	BIT
IN	commitOnFailure	BIT
IN	autoRollback	BIT
IN	scanForNewResourcesToAutoAdd	BIT
IN	runInBackgroundTransaction	BIT
IN	entries	VECTOR (TypeDefinitions.IntrospectionPlanVectorType)
IN	dsAttributes	VECTOR (TypeDefinitions.AttributesVectorType)
OUT	taskId	VARCHAR(32768)
OUT	totalResults	INTEGER
OUT	completed	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	dsPath	'/shared/examples/ds_orders'
IN	updateAllIntrospectedResources	NULL
IN	failFast	1
IN	commitOnFailure	0
IN	autoRollback	1

Direction	Parameter Name	Parameter Value
IN	scanForNewResourcesToAdd	0
IN	runInBackgroundTransaction	0
IN	entries	(see examples above)
IN	dsAttributes	(see example above)
OUT	taskId	'1000'
OUT	totalResults	NULL
OUT	Completed	NULL

rebindFolder

This procedure provides the capability to rebind all the resources in a folder. For example, if a View points to a data source table, you may want to rebind to a different data source that has the same structure. This may be useful when redeploying from Dev to Test to Production or simply rebinding to a different development instance of the database. Rules:

1) If a resource in the folder has both the source and the target sources present, it will use rebindResource to do an explicit rebind.

2) If a resource in the folder does not have the source present, it will rebind using explicit text modification techniques instead of rebindResource. The following text modification techniques are supported for the given resource type:

resourceType = 'TABLE'

subtype = 'SQL_TABLE' -- Regular View not a database table

resourceType = 'PROCEDURE'

subtype = 'SQL_SCRIPT_PROCEDURE' -- Custom Procedure or Parameterized query

subtype = 'EXTERNAL_SQL_PROCEDURE' -- Packaged Query Procedure

subtype = 'BASIC_TRANSFORM_PROCEDURE' -- XSLT Basic Transformation definition

subtype = 'XSLT_TRANSFORM_PROCEDURE' -- XSLT Transformation text

subtype = 'STREAM_TRANSFORM_PROCEDURE' -- XSLT Stream Transformation text

3) If a resource in the folder does not have the target present, that is an error and an exception is raised.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startingResourceFolder	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	rebindFromFolder	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	rebindToFolder	/shared/ASAssets/Utilities/TypeDefinitions.pathType
OUT	success	BIT
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	startingResourceFolder	'/shared/examples/rebind'
IN	rebindFromFolder	'/shared/examples/ds_orders'
IN	rebindToFolder	'/shared/examples/ds_orders_Copy1'
OUT	success	1 or 0
OUT	faultResponse	XML not shown here

rebindResource

This procedure provides the capability to rebind the resources inside of the requested resource. For example, if a View points to a data source table, you may want to rebind to a different data source that has the same structure. This may be useful when redeploying from Dev to Test to Production or simply rebinding to a different development instance of the database.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
IN	rebindVector	VECTOR (rebindRow ROW (oldPath TypeDefinitions.pathType, oldType VARCHAR(255), newPath TypeDefinitions.pathType, newType VARCHAR(255)))

Direction	Parameter Name	Parameter Type
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/PRODUCT_VIEW'
IN	resourceType	'TABLE'
IN	rebindVector	{('/shared/examples/ds_inventory/products', 'TABLE', '/shared/ASAssets/Utilities/repository/examples/source/ds_inventory/products', 'TABLE')}
OUT	success	1 or 0
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

recoverFailedCacheRefresh

Occasionally a cache refresh request will exit without CIS noticing it. This script clears the cache status that says a refresh is "in progress" so that CIS will be able to kick off a new refresh.

Because the data will presumably be in an inconsistent state if an incremental cache refresh is cancelled, this script will change the "in progress" state to "failed" so that any future incremental refresh requests will be forced to do a full refresh.

DO NOT run this on resources whose cache refreshes really are in progress.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	inResourceType	/lib/resource/ResourceDefs.ResourceType

Direction	Parameter Name	Parameter Type
OUT	resultCode	INTEGER
OUT	resultMessage	VARCHAR(65536)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/ds_orders/orders'
IN	inResourceType	'TABLE'
OUT	resultCode	0
OUT	resultMessage	'Cache status update successful.'

refreshResourceStatistics

Refreshes the statistics on a resource. For enabling statistics gathering, also see `repository/updateResourceStatisticsConfig`. Cardinality statistics must be configured on the Data Source. You can execute against a data source or a table/view in the data source.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/ds_inventory/products'
IN	resourceType	'TABLE'
OUT	success	1 or 0

Direction	Parameter Name	Parameter Value
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

reintrospectDataSource

This procedure starts either a blocking or non-blocking data source introspection.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	fullResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	isBlocking	BIT

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	fullResourcePath	'/shared/examples/ds_orders'
IN	isBlocking	1

removeAllFolders

This procedure removes one or more folders and their respective contents. The parameter `fullResourcePathList` should be a comma separated list of full folder paths to remove.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	fullResourcePathList	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'

Direction	Parameter Name	Parameter Value
IN	fullResourcePathList	'/shared/myfolder1,/shared/myfolder2'

removePathQuotes

This procedure removes all quote characters from a resource path string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	LONGVARCHAR
OUT	resultPath	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/"XML"/"path with spaces"/test'
OUT	resultPath	'/shared/XML/path with spaces/test'

replaceStringInAnnotations

This procedure is used to replace a target string in the annotations of one or more resources contained under the resource identified by the value “startPath”. The "searchStr" is case sensitive and used like a wild card on the annotation. If the "searchStr" is null then this procedure will only search for null or empty annotations and update the annotation with the text in "replaceStr". Conversely, if "replaceStr" is null and this procedure will update found resources based on "searchStr" text and set the annotation to null effectively removing the annotation. The “startPath” is a source folder path to assess and fix and has the following rules:

Values: folder exact match: /shared/tmp/1folder/_folder/XML – use this scenario to only search for what is in this exact folder and no sub-folders.

Values: wildcard begin: %/1folder/_folder/XML

Values: wildcard end: /shared/tmp/1folder/_folder/XML% - use this scenario to iterate through sub-folders.

Values: wildcard surrounding: %/1folder/_folder/XML%

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startPath	/lib/resource/ResourceDefs.ResourcePath

Direction	Parameter Name	Parameter Type
IN	startResourceType [deprecated]	/lib/resource/ResourceDefs.ResourceType
IN	searchStr – the string to search for what is to be removed/replaced in the target resource. The string search is is case sensitive but is searched like a wildcard so the text can occur anywhere in the annotation. The string may contain multi-line text. If this string is null then specifically search for null or empty annotations.	LONGVARCHAR
IN	replaceStr – the new string in which to replace with. The string may contain multi-line text. If this string is null then set the target annotation to null.	LONGVARCHAR
OUT	success	BIT
OUT	updateResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: The folder /shared/ASAssets/Utilities/examples has an annotation of "This folder contains pre-created resources."**

Direction	Parameter Name	Parameter Value
IN	startPath	'/shared/examples'
IN	startResourceType [deprecated]	NULL
IN	searchStr	'folder'
IN	replaceStr	'container'
OUT	success	1
OUT	updateResponse	[XML response]
OUT	faultResponse	NULL

replaceStringInResources

This procedure is used to replace a target string in the body of one or more resources contained under the resource identified by the value "startPath". This procedure only looks at views and procedures as they may have a script body whereas other resources do not. All occurrences of the string are replaced. A related procedure, “/Utilities/repository/replaceStringInAnnotations”, is used to replace strings in the annotations of a resource.

Values: folder exact match: /shared/tmp/1folder/_folder/XML – use this scenario to only search for what is in this exact folder and no sub-folders.

Values: wildcard begin: %/1folder/_folder/XML

Values: wildcard end: /shared/tmp/1folder/_folder/XML% - use this scenario to iterate through sub-folders.

Values: wildcard surrounding: %/1folder/_folder/XML%

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startPath	/lib/resource/ResourceDefs.ResourcePath
IN	searchStr – the string to search for what is to be removed/replaced in the target resource. The string may contain multi-line text.	LONGVARCHAR
IN	replaceStr – the string to replace with. It may contain multi-line text. It may not be null but it can be an empty string if the use case is to remove the “searchStr” text.	LONGVARCHAR
IN	caseSensitive – 1/null [default]=Search for “searchStr” with case sensitivity (actual text as is). 0=Perform search with no case sensitivity.	BIT
OUT	numResourcesUpdated	INTEGER
OUT	success	BIT
OUT	updateResponse	XML
OUT	faultResponse	XML

4. **Examples:**

4.1. **Assumptions: The resource /shared/ASAssets/Utilities/examples/xml/getNodeFromXML_SAVE has an annotation of “CSW Version.”**

Direction	Parameter Name	Parameter Value
IN	startPath	‘/shared/examples’

Direction	Parameter Name	Parameter Value
IN	startResourceType	'CONTAINER'
IN	searchStr	'CSW Version'
IN	replaceStr	'DV Version'
OUT	success	1
OUT	updateResponse	[XML response]
OUT	faultResponse	NULL

resourceExists (Custom Function)

Determine if a resource exists or not.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
OUT	success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/proc2'
IN	resourceType	'PROCEDURE'
OUT	success	1

returnFolderNameAndFolderPath

Return the root folder name and the remaining folder path. Used for traversing folder structures either top down or bottom up. This procedure is also used when creating a folder structure from beginning to end.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	mode	CHAR(1)
OUT	folderName	VARCHAR(255)
OUT	folderPath	/shared/ASAssets/Utilities/TypeDefinitions.pathType

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository'
IN	mode	'B'
OUT	folderName	'shared'
OUT	folderPath	'/Utilities/repository'

searchAnnotations

This procedure is used to search a target string in the annotations of one or more resources contained under the resource identified by the value startPath. The search string may be multi-line text. The input parameter "caseSensitive" allows the user to determine if they want to search for annotations using case sensitivity.

Values: folder exact match: /shared/tmp/1folder/_folder/XML – use this scenario to only search for what is in this exact folder and no sub-folders.

Values: wildcard begin: %/1folder/_folder/XML

Values: wildcard end: /shared/tmp/1folder/_folder/XML% - use this scenario to iterate through sub-folders.

Values: wildcard surrounding: %/1folder/_folder/XML%

5. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startPath	/lib/resource/ResourceDefs.ResourcePath
IN	searchStr – the string to search for in the target resource. The string may contain multi-line text. If the string is null then specifically search for null or empty annotations.	LONGVARCHAR
IN	caseSensitive – 1/null [default]=Search for "searchStr" with case sensitivity (actual text as is). 0=Perform search with no case sensitivity.	BIT
OUT	result PIPE (resourceId - The resource identifier	 BIGINT

Direction	Parameter Name	Parameter Type
	resourcePath - The full path to the resource resourceType - The type of resource subtype - The sub-type of the resource owner - The resource owner ownerId - The resource owner id len - The length of the annotation annotation - The resource text that where the keywordList was found)	VARCHAR(4000) VARCHAR(255) VARCHAR(255), VARCHAR(255) BIGINT INTEGER LONGVARCHAR

6. **Examples:**

6.1. **Assumptions: The folder /shared/ASAssets/Utilities/examples/string.**

Direction	Parameter Name	Parameter Value
IN	startPath	'/shared/ASAssets/Utilities/examples/string'
IN	searchStr	'All rights reserved'
IN	caseSensitive	1
OUT	result	Cursor of 3 rows

searchResources

This procedure searches for keywords within the script text of a resource. It returns a cursor of resource paths, types and subtypes for resources that contain the keywords. It also returns the text position of the occurrence of any keyword matches and which keyword was matched. It will not search a resource that is impacted. The impactLevel may be 'NONE' or 'SYNTAX_ERROR'. It will skip starting resource paths that do not exist.

The following resource types and sub-types are supported:

resourceType = 'PROCEDURE'

subtype = 'SQL_SCRIPT_PROCEDURE' -- Get Regular Procedure

subtype = 'EXTERNAL_SQL_PROCEDURE' -- Get Packaged Query Procedure

subtype = 'XSLT_TRANSFORM_PROCEDURE' -- Get XSLT Transformation text

subtype = 'XQUERY_TRANSFORM_PROCEDURE' -- Get XQuery Transformation text

resourceType = 'TABLE'

subtype = 'SQL_TABLE' -- Get Regular View

This procedure uses RegexPosition which has the following rules:

Finds an occurrence of a regular expression match in a VARCHAR and returns the position of the match (similar to the SQL POSITION function, positions start at 1 with 0 indicating a match was not found.) The value of the occurrence input value determines which occurrence to return a numbered occurrence starting at 1 from left to right. (Use negative values to number occurrences from right to left.) If a NULL value is passed in as the value of any of the inputs, a NULL is returned. Zero may not be used as a value for an occurrence.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.5 in CIS 4.0.1) See the javadoc for java.util.regex.Pattern for details on what is supported. The following characters require an escape to be used in front of the character: '\', '(', ')', '[', '{', '?', '*', '+','''. The escape character is a backslas "\". For example: \\, \(, \), \[, \{, \?, *, \+, \'

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**startingFolders** - comma separated list of starting folder to begin searching.	LONGVARCHAR
IN	**keywordList** - comma separated list of keyword strings to search for. If the string contains a comma then it must be enclosed in double quotes like "a,b",c,"d,". The keyword list may also contain regular expressions. For example to search for the data type Integer or integer but not INTEGER use the regular expression [Ii]nteger in the keyword list. The following characters in the text require an escape using a backslash "\" character: '\', '(', ')', '[', '{', '?', '*', '+','''	LONGVARCHAR
IN	**keywordOccurrence** - comma separated list of keyword occurrences which exactly match the number of keywords. If this entry is left null then it is assumed that the occurrence for each keyword is 1. The value of the occurrence input value determines which occurrence to return (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.)	LONGVARCHAR

Direction	Parameter Name	Parameter Type
OUT	result	OUT result PIPE (**startOrder** INTEGER, -- The starting path execution order **startPath** VARCHAR(1024), -- The starting path extracted from the comma separated startingFolderss **resourcePath** VARCHAR(1024), -- The full path to the resource **resourceType** VARCHAR(255), -- The type of resource **subtype** VARCHAR(255), -- The sub-type of the resource **impactLevel** VARCHAR(255), -- The impact level of the resource NONE and SYNTAX_ERROR are permitted. **pos** INTEGER, -- The position of the first occurrence of any of the keywords **keyword** LONGVARCHAR, -- The keyword that was found from the keyword list **keywordNum** INTEGER, -- The keyword position number within the keyword list **occurrence** INTEGER – The occurrence of the keyword)

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	**startingFolders**	/shared/examples, /shared/ASAssets/Utilities/repository/examples
IN	**keywordList**	“ProductID,”, [Ii]nsert, ProductName
IN	**keywordOccurrence**	1,2,-1
OUT	**result**	See Chart 1 below

4.2. Chart 1: columns 1-3

startOrder	startPath	resourcePath
1	/shared/examples	/shared/examples/CompositeView
1	/shared/examples	/shared/examples/LookupProduct
1	/shared/examples	/shared/examples/ViewOrder
1	/shared/examples	/shared/examples/ViewSupplier
1	/shared/examples	/shared/examples/getInventoryTransactions
1	/shared/examples	/shared/examples/productCatalog_Transformation
2	/shared/ASAssets/Utilities/repository/examples	/shared/ASAssets/Utilities/repository/examples/returnGetChildResourcesResponseXML
2	/shared/ASAssets/Utilities/repository/examples	/shared/ASAssets/Utilities/repository/examples/test_multiple_cursors
2	/shared/ASAssets/Utilities/repository/examples	/shared/ASAssets/Utilities/repository/examples/test_searchResources
2	/shared/ASAssets/Utilities/repository/examples	shared/Utilities/repository/examples/test_updateResourcesSqlTable

4.3. Chart 1: columns 4-9

resourceType	subtype	pos	keyword	keywordNum	occurrence
TABLE	SQL_TABLE	41	ProductName	3	-1
PROCEDURE	SQL_SCRIPT_PROCEDURE	277	ProductID,	1	1
TABLE	SQL_TABLE	53	ProductID,	1	1
TABLE	SQL_TABLE	35	ProductID,	1	1
PROCEDURE	XQUERY_TRANSFORM_PROCEDURE	2068	ProductName	3	-1
PROCEDURE	XSLT_TRANSFORM_PROCEDURE	3072	ProductName	3	-1
PROCEDURE	SQL_SCRIPT_PROCEDURE	10578	ProductName	3	-1
PROCEDURE	SQL_SCRIPT_PROCEDURE	850	[Ii]nsert	2	2
PROCEDURE	SQL_SCRIPT_PROCEDURE	919	ProductID,	1	1
PROCEDURE	SQL_SCRIPT_PROCEDURE	2300	ProductName	3	-1

updateBasicTransformationProcedure

This procedure is used to update a basic transformation procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourceType	VARCHAR
IN	annotation	LONGVARCHAR
IN	attributeVector	RepositoryDefinitions.AttributeCompleteVectorType
OUT	Success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/basicXSLT'
IN	transformSourcePath	'/shared/examples/ds_XML/productCatalog.xml'
IN	transformSourceType	'TREE'
IN	annotation	'Product catalog transformation'
IN	attributeVector	NULL
OUT	success	1

updateConnector

This procedure updates a JMS connector

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	name	VARCHAR(100)
IN	groupName	VARCHAR(100)
IN	jmsClientID	VARCHAR(1024)
IN	annotation	VARCHAR(1024)
IN	jndiContextFactory	VARCHAR(1024)
IN	jndiProperties	LONGVARCHAR
IN	jndiProviderUrl	VARCHAR(1024)
IN	jndiUser	VARCHAR(50)
IN	jndiPassword	VARCHAR(50)
IN	queueConnectionFactory	VARCHAR(1024)
IN	minPool	INTEGER
IN	maxPool	INTEGER
IN	poolTimeout	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	name	'myMQ'

Direction	Parameter Name	Parameter Value
IN	groupName	'<Group Name>'
IN	jmsClientID	'<JMS Client ID>'
IN	annotation	'This is a JMS message queue'
IN	jndiContextFactory	'<JNDI context factory>'
IN	jndiProperties	'<JNDI Properties XML>'
IN	jndiProviderUrl	'<JNDI Provider URL>'
IN	jndiUser	'myMQuser'
IN	jndiPassword	'myMQpassword'
IN	queueConnectionFactory	'<Queue Connection Factory>'
IN	minPool	1
IN	maxPool	10
IN	poolTimeout	300

updateDefSetDef

Programmatically updates an entry in a definition set. Inserting a definition that already exists will perform an update instead. Updating a definition that does not exist will do nothing.

NOTE: Updating data types is apparently not supported by the Admin API, so you must first delete then insert to update data types.

Usage Note: The calling user must have:
- The ACCESS_TOOLS right
- Read and write permission on the definition set
- Read permission on any of the definition set's parent folders

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	defSetPath	VARCHAR(4096)
IN	updateType values: 'INSERT', 'UPDATE', or 'DELETE'	VARCHAR(255)
IN	defName	VARCHAR(255)
IN	defType values: 'EXCEPTION_DEFINTION', 'TYPE_DEFINITION', or 'CONSTANT_DEFINITION'	VARCHAR(255)
IN	dataType (NULL for exception definitions)	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	defValue (NULL for exception or type definitions)	VARCHAR(255)
OUT	None: Throws exception upon failure	

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	defSetPath	'/shared/ASAssets/Utilities/repository/examples/testDefSet'
IN	updateType	'INSERT'
IN	defName	'MyNewConstant'
IN	defType	'CONSTANT_DEFINITION'
IN	dataType	'VARCHAR(255)'
IN	defValue	'Hello World!'

UpdateDsColumnAnnotation

This procedure is used to update annotations for data source table columns since there is no Admin API this particular operation. For other types of table/view columns, please use the updateSqlTable() admin API.) This is a wrapper script that automatically detects which version of CIS is running and calls the appropriate CJP.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	column_path	LONGVARCHAR
IN	annotation	LONGVARCHAR
OUT	result	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
IN	column_path	'/shared/examples/ds_orders/orders/OrderID'
IN	Annotation	'OrderID column annotation'
OUT	Result	'Column annotation updated.'

updateExternalSQLProcedure

This procedure is used to update a packaged query procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	externalSqlText	LONGVARCHAR
IN	externalDatasourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	parameterVector	RepositoryDefinitions.AttributeCompleteVectorType
IN	annotation	LONGVARCHAR
IN	attributeVector	RepositoryDefinitions.AttributeCompleteVectorType
OUT	success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/packagedCustomerQuery'
IN	externalSqlText	'select * from customers'
IN	externalDatasourcePath	'/shared/examples/ds_orders'
IN	parameterVector	NULL
IN	annotation	'Customer information'
IN	attributeVector	NULL
OUT	success	1

updateImpactedResource

This procedure is used to update a single impacted resource. The main objective is to work around issues in CIS that are fixable by simply opening, modifying and saving a resource. Typical issues that this can repair are views or procedures that are impacted after import into CIS. Typical error messages might be "session may not be null" or "session is closed". This procedure will not attempt to fix views or procedures that contain an impact level of "SYNTAX_ERROR". Moved from Best Practices to Utilities.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Debug – Y or N	CHAR(1)
IN	resourcePath	LONGVARCHAR
IN	resourceType – TABLE or PROCEDURE	VARCHAR

Direction	Parameter Name	Parameter Type
OUT	success – 0 or 1	BIT
OUT	message – exception message if success=0 otherwise null	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	resourcePath	'/shared/project/myview'
IN	resourceType	TABLE
OUT	success	1
OUT	message	null

updateImpactedResources

This procedure crawls through a starting folder and attempts to update a VIEW or PROCEDURE resource that is impacted. There are some errors that can be resolved that are not syntax related errors. There are some Composite versions that will mark a resource as impacted. Simply reading those resources and saving them back out fixes the impacted issue. This procedure attempts to automate that process. An example of an error is "session is null". This type of error can be fixed simply by reading and writing the resource.

inExcludePathsKeywords is a comma separated list of keywords used to exclude paths containing these any of the keywords (case insensitive.) Examples: Analysis, Archive, save, validation

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	inStartingFolders	LONGVARCHAR
IN	inExcludePathsKeywords	LONGVARCHAR
OUT	success	BIT

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	inStartingFolders	'/shared/myImportedFolder'
IN	inExcludePathsKeywords	NULL
OUT	success	1

updateResourceAnnotations

Updates a resource's annotations, including columns (if any.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	resourceType	/lib/resource/ResourceDefs.ResourceType
IN	annotation	LONGVARCHAR
IN	columnAnnotations	VECTOR(TypeDefinitions.ColumnAnnotationRow)
OUT	success	BIT
OUT	updateResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
IN	annotation	'this is a table annotation'
IN	columnAnnotations	(Vector of column specifications and annotations.)
OUT	success	1
OUT	updateResponse	(Update response XML)
OUT	faultResponse	NULL

updateResourceCacheConfig

This procedure updates a resource's cache configuration setting.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
IN	cacheConfigured	VARCHAR(255)
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/proc1'
IN	resourceType	'PROCEDURE'
IN	cacheConfigured	'true'
OUT	success	1 or 0
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateResourceCacheConfiguration

Sets a resource's cache configuration. Other than "inResourcePath" and "inResourceType", any value can be set to NULL.

NOTE: Only supports configuring resources with a single cursor or a set of scalar outputs. Use `/services/webservices/system/admin/resource/operations/updateResourceCacheConfig` directly, otherwise.

NOTE 2: Does NOT create any database tables necessary for the storage of cache data. This must still be done using the Studio GUI or by hand-edited DDL on the caching data source itself.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inResourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	inResourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
IN	inConfigured	BIT

Direction	Parameter Name	Parameter Type
IN	inEnabled	BIT
IN	inStorageMode	VARCHAR(255)
IN	inStorageDataSourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	inStorageTargetName	/lib/resource/ResourceDefs.ResourceName (VARCHAR(255) as of CIS 5.1)
IN	inStorageTargetPath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
IN	inStorageTargetType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40) as of CIS 5.1)
IN	inRefreshMode	VARCHAR(255)
IN	inScheduleMode	VARCHAR(255)
IN	inStartTime	TIMESTAMP
IN	inInterval	INTEGER
IN	inExpirationPeriod	BIGINT
IN	inClearRule	VARCHAR(255)
IN	inIncremental	BIT
IN	inStorageBucketMode	VARCHAR(255)
IN	inStorageBucketCatalog	VARCHAR(255)
IN	inStorageBucketSchema	VARCHAR(255)
IN	inStorageBucketPrefix	VARCHAR(255)
IN	inStorageBucketNumBuckets	INTEGER
IN	inStorageDropCreateIdx	BIT
IN	inFirstRefreshCallback	VARCHAR(32768)
IN	inSecondRefreshCallback	VARCHAR(32768)
OUT	success	BIT
OUT	updateResponse	XML
OUT	faultResponse	XML

2. **Examples:**

    2.1. **Assumptions: A database table "customers_cache" has been created in the database pointed to by /shared/examples/ds_orders with the correct signature for caching the "customers" table in the same database.**

Direction	Parameter Name	Parameter Value
IN	inResourcePath	'/shared/examples/ds_orders/customers'
IN	inResourceType	'TABLE'
IN	inConfigured	1
IN	inEnabled	1
IN	inStorageMode	'DATA_SOURCE'
IN	inStorageDataSourcePath	'/shared/examples/ds_orders'
IN	inStorageTargetName	'result'
IN	inStorageTargetPath	'/shared/examples/ds_orders/customers_cache'
IN	inStorageTargetType	'TABLE'
IN	inRefreshMode	'MANUAL'
IN	inScheduleMode	NULL
IN	inStartTime	NULL
IN	inInterval	NULL
IN	inExpirationPeriod	0
IN	inClearRule	'NONE'
IN	inIncremental	1
IN	inStorageBucketMode	NULL
IN	inStorageBucketCatalog	NULL
IN	inStorageBucketSchema	NULL
IN	inStorageBucketPrefix	NULL
IN	inStorageBucketNumBuckets	0
IN	inStorageDropCreateIdx	0
IN	inFirstRefreshCallback	'/shared/customersCacheInit'
IN	inSecondRefreshCallback	'/shared/customersCacheUpdate'
OUT	success	1 or 0
OUT	updateResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateResourceCacheEnabled

Sets or unsets the "enabled" flag of the input resource's cache configuration.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
IN	cacheEnabled	VARCHAR(255)
OUT	success	BIT
OUT	updateResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/examples/ds_orders/orders'
IN	resourceType	'TABLE'
IN	cacheEnabled	'false'
OUT	success	1 or 0
OUT	updateResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateResourceDataSource

This procedure is used to update a resource that is a Data Source type. Use this to rebind resources such as an XML Schema and a source input directory for an XML file.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	resourceType	VARCHAR(255)
IN	resourceSubType	VARCHAR(255)
IN	dataSourceType	VARCHAR(255)
IN	dataSourceAttrVector	AttributeType ROW (attrName VARCHAR, attrType VARCHAR, attrValue VARCHAR)
OUT	success	BIT

Direction	Parameter Name	Parameter Type
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/Advisory'
IN	resourceType	'DATA_SOURCE'
IN	resourceSubType	'XML_FILE_DATA_SOURCE'
IN	dataSourceType	'File-XML'
IN	dataSourceAttrVector	VECTOR [('noNamespaceSchemaLocation','STRING', 'file:/CompositeSoftware/Advisory.xsd'), ('root','STRING','file:/CompositeSoftware')]
OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	NULL

updateResourceOwner

This script is used to change resource ownership of a resource. This script can be used to change ownership of a single resource or a folder with multiple resource. By default, the ownership is pushed recursively to all the resources in a folder.

The library function used in this script requires ACCESS TOOLS, READ ALL RESOURCES, READ ALL USERS role, therefore ideally, this script should be executed by an Admin user. The script has no output parameters and may throw an exception.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	resourcePath	LONGVARCHAR
IN	resourceType	VARCHAR(255)
IN	newOwnerName	VARCHAR(255)
IN	newOwnerDomain	VARCHAR(255)

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'Y'
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source'
IN	resourceType	'CONTAINER'
IN	newOwnerName	'user1'
IN	newOwnerDomain	'composite'

updateResourcePrivileges

This procedure is used to update the privileges of one or more resources. It is typically used after a deployment to completely reset the privileges of all the resources that were deployed.

It can be used to set privileges exactly or to modify existing privileges. It can also be used to recursively set privileges on child resources of containers or data sources, dependencies (resources that are used by the target resource), and/or dependents (resources that use the target resource.)

A public data type, ResourcePrivsListType has been defined for the resource privileges list input:

```
VECTOR( -- list of resources and privilege settings
    ROW(
        resourcePath /lib/resource/ResourceDefs.ResourcePath, -- the path to the resource
        resourceType /lib/resource/ResourceDefs.ResourceType, -- the type of the resource
        resourceprivs VECTOR( -- list of privilege settings for the resource
            ROW(
                name VARCHAR,        -- user or group name
                domainName VARCHAR, -- user or group domain
                nameType VARCHAR,    -- a constant value of either 'USER' or 'GROUP'
                privs VARCHAR        -- a space separated list of the privileges
                                       (i.e. 'READ WRITE EXECUTE SELECT')
            )
        )
    )
)
```

For example, the following VECTOR could be used to set the privileges on a view and revoke them on one of the view's columns:

```
VECTOR [
    (
        '/shared/examples/CompositeView',
        'TABLE',
        VECTOR [
            (
                'bob',
                'composite',
                'USER',
                'READ SELECT'
            )
        ]
    ),
    (
        '/shared/examples/CompositeView/CustomerContactPhone',
        'COLUMN', -- note that in this context COLUMN is a valid resource type
        VECTOR [
            (
                'bob',
                'composite',
                'USER',
                'NONE'
            )
        ]
    )
]
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	setExactly	BIT
IN	recurseChildren	BIT
IN	recurseDependencies	BIT
IN	recurseDependents	BIT
IN	resourcePrivsList	VECTOR (ROW (resourcePath ResourceDefs.ResourcePath, resourceType ResourceDefs.ResourceType, resourceprivs VECTOR (ROW (name VARCHAR, domainName VARCHAR, nameType VARCHAR, privs VARCHAR))))
OUT	success	BIT
OUT	responseXML	XML
OUT	faultXML	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	setExactly	1
IN	recurseChildren	1
IN	recurseDependencies	0
IN	recurseDependencies	0
IN	resourcePrivsList	VECTOR [('/shared/examples', 'CONTAINER', VECTOR [('bob', 'composite', 'USER', 'READ SELECT'), …]), …]
OUT	success	1
OUT	responseXML	XML not shown here
OUT	faultXML	NULL

updateResourcesSqlTable

This procedure is used to update the content of a SQL Table View including the indexes for that VIEW. In order to achieve this, an array of the columns and their types along with an array of the SQL indexes are passed into this procedure..

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath - Full resource path which includes the path and the resource name	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	scripttext - SQL Table text to be updated	LONGVARCHAR

Direction	Parameter Name	Parameter Type
IN	columnList - a vector array of sql columns and definitions	childResourceType ROW (resourceName VARCHAR, resourcePath TypeDefinitions.pathType, resourceType VARCHAR, columnName VARCHAR, columnType VARCHAR);
IN	sqlIndexList - a vector array of sql indexes	sqlIndexType ROW (sqlIndexName VARCHAR(255), sqlIndexType VARCHAR(255), sqlIndexUnique BIT, sqlIndexColName VARCHAR(255), sqlIndexColOrder VARCHAR(255));
IN	foreignKeyList - a vector array of foreign keys	foreignKeyType ROW (fkName VARCHAR(255), fkPrimaryKeyName VARCHAR(255), fkPrimaryKeyTable TypeDefinitions.pathType, fkForeignKeyColumnName VARCHAR(255), fkPrimaryKeyColumnName VARCHAR(255));
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/target/ PRODUCT_VIEW2'
IN	scripttext	'SELECT * FROM /shared/examples/ds_inventory/products products'
IN	columnList	[('products','$products','TABLE','ProductID','INTEGER'),

Direction	Parameter Name	Parameter Value
		('products','$products','TABLE','ProductName','VARCHAR(50)'), ('products','$products','TABLE','ProductDescription','VARCHAR(255)'), ('products','$products','TABLE','CategoryID','INTEGER'), ('products','$products','TABLE','SerialNumber','VARCHAR(50)'), ('products','$products','TABLE','UnitPrice','DECIMAL(12,2)'), ('products','$products','TABLE','ReorderLevel','INTEGER'), ('products','$products','TABLE','LeadTime','VARCHAR(30)')]
IN	sqlIndexList	[('productsPK','PRIMARY_KEY',1,'ProductID','ASCENDING')]
IN	foreignKeyList	[('categoriesFK','categoriesPK','$categories','CategoryID','CategoryID')]
OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

$products = /shared/ASAssets/Utilities/repository/examples/source/ds_inventory/products
$categories = /shared/ASAssets/Utilities/repository/examples/source/ds_inventory/categories

updateSqlScript

This procedure is used to update the content of a SQL Procedure script.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	scripttext	LONGVARCHAR
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/proc2'

Direction	Parameter Name	Parameter Value

IN	scripttext	‘PROCEDURE proc2()
		BEGIN
		DECLARE var varchar;
		END’

OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateSqlTable

Updates the definition of a SQL Table resource.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	inScripttext	LONGVARCHAR
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	‘/shared/ASAssets/Utilities/repository/examples/target/PRODUCT_VIEW’
IN	inScripttext	‘SELECT * FROM /shared/examples/ds_inventory/products products’
OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateSqlTableTextAndModel

Updates the SQL text and proprietary model of a SQL Table resource. This is generally used to copy the SQL and model of an existing view to another view without impacting any of the other attributes (privileges, caching, etc.) of the target view. The binary code that describes a proprietary model cannot be generated without Studio, so the only source for model binaries is repository/getBasicResourceCursor_SQL_TABLE()

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	inScripttext	LONGVARCHAR
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/target/PRODUCT_VIEW'
IN	inScripttext	'SELECT * FROM /shared/examples/ds_inventory/products products'
OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateStreamTransformationProcedure

This procedure is used to update a streaming transformation procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourceType	VARCHAR
IN	streamModel	RepositoryDefinitions.XsltModelVectorType
IN	annotation	LONGVARCHAR
IN	isExplicitDesign	BIT

Direction	Parameter Name	Parameter Type
IN	parameterVector	RepositoryDefinitions.AttributeCompleteVectorType
IN	attributeVector	RepositoryDefinitions.AttributeCompleteVectorType
OUT	Success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/streamingXSLT'
IN	transformSourcePath	'/shared/examples/ds_XML/productCatalog.xml'
IN	transformSourceType	'TREE'
IN	streamModel	NULL
IN	annotation	'Product catalog transformation'
IN	isExplicitDesign	0
IN	parameterVector	NULL
IN	attributeVector	NULL
OUT	success	1

updateTableColumnStatisticsConfiguration

This procedure updates the statistics configuration and manual override information for the table specified in resourcePath, as it's columns.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath
IN	cardinalityMin	INTEGER
IN	cardinalityMax	INTEGER
IN	cardinalityExpected	INTEGER
IN	gatherEnabled	VARCHAR(20)
IN	maxTime	INTEGER
IN	columnSettings	VECTOR(ROW(name VARCHAR(1000), flags VARCHAR(1000), columnMin DOUBLE, columnMax DOUBLE,

Direction	Parameter Name	Parameter Type
		columnDistinct DOUBLE))
OUT	responseXML	XML
OUT	faultXML	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/ds_orders/orders'
IN	cardinalityMin	36
IN	cardinalityMax	36
IN	cardinalityExpected	36
IN	gatherEnabled	'DEFAULT'
IN	maxTime	-1
IN	columnSettings	VECTOR[…]
OUT	responseXML	<XML>
OUT	faultXML	NULL

updateTrigger

Update the definition of a Trigger resource that invokes a SQL Procedure containing a parameter.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	fullResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	procedurePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	procedureParam	VARCHAR(255)
OUT	success	BIT
OUT	createResponse	XML
OUT	faultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	fullResourcePath	'/shared/ASAssets/Utilities/repository/examples/source/trigger1'
IN	procedurePath	'/shared/ASAssets/Utilities/repository/examples/source/proc1'
IN	procedureParam	'string'
OUT	success	1
OUT	createResponse	XML not shown here
OUT	faultResponse	XML not shown here

updateXsltTransformationProcedure

This procedure is used to update a custom XSLT transformation procedure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	transformSourceType	VARCHAR
IN	xsltText	VARCHAR
IN	xsltModel	RepositoryDefinitions.XsltModelVectorType
IN	annotation	LONGVARCHAR
IN	isExplicitDesign	BIT
IN	parameterVector	RepositoryDefinitions.AttributeCompleteVectorType
IN	attributeVector	RepositoryDefinitions.AttributeCompleteVectorType
OUT	Success	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/examples/customXSLT'
IN	transformSourcePath	'/shared/examples/ds_XML/productCatalog.xml'
IN	transformSourceType	'TREE'
IN	xsltText	NULL or XSLT text
IN	xsltModel	NULL
IN	annotation	'Product catalog transformation'

Direction	Parameter Name	Parameter Value
IN	isExplicitDesign	0
IN	parameterVector	NULL
IN	attributeVector	NULL
OUT	success	1

RepoUtils

This section describes the custom java procedure 'RepoUtils' which contains several repository lookup and manipulation utilities.

RepoUtils/applyReservedListToPath

This CJP is used to fix the leading characters and CIS reserved words used in a folder path. Any path that contains a reserved word, leading underscore '_' or a number '0123456789', or a special character must have double quotes inserted around that portion of the folder. This procedure would be called in conjunction with other procedures. For example, when generating a view based off of another view, the SELECT statement's FROM clause would require that the path to the underlying view be fixed with double quotes if it meets any of the quoting criteria above. This CJP uses $CIS_HOME/conf/customjars/RepoUtils.properties to store the quoting rules and reserved words as regular expressions. The properties file can be updated at any time and this CJP will pick up the change without requiring a CIS restart.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath – path to a resource	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	debug – Y/N or T/F	CHAR(1)
OUT	fixedResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType

2. **Examples:**

2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
IN	resourcePath	'/shared/tmp/1folder/_folder/XML/local'
IN	debug	'N'
OUT	fixedResourcePath	'/shared/tmp/"1folder"/"_folder"/"XML"/"local"'

RepoUtils/applyReservedListToWord

This CJP is used to fix the leading characters and CIS reserved words used in a word. It is assumed that a word is really a portion of a path representing the value in between two slashes '/'. Any word that contains a reserved word, leading underscore '_' or a number '0123456789', or a special character must have double quotes inserted around it. This procedure would be called in conjunction with other procedures. For example, when generating a view based off of another view, the SELECT statement's FROM clause would require that the path to the underlying view be fixed with double quotes if it meets any of the quoting criteria above. This CJP uses $CIS_HOME/conf/customjars/RepoUtils.properties to store the quoting rules and reserved words as regular expressions. The properties file can be updated at any time and this CJP will pick up the change without requiring a CIS restart.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	resourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType
IN	debug	CHAR(1)
OUT	fixedResourcePath	/shared/ASAssets/Utilities/TypeDefinitions.pathType

2. **Examples:**

2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
IN	word	‘XML’
IN	debug	‘N’
OUT	result	‘”XML”’

RepoUtils/EncryptPassword (Custom Function)

This CJP encrypts an input string using Composite’s password encryption API. This uses the TEAV encryption algorithm.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inString	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	inString	‘P4ssw0rd’
OUT	result	‘5AC41E7EC9AC80CE’

RepoUtils/ForceWriteRepoUtils

Forces a new RepoUtils.properties file to be written out to the CIS customjars folder.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	BOOLEAN

2. **Examples:**

2.1. **Assumptions:**

Direction	Parameter Name	Parameter Value
OUT	result	true

RepoUtils/GetAnsi2NativeMapping

Given a Composite (ANSI) data type and a path to a data source, this procedure returns the data source's data type equivalent.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datasourcePath	VARCHAR(2147483647)
IN	cisType	VARCHAR(2147483647)
OUT	result	CURSOR(cisType VARCHAR(2147483647), cisNormalizedType VARCHAR(2147483647), cisBaseType VARCHAR(2147483647), cisScale INTEGER, cisPrecision INTEGER, dataTypeId INTEGER, dataTypeName VARCHAR(2147483647), nativeType VARCHAR(2147483647), nativeBaseType VARCHAR(2147483647), nativeScale INTEGER, nativePrecision INTEGER)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	datasourcePath	'/shared/examples/ds_orders'
IN	cisType	'LONGVARCHAR'
OUT	result	('LONGVARCHAR', 'LONGVARCHAR', 'LONGVARCHAR', -1, -1, -983, 'LONGVARCHAR', 'varchar(2147483647)', 'varchar', '2147483647', '-1')

RepoUtils/getReservedWordList

This CJP retrieves the reserved word list stored in $CIS_HOME/conf/customjars/RepoUtils.properties The properties file can be updated at any time and this CJP will pick up the change without requiring a CIS restart.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR (ReservedWord LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
OUT	result	{('abs'), ('absolute'),('acos'), ('action'), …}

RepoUtils/GetSystemProperties

Returns all the properties defined in the JVM running the current instance of CIS.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR(Property VARCHAR(2147483647), Value VARCHAR(2147483647)

Direction	Parameter Name	Parameter Type
		)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	(‘path.separator’, ‘;’), (‘user.home’, ‘C:\Users\bob’), …

RepoUtils/GetUserGroups

Returns all the groups that a user belongs to. This functionality isn’t available in the administrative web services API, so it can only be done from within Java itself.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	DomainName	VARCHAR(2147483647)
IN	UserName	VARCHAR(2147483647)
OUT	result	CURSOR(DomainName VARCHAR(2147483647), GroupName VARCHAR(2147483647))

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	DomainName	‘composite’
IN	UserName	‘admin’
OUT	result	(‘composite’, ‘all’), (‘composite’, ‘admin’)

RepoUtils/isReservedWord (Custom Function)

This CJP is used to detect whether a string is a reserved word (or would otherwise need quoting when referenced as part of a resource path.) It is assumed that a word is really a portion of a path representing the value in between two slashes '/'. Any word that contains a reserved word, leading underscore '_' or a number '0123456789', or a special character must have double quotes inserted around it. This procedure would be called in conjunction with other procedures. This

CJP uses $CIS_HOME/conf/customjars/RepoUtils.properties to store the quoting rules and reserved words as regular expressions. The properties file can be updated at any time and this CJP will pick up the change without requiring a CIS restart.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inWord	System.Text (VARCHAR(2147483647))
OUT	fixedResourcePath	BOOLEAN

2. **Examples:**

   2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
IN	word	‘XML’
OUT	result	TRUE

RepoUtils/UpdateDsColumnAnnotation

This procedure is used to update annotations for data source table columns since there is no Admin API this particular operation. For other types of table/view columns, please use the updateSqlTable() admin API.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	column_path	LONGVARCHAR
IN	annotation	LONGVARCHAR
OUT	result	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: Dependency on** configureReservedList

Direction	Parameter Name	Parameter Value
IN	column_path	'/shared/examples/ds_orders/orders/OrderID'
IN	Annotation	'OrderID column annotation'
OUT	Result	'Column annotation updated.'

CIS Repository Definition Procedures

This section describes each 'CIS Repository' definition procedures. The repository definitions provide a global set of type definitions across all of the API's. The definitions are broken down into two files: one for recursive type definitions and one for non-recursive type definitions.

definitions/RepositoryDefinitions

Provides global type definitions for the procedures in the /shared/ASAssets/Utilities/repository directory.

1. **Parameters: none**

2. **Examples: none**

definitions/RepositoryDefinitionsRecursive

Provides global type definitions for the recursive procedures in the /shared/ASAssets/Utilities/repository directory.

If an ***error*** is encountered in the 'RepositoryDefinitionsRecursive' file and you get a java memory error when trying to save this file, it is because CIS cannot save the file due to the recursive nature of the procedure 'getResourceTreeList'.

Go to 'getResourceTreeList' and read the comments in the procedure body regarding the temporary commenting of the recursive section until you can properly save the 'RepositoryDefinitionsRecursive' file. Once you save this file, you can then uncomment and save the 'getResourceTreeList' procedure.

1. **Parameters: none**

2. **Examples: none**

CIS Repository Execute Procedures

This section describes procedures that are used for executing code inside the CIS engine.

execute/executeProcedure

This procedure calls a procedure in either a blocking or non-blocking manner. It does not return a result set, but is used merely for functional or performance testing. Procedure inputs should be specified in the `resourceParams` field. Each parameter should be specified using the format <type>=<value> and separated by the '|' character. For example, procedure that takes a single INTEGER input might have a `resourceParams` value of 'INTEGER=1'. However, a procedure that takes a VARCHAR and an INTEGER as input parameters might have a `resourceParams` value of 'VARCHAR='abc'|INTEGER=1'.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	blocking	BIT
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceParams	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	blocking	1
IN	resourcePath	'/shared/examples/LookupProduct'
IN	resourceParams	'INTEGER=1'

execute/executeProcedureResults

This procedure calls a procedure in a blocking manner and returns a result set. Procedure inputs should be specified in the `resourceParams` field. Each parameter should be specified using the format <type>=<value> and separated by the '|' character. For example, procedure that takes a single INTEGER input might have a `resourceParams` value of 'INTEGER=1'. However, a procedure that takes a VARCHAR and an INTEGER as input parameters might have a `resourceParams` value of 'VARCHAR='abc'|INTEGER=1'. Output will be returned in either of the two output parameters, depending on whether the procedure being called returns scalar values or a cursor. The unused output value will be NULL.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceParams	LONGVARCHAR
OUT	outputScalarResultResponse	XML
OUT	outputCursorResultResponse	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	resourcePath	'/shared/examples/LookupProduct'
IN	resourceParams	'INTEGER=1'
OUT	outputScalarResultResponse	NULL
OUT	outputCursorResultResponse	XML not shown here

CIS Repository Server Procedures

This section describes procedures that are used for working with server settings. Calling user will need READ_ALL_CONFIG and/or MODIFY_ALL_CONFIG rights to make use of these.

server/addLicense

This procedure adds a license key to the local CIS instance.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	licenseText	VARCHAR(4096)

2. **Examples:**

   2.1. **Assumptions: Calling user has MODIFY_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	licenseText	XML not shown here

server/getServerAttribute (Custom Function)

This procedure retrieves the attribute value at the specified attribute path. This procedure only returns values for simple attribute types like STRING, INTEGER, etc. See `server/getServerAttributeList` and `server/getServerAttributeMap` for retrieving more complex attribute types.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	attributePath	LONGVARCHAR
OUT	keyValue	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	absolutePath	'/server/webservices/timezoneBaseOnGMT'
OUT	keyValue	'true'

server/getServerAttributeList

This procedure retrieves the attribute list value at the specified attribute path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	attributePath	LONGVARCHAR
OUT	result	CURSOR (keyName LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	absolutePath	'/studio/data/examplelist'
OUT	result	(no rows)

server/getServerAttributeMap

This procedure retrieves the attribute map value at the specified attribute path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	attributePath	LONGVARCHAR
OUT	result	CURSOR (keyName LONGVARCHAR, keyValue LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	absolutePath	'/studio/data/examplemap'

Direction	Parameter Name	Parameter Value
OUT	result	(no rows)

server/getServerAttributeMapByKey (Custom Function)

This procedure retrieves the attribute value at the specified attribute map path and map key.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	attributePath	LONGVARCHAR
IN	keyName	LONGVARCHAR
OUT	keyValue	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	absolutePath	'/studio/data/examplemap'
IN	keyName	'this_key_name_does_not_exist'
OUT	result	NULL

server/updateServerAttribute

This procedure updates an attribute value at the specified attribute path. Attributes types can be 'STRING', 'BOOLEAN', 'LIST', or 'MAP'. Examples of attribute list and map values in XML appear below:

Example of LIST attribute value:

```
<common:item>
      <common:type>STRING</common:type>
      <common:value>a1</common:value>
</common:item>
<common:item>
      <common:type>STRING</common:type>
      <common:value>b2</common:value>
</common:item>
```

Example of MAP attribute value:

```
<common:entry>
      <common:key>
            <common:type>STRING</common:type>
            <common:value>Environment</common:value>
      </common:key>
      <common:value>
            <common:type>STRING</common:type>
            <common:value>DEV1</common:value>
      </common:value>
</common:entry>
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	ATTR	VARCHAR(1024)
IN	ATTR_TYPE	VARCHAR(1024)
IN	NEWVAL	VARCHAR(1024)

2. **Examples:**

   2.1. **Assumptions: Calling user has MODIFY_ALL_CONFIG right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	ATTR	'/server/sql/language/caseSensitive'
IN	ATTR_TYPE	'BOOLEAN'
IN	NEWVAL	'true'

CIS Repository User Procedures

This section describes procedures that are used for managing users and groups inside the CIS engine. Calling user will need MODIFY_ALL_USERS right to make use of most of these.

user/createGroup

This procedure creates a group in the specified domain. Group privileges are a space separated list of the global rights a group should have: ACCESS_TOOLS, UNLOCK_RESOURCE, READ_ALL_RESOURCES, MODIFY_ALL_RESOURCES, etc. See /services/webservices/system/admin/user/operations/UserSchema for the full list.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	groupName	VARCHAR(255)
IN	groupDomain	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	groupPrivileges	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: Calling user has MODIFY_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	groupName	'myDevGroup'
IN	groupDomain	'composite'
IN	groupPrivileges	'ACCESS_TOOLS READ_ALL_STATUS'

user/createResourcePrivilege

This procedure adds or updates privileges on a resource for the specified user or group. Calling user must have the ability to reassign ownership and privileges on the resource. Valid values:

recurse: '1' or '0'

nameType: 'GROUP' or 'USER'

privileges: space separated list of one or more of READ, WRITE, SELECT, INSERT, UPDATE, DELETE, and GRANT.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	owner	VARCHAR(255)
IN	ownerDomain	VARCHAR(255)
IN	resourcePath	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096))
IN	resourceType	/lib/resource/ResourceDefs.ResourceType (VARCHAR(40))
IN	recurse	CHAR(1)
IN	name	VARCHAR(255)
IN	domainName	VARCHAR(255)
IN	nameType	VARCHAR(255)
IN	privileges	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: Calling user has appropriate privileges and/or rights**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	owner	'admin'
IN	ownerDomain	'composite'
IN	resourcePath	'/shared/examples/CompositeView'
IN	resourceType	'TABLE'
IN	recurse	'0'
IN	name	'all'
IN	domainName	'composite'
IN	nameType	'GROUP'
IN	privileges	'READ SELECT'

user/createUser

This procedure creates a user in the specified domain.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	username	VARCHAR(255)
IN	password	VARCHAR(255)
IN	forcePassword	CHAR(1)
IN	domainName	VARCHAR(255)
IN	groupNameAndDomainList	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: Calling user has MODIFY_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	username	'bob'
IN	password	'password'
IN	forcePassword	'0'
IN	domainName	'composite'

Direction	Parameter Name	Parameter Value
IN	groupNameAndDomainList	'bobsgroup|composite dummygroup|composite'

user/deleteGroup

This procedure deletes a group in the specified domain.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	groupName	VARCHAR(255)
IN	groupDomain	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: Calling user has MODIFY_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	groupName	'myDevGroup'
IN	groupDomain	'composite'

user/deleteUser

This procedure deletes a group in the specified domain.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1), either 'Y' or 'N'
IN	userName	VARCHAR(255)
IN	domainName	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: Calling user has MODIFY_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	userName	'bob'
IN	domainName	'composite'

user/getDomainGroups

This procedure retrieves all the groups in the specified domain.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inDomainName	VARCHAR(255)
OUT	result	CURSOR (name VARCHAR(32768), domainName VARCHAR(32768), id INTEGER, explicitRights VARCHAR(32768), effectiveRights VARCHAR(32768), inheritedRights VARCHAR(32768), annotation VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	inDomainName	'composite'
OUT	result	('admin', 'composite', 1, 'ACCESS_TOOLS, …', 'ACCESS_TOOLS, …', 'NONE', 'Administrator group'), …

user/getDomainUsers

This procedure retrieves all the users in the specified domain.

- `repository/lowerLevelProcedures/getDomainsXSLT` - this performs the actual invocation to the CIS repository API and returns a transformed XML list of domains in tabular format.
- `repository/lowerLevelProcedures/getDomainUsersXSLT` – this performs the actual invocation to the CIS repository API and returns a transformed XML list of domain users in tabular format.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inDomainName	VARCHAR(255)
OUT	result	OUT results PIPE (name VARCHAR, domainName VARCHAR, id INTEGER, explicitRights VARCHAR(1024), effectiveRights VARCHAR(1024), inheritedRights VARCHAR(1024), annotation VARCHAR(1024), isLocked VARCHAR, groupName VARCHAR, groupDomain VARCHAR)

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	inDomainName	'composite'
OUT	result	('admin', 'composite', -1973, 'ACCESS_TOOLS, …', 'ACCESS_TOOLS, …', 'NONE', 'Administrator user', 'N', 'all', 'composite',), …

user/getGroup

This procedure determines the existence of a group and returns its global rights.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	groupName	VARCHAR(255)
IN	groupDomain	VARCHAR(255)
OUT	groupExists	BIT

Direction	Parameter Name	Parameter Type
OUT	groupExplicitRights	VARCHAR(255)
OUT	groupEffectiveRights	VARCHAR(255)
OUT	groupInheritedRights	VARCHAR(255)

2. **Examples:**

   2.1. **Assumptions: Calling user has READ_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	groupName	'admin'
IN	groupDomain	'composite'
OUT	groupExists	1
OUT	groupExplicitRights	'ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE'
OUT	groupEffectiveRights	'ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE'
OUT	groupInheritedRights	'NONE'

user/getUser

This procedure determines the existence of a user and returns its global rights.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	userName	VARCHAR(255)
IN	userDomain	VARCHAR(255)

Direction	Parameter Name	Parameter Type
OUT	userExists	BIT
OUT	userExplicitRights	VARCHAR(255)
OUT	userEffectiveRights	VARCHAR(255)
OUT	userInheritedRights	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: Calling user has READ_ALL_USERS right**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	userName	'admin'
IN	userDomain	'composite'
OUT	userExists	1
OUT	userExplicitRights	'ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE'
OUT	userEffectiveRights	'ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE'
OUT	userInheritedRights	'ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE'

17 How To Use ‘Request’ Procedures

Introduction

This section describes the routines for examining the SQL of calling requests.

terminateIdleSessions

This procedure terminates any ODBC/JDBC/ADO.NET/HTTP/HTTPS sessions that have been idle for longer than the specified input (in minutes). The caller must have ACCESS_TOOLS and MODIFY_ALL_STATUS rights.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	idleMinutes	INTEGER

OUT	result	CURSOR (
		    session_id BIGINT,
		    success BIT,
		    faultXML XML
		)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	idleMinutes	5
OUT	result	(12345, 1, NULL), …

terminateRequest

This procedure terminates a request given a request ID. The calling user must have the ACCESS_TOOLS and MODIFY_ALL_STATUS global rights.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	requestID	VARCHAR
OUT	success	BIT
OUT	responseXML	XML
OUT	faultXML	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	requestID	'12345'
OUT	success	1
OUT	responseXML	<XML>
OUT	faultXML	[NULL]

terminateSession

This procedure terminates a session given a session ID. The calling user must have the ACCESS_TOOLS and MODIFY_ALL_STATUS global rights.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sessionID	VARCHAR
OUT	success	BIT
OUT	responseXML	XML
OUT	faultXML	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sessionID	'12345'
OUT	success	1
OUT	responseXML	<XML>
OUT	faultXML	[NULL]

RequestUtils

This section will show how to use the ‘Request’ CJP procedures.

RequestUtils/DirectSqlRequest (Custom Function)

Walks the stack of requests that resulted in the call to this CJP and returns the SQL of this CJP's immediate parent request (or lowest level request that generated an SQL statement.) Note that this CJP does not return code for procedure requests.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	requestSrc	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	requestSrc	'SELECT * FROM /shared/examples/procedure_that_directly_calls_DirectSqlRequest()'

RequestUtils/OriginalRequest (Custom Function)

Walks the stack of requests that resulted in the call to this CJP and returns the original client request (SQL statement or procedure call.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	requestSrc	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	requestSrc	'SELECT * FROM myCat.mySchema.published_procedure_that_eventually_calls_OriginalRequest()'

RequestUtils/ReadInEqClause (Custom Function)

Accepts SQL text (as returned by one of the RequestUtils) and a column name. The SQL text is parsed and scanned for any expressions like 'column name' = 'val' or 'column name' IN (val1, val2, ..valX). All the values found are returned in a cursor.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sql	VARCHAR(2147483647)
IN	columnName	VARCHAR(2147483647)

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR(value VARCHAR(2147483647))

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sql	'SELECT * FROM published_view WHERE mycol IN (1, 2, 3)'
IN	columnName	'mycol'
OUT	result	('1'), ('2'), ('3')

RequestUtils/TopSqlRequest (Custom Function)

Walks the stack of requests that resulted in the call to this CJP and returns the SQL of the original client request (or highest level request that generated an SQL statement, e.g. if the original request is a procedure, then TopSqlRequest will return the first SQL request in the chain.) Note that this CJP does not return code for procedure requests.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	requestSrc	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	requestSrc	'SELECT * FROM /shared/examples/procedure_that_eventualy_calls_TopSqlRequest()'

18 How To Use ‘String’ Procedures

Introduction

This section will show how to use the ‘String’ manipulation procedures.

addQuotesInList (Custom Function)

Wraps the values in a CSV string in quotes. The function decomposes the list into its constituent values using the input delimiter as a separator and then reconstructs the CSV using the input delimiter and spacing requirements as specified by the input parameters.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inString	LONGVARCHAR
IN	quoteType	CHAR(1) (either ' or ")
IN	delimiter	VARCHAR
IN	trimList	INTEGER (1 or 0)
IN	numSpacesAfterDelim	INTEGER
OUT	modifiedString	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inString	'a, b, c'
IN	quoteType	'"'
IN	delimiter	','
IN	trimList	1
IN	numSpacesAfterDelim	1
OUT	modifiedString	'"a", "b", "c"'

basename (Custom Function)

Returns the resource name from an absolute resource path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
OUT	outputString	/lib/resource/ResourceDefs.ResourceName (VARCHAR(255) as of CIS 5.1)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	'/shared/examples/CompositeView'
OUT	outputString	'CompositeView'

concatNotNull (Custom Function)

Concatenate the two strings together according to the mode parameter:

mode	Action
0 / NULL	Return null string when either inputString1 (part1) or inputString2 (part2) or both is null.
1	Replace null with blank ('') in inputString1 (part1) or inputString2 (part2) so that a string is returned no matter what.
2	Return empty string when inputString2 is null. This is a prefix example.
3	Return empty string when inputString1 is null. This is a suffix example.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString1	LONGVARCHAR
IN	inputString2	LONGVARCHAR
IN	mode	INTEGER
OUT	outputString	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString1	'a'
IN	inputString2	NULL
IN	mode	1
OUT	outputString	'a'

dirname (Custom Function)

Returns the resource's parent folder from an absolute resource path.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)
OUT	outputString	/lib/resource/ResourceDefs.ResourcePath (VARCHAR(4096) as of CIS 5.1)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	'/shared/examples/CompositeView'
OUT	outputString	'/shared/examples'

emptyStr (Custom Function)

Return a blank if input is null or blank – used extensively when writing to the log – null strings result in null output which is not useful. If the string is not blank, then return the original string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inputString	LONGVARCHAR
OUT	outputString	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	null

Direction	Parameter Name	Parameter Value
OUT	outputString	‘’

entityConstants

This procedure is used by "entityExtract" to perform keyword/entity extraction on text. In this procedure are 4 vectors defined as constants that feed into `string/entityExtract()`. Review these vectors and make modifications as required to adjust what gets stripped out of the text and what simply gets ignored and not returned as a keyword entity.

entityExtract

Take an incoming string and return a vector of keyword entities except common words like “a, an, the, of” etc.

1. **Parameters:**

   1.1. **Assumptions:** Depends on 4 vectors defined as constants define in the procedure `string/entityConstants()`. Review that procedure and make modifications as required.

      1.1.1. **symbols1Vector –** Single value symbols and punctuation such as '!,@,#,$,%' and etc. that get replaced with a non-null, empty character ''.

      1.1.2. **symbols2Vector –** Multi-character value symbols such as ' - ' that get replaced with a single space, ' '.

      1.1.3. **symbols3Vector –** Hidden character symbols such as tabs that get replaced with a single space, ' '.

      1.1.4. **nonEntityVector –** non-Entity based words that are not extracted from the incoming text. Example include: ‘a, and, the, of, this, that’ and many more.

Direction	Parameter Name	Parameter Type
IN	inText any text.	LONGVARCHAR
IN	inDelimiter space, pipe or some single character delimiter	CHAR(1)
IN	inNumWords the number of words to produce. If 0 or null then produce all words.	INTEGER
IN	random true(1)=random-produce a list randomly	BIT

Direction	Parameter Name	Parameter Type
	from the incoming text false(0)=sequential-produce a list sequentially from the incoming text	
IN	removeSymbols true(1)=remove symbols and punctuation prior to extraction false(0)=do not remove symbols and punctuation prior to extraction	BIT
OUT	entityVector	VECTOR(VACHAR)

2. **Examples:**

Direction	Parameter Name	Parameter Value
IN	inText	'A long string of text'
IN	inDelimiter	' '
IN	inNumWords	0
IN	random	0
IN	removeSymbols	1
OUT	entityVector	{long,string,text}

entityExtractToPipe

Invoke `string/entityExtract()` and return the keywords in a pipe cursor.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inText any text.	LONGVARCHAR
IN	inDelimiter space, pipe or some single character delimiter	CHAR(1)
IN	inNumWords the number of words to produce. If 0 or null then produce all words.	INTEGER
IN	random true(1)=random-produce a list randomly from the incoming text	BIT

Direction	Parameter Name	Parameter Type
	false(0)=sequential-produce a list sequentially from the incoming text	
IN	removeSymbols true(1)=remove symbols and punctuation prior to extraction false(0)=do not remove symbols and punctuation prior to extraction	BIT
OUT	keywordsPipe	Pipe (keyword LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inText	'A long string of text'
IN	inDelimiter	' '
IN	inNumWords	0
IN	random	0
IN	removeSymbols	1
OUT	keywordsPipe	long string text

entityExtractToString (Custom Function)

Invoke `string/entityExtract()` and return the keywords in a single comma separate string variable.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inText any text	LONGVARCHAR
IN	inDelimiter space, pipe or some single character delimiter	CHAR(1)
IN	inNumWords the number of words to produce. If 0 or null then produce all words.	INTEGER

Direction	Parameter Name	Parameter Type
IN	random true(1)=random-produce a list randomly from the incoming text false(0)=sequential-produce a list sequentially from the incoming text	BIT
IN	removeSymbols true(1)=remove symbols and punctuation prior to extraction false(0)=do not remove symbols and punctuation prior to extraction	BIT
OUT	outWords comma separated list of words	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inText	'A long string of text'
IN	inDelimiter	' '
IN	inNumWords	0
IN	random	0
IN	removeSymbols	1
OUT	outWords	long, string, text

escapeCSV (Custom Function)

Looks for separators or qualifiers in a string and escapes them if present.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inStr	LONGVARCHAR
IN	separator	VARCHAR(1)
IN	qualifier	VARCHAR(1)
OUT	outputString	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	‘john, paul, ringo, ”bob”, and george’
IN	separator	‘,’
IN	qualifier	‘”’
OUT	outStr	‘”john, paul, ringo, ””bob””, and george”’

extractBiDelimitedText (Custom Function)

Extract bi-delimited text refers to the ability to locate text based on a search term where the search term encloses the sought after text in a beginning and ending delimiter thus bi-delimited. Where this can be useful is when you have to throw an exception and you want to embed both the custom error code and error message in the exception: raise ex VALUE 'Custom exception: ERROR_CODE(C1) ERROR_MESSAGE(Error is foo.)'. This allows you to have a generic error processing routine that can extract specific error codes and values. Of course this is not limited to that. You can be very creative in the usage of this routine.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inText	LONGVARCHAR
IN	searchTerm a word with no spaces that represents the keyword search term to locate	VARCHAR
IN	openingDelim The beginning delimiter which directly follows the searchTerm. Allowed: Single: '[', '(', '{', '<' Doubles: '[[', '((', '{{', '<<'	VARCHAR
IN	closingDelim The ending delimiter which directly follows the content. Allowed: Single: ']', ')', '}', '>' Doubles: ']]', '))', '}}', '>>'	VARCHAR
IN	inOccurrence The value of the occurrence input value determines which occurrence to return (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.) If a NULL value or zero is passed in for occurrence, a default of 1 is used.	INTEGER
IN	trimText 0=do not trim result, 1=do trim result text (default=0)	INTEGER

Direction	Parameter Name	Parameter Type
IN	caseSensitive 0=no search term case sensitivity, 1=case sensitive search term.	INTEGER
OUT	result	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inText	'john, paul, ringo, "bob", and george'
IN	searchTerm	','
IN	openingDelim	'"'
IN	closingDelim	
IN	inOccurrence	
IN	trimText	
IN	caseSensitive	
OUT	result	

extractDelimitedText

This is a generic procedure for extracting values from a list using a single or multi-character delimiter. The results are returned as a cursor of objects. This does not use the expensive REGEX function or VECTORS for parsing the delimited text. It does not parse character by character. It uses a more efficient INSTR and SUBSTRING to extract the results.

Example Usage:

1. Extract the last object in a DV path:

   inputString=/a1/b2/c3/d4

   delimiter=/

   occurrence=1

   reverseInputText=1

   reverseOutput=1

   trimText=1

   includeDelimiter=0

   result: d4

2. Extract a comma-separated list

   inputString=a1,b2,c3,d4

```
delimiter=,
occurrence=0
reverseInputText=0
reverseOutput=0
trimText=1
includeDelimiter=0
result:  a1
         b2
         c3
         d4
```

3. Extract the 4th object [database] in a DV path:

```
    object: 1         2              3  4         5         6
inputString=/services/databases/db/catalog/schema
delimiter=/
occurrence=4
reverseInputText=0
reverseOutput=0
trimText=1
includeDelimiter=0
result:  db
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**inputString** - A text string containing a list with delimeters to be extracted into a cursor of output.	LONGVARCHAR
IN	**delimiter** - 1 or more characters such as single characters: space , | / or double characters such as || or any repeatable combination.	VARCHAR
IN	**occurrence** - 0=all occurrences, n>0=the occurrence of delimited text or null if the occurrence requested is not applicable.	SMALLINT
IN	**reverseInputText** - 0=do not reverse input text, 1=reverse input text before parsing	SMALLINT
IN	**reverseOutput** - 0=do not reverse output text, 1=reverse output text before returning.	SMALLINT
IN	**trimText** - 0=do not trim, 1=do trim result	SMALLINT

Direction	Parameter Name	Parameter Type
IN	**includeDelimiter -** 0=exclude delimiter from output, 1=include delimiter in output	SMALLINT
OUT	**result** – cursor of objects. A cursor of extracted text. If no delimiter was found, then the original input text is returned. If the delimiter appears at the beginning of the text before any other characters, an empty row is output and counts as 1 occurrence. For example: /a/b/c/d with delimiter=/ and occurrence=2 would result in 'a' being returned.	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inputString	‘john || paul || ringo || ”bob” || george’
IN	delimiter	‘,’
IN	occurrence	0
IN	reverseInputText	0
IN	reverseOutput	0
IN	trimText	1
IN	includeDelimiter	0
OUT	result	‘john’ ‘paul’ ‘ringo’ ‘”bob”’ ‘george’

extractTextList

The extractTextList is used to extract a separated list of values containing embedded separators within double quotes, single quotes. The result is returned as a cursor based on the boundaries of the the qualifiers: double quotes, single quotes or paired parenthesis. The separator value is preserved within the qualifier if the flag for that qualifier is set to 1 (true).

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	textList	LONGVARCHAR

Direction	Parameter Name	Parameter Type
IN	Separator The separator value (typically a comma) that will be used to define the boundary of text expressions. Becuase this uses regex, any special separators need to be escaped with a \ character. It is not necessary to escape a comma separator in regex. The following are potential use cases and are shown within brackets [] to better show the use of spaces: [\ ] - this is used to split text on a space such as finding all the words in a sentence. [\\] - this is a backslash separator escaped with a backslash [\^] - this is a caret separator escaped with a backslash.	VARCHAR
IN	preserveDoubleQuotes 1 or 0/null (default). Indicates whether to preserve the context of commas within the boundaries of a double quoted qualifier string.	BIT
IN	preserveSingleQuotes 1 or 0/null (default). Indicates whether to preserve the context of commas within the boundaries of a single quoted qualifier string.	BIT
IN	preserveParenthesis 1 or 0/null (default). Indicates whether to preserve the context of commas within the boundaries of left and right parenthesis qualifier pairs.	BIT
IN	preserveQualifier 1/null (default) or 0. In this case, the default is to preserve the qualifer value on output. The qualifiers may be double quotes, single quotes or left and right parenthesis. If set to 0 (do not preserve), the qualifiers are only remmoved if they exist as pairs in the first and last characters and the length of the text being returned is at least 2 characters. Otherwise if the above conditions are not met, any attempt to remove embedded qualifiers will not be completed. The assumption is that qualifiers exist at the boundaries of the comma separator such as "orders,customers", orders which would yield: "orders,customers" orders	BIT

Direction	Parameter Name	Parameter Type
	This example would not remove the qualifier: text "more text" text,text text "more text" text text	
IN	trimResults 1 or 0/null (default) - if set to 1, then trim the results of any white space otherwise do not.	BIT
OUT	result	PIPE(textExpression LONGVARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	textList	"orders,customers",orders,customers
IN	Separator	‘,’
IN	preserveDoubleQuotes	1
IN	preserveSingleQuotes	1
IN	preserveParenthesiss	1
IN	preserveQualifier	null
IN	trimResults	null
OUT	result	"orders,customers" orders customers

findOpenClosePair

Search through the sql script to find the corresponding left open and right close pairs. This script was originally developed to parse through SQL text to find functions. Only a single character value is supported for these delimiters. Valid pairs include: (), [], {}, <>, /\

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sqlScript	LONGVARCHAR
IN	begPos	INTEGER
IN	openingDelim	VARCHAR

Direction	Parameter Name	Parameter Type
IN	closingDelim	VARCHAR
IN	trimText	INTEGER
OUT	openingDelimPos	INTEGER
OUT	closingDelimPos	INTEGER
OUT	extractedScript	LONGVARCHAR
OUT	extractedScriptBefore	LONGVARCHAR
OUT	extractedScriptAfter	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sqlScript	'P1(P2(a))'
IN	begPos	1
IN	openingDelim	(
IN	closingDelim	)
IN	trimText	1
OUT	openingDelimPos	3
OUT	closingDelimPos	17
OUT	extractedScript	'P2(a)'
OUT	extractedScriptBefore	'P1'
OUT	extractedScriptAfter	''

findString (Custom Function)

Given two input strings, this function returns an integer value representing the starting position of the first string within the second string. The third parameter indicates which direction to begin searching. When 'F', begin searching forward from the beginning of the second string. When 'R', begin search in reverse from the end of the second string.

- This function is case-sensitive.
- All string types, all numeric types, and all data types are accepted as input arguments.
- The output is always an integer provided none of the input strings is NULL. Otherwise, NULL is returned.
- If any of the arguments is NULL, the function returns NULL.
- If the first argument is a blank string, the function returns 1 (one).

- If the first argument is not found within the second argument, the function returns 0 (zero).
- A blank non-null string (‘’) for the search string always produces 1 for an output.

Examples:

```
direction='F'
        POSITION(‘it’ IN ‘case-sensitivity’),
                Output: 10          ^
        POSITION(‘’ IN ‘mistake’)
                Output: 1
        POSITION(‘no’ IN ’yes’)
                Output: 0
direction='R'
        POSITION(‘it’ IN ‘case-sensitivity’)
                Output: 14          ^
        POSITION(‘no’ IN ’yes’)
                Output: 0
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	searchString	LONGVARCHAR
IN	stringToSearch	LONGVARCHAR
IN	direction	CHAR(1)
OUT	pos	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	searchString	‘case-sensitivity’
IN	stringToSearch	‘it’
IN	direction	‘R’
OUT	pos	14

findStringInList (Custom Function)

Given two input strings, this function returns an integer value representing the field position of the first string within the second delimited string. The third parameter indicates the delimiter string to use. The value of `pos` is 1-based (0 indicates the `searchString` parameter was not found.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	searchString	LONGVARCHAR
IN	stringToSearch	LONGVARCHAR
IN	delimiter	VARCHAR
OUT	pos	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	searchString	‘csv’
IN	stringToSearch	‘my,csv,value’
IN	delimiter	‘,’
OUT	pos	2

fixQuotes (Custom Function)

Turns single quotes into two single quotes so that when you are constructing a dynamic SQL statement this procedure insures that values with quotes in them are fixed to be double quoted and the SQL statement will execute correctly.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inMessage	LONGVARCHAR
OUT	outMessage	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inMessage	this is a 'quoted' string
OUT	outMessage	this is a "quoted" string

getConstant (Custom Function)

This procedure gets a constant value from a dynamic constant path. All constants must be of type VARCHAR. If the value is actually an integer THEN the application must take care of casting to the proper value.

The constant path should be a procedure that outputs the named constant as a scalar value.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	constantPath	TypeDefinitions.pathType
IN	constantName	VARCHAR(255)
OUT	outValue	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	constantPath	'/shared/ASAssets/Utilities/documentation/constants'
IN	constantName	'resourcePath'
OUT	outValue	'/shared'

getDelimitedOccurrence (Custom Function)

Given a delimited string, this procedure will return the value at a specified field number. Searches may start at the beginning (`mode` value of 'F') or from the end (`mode` value of 'R') of the string.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inText	LONGVARCHAR
IN	mode	CHAR(1)
IN	delimiter	VARCHAR
IN	inOccurrence	INTEGER
IN	inTrimText	INTEGER
OUT	result	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inText	'my,csv,value'
IN	mode	'F'
IN	delimiter	‘,’
IN	inOccurrence	2
IN	inTrimText	0

Direction	Parameter Name	Parameter Value
OUT	result	'csv'

getDelimitedSum (Custom Function)

Given a delimited string, this procedure will return the sum of the delimited fields starting at a specified field position.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inText	LONGVARCHAR
IN	delimiter	VARCHAR
IN	inSstartingOccurrence	INTEGER
OUT	result	DECIMAL(32,2)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inText	'1, 2, 3, 4, 5, 6, 7, 8, 9'
IN	delimiter	‘,’
IN	inSstartingOccurrence	2
OUT	result	44

indent (Custom Function)

This procedure indents text.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**depthNum** – The depth of the incoming resource. Tells this procedure how many times to indent. If null or zero, the original value is returned and not indented.	INTEGER
IN	**indent** – The amount of spaces to indent.	VARCHAR
IN	**inValue** – The value to indent.	LONGVARCHAR
OUT	**outValue** – debug flag	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	depthNum	2 [the depth is 2 – 2nd level]
IN	indent	‘ ‘ [4 spaces]
	inValue	‘Text’ [the word Text]
OUT	outValue	‘ Text’ [8 spaces precede the work Text]

isEmpty (Custom Function)

For a given input string, return 1 if empty or 0 if not.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	request	LONGVARCHAR
OUT	response	BIT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	request	‘’
OUT	response	1

last4ofSSN (Custom Function)

Return ‘x0000’ format for a 9 digit string assumed to be an SSN. Concatenates an ‘x’ in front of the last 4 characters.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inSSN	VARCHAR(255)
OUT	outSSN	VARCHAR(255)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSSN	‘123456789’
OUT	outSSN	‘x6789’

modifyConstant (Custom Function)

This script is used to modify a Constants procedure and change values at the time of deployment in order to enable automation. The Constants file is any SQL Script procedure that contains a format:

```
SET varname = value;
```

This script will search for "SET varname" in the script text in order to modify the value. The value that is passed in is modified between the = and ; which means that any single quotes must be provided if the original value has them. For example a Constants procedure may have many variables set but we are only interested in setting one of the variables which will be uniquely defined within the context of the procedure. Example:

```
SET EnvironmentType = 'DEV';
```

In the above example the objective is to search for the namePair=EnvironmentType and replace the value with valuePair='UAT'. Notice that the value contains all necessary surrounding quotes. The table below shows how to escape values such as single and double quotes if necessary.

The following values that are passed in may be escaped for the values passed in by the variable valuePair:

Description	Value	Escaped Value
quote	"	&quot;
apostrophe	'	&apos;
ampersand	&	&amp;
less than	<	&lt;
greater than	>	&gt;

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	constantsPath	VARCHAR(4096)
IN	namePair	VARCHAR(255)
IN	valuePair	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	constantsPath	'/shared/ASAssets/Utilities/documentation/constants'

Direction	Parameter Name	Parameter Value
IN	namePair	'resourcePath'
IN	valuePair	'"/shared/examples"'

normalizeRowsToPipe

De-dupe an incoming cursor of strings. For example, maybe there is a list of keywords and you only want the unique values. Output the keywords as a pipe cursor. It can be used in conjunction with `string/entityExtractToPipe().`

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	curs	CURSOR(str VARCHAR)
OUT	pipeStr	PIPE (str VARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	curs	{‘Words’, ‘Extracted’, ‘Words’}
OUT	pipeStr	{‘Words’, ‘Extracted’}

normalizeRowsToString

De-dupe an incoming cursor of strings.   For example, maybe there is a list of keywords and you only want the unique values. Output the values in a single comma-delimited string. It can be used in conjunction with `string/entityExtractToPipe()` to produce the keywords cursor.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	curs	CURSOR(str VARCHAR)
OUT	result Comma separate list of words	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	curs	{'Words' 'Extracted', 'Words'}
OUT	result	'Words, Extracted'

numOccurrences

Given two input strings, this function returns an integer value representing the number of occurrences of "searchString" within "stringToSearch".

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	searchString	LONGVARCHAR
IN	stringToSearch	LONGVARCHAR
OUT	num	INTEGER

4. **Examples:**

   4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	searchString	'This is a string'
IN	stringToSearch	'is'
OUT	num	1

p_DelimitedStringToCursor

Converts a delimited VARCHAR into a cursor of VARCHARs. The delimiter is currently limited to a single character. The delimiter character can optionally be included in each token. If any input parameter is NULL, the result is NULL.

5. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inStr	LONGVARCHAR
IN	inDelimiter	VARCHAR
IN	inIncludeDelimiter	BIT
OUT	outTokens	PIPE (strToken LONGVARCHAR)

6. **Examples:**

6.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	‘This is a string’
IN	inDelimiter	‘ ‘
IN	inIncludeDelimiter	0
OUT	outTokens	{‘this’,’is’,’a’,’string’}

p_FixedStringToCursor

Converts a VARCHAR into a cursor of fixed length VARCHARs. If any input parameter is NULL, the result is NULL.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inStr	VARCHAR
IN	inLength	INTEGER
IN	inNormalizeFinalToken	BIT
OUT	outTokens	PIPE (strToken VARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	‘string1string2string3string4’
IN	inDelimiter	7
IN	inIncludeDelimiter	0
OUT	outTokens	{‘string1’,’string2’,’string3’,’string4’}

ParseCSVLine

Converts a line of CSV text to a cursor containing the CSV values.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	csvLine	System.Text (VARCHAR(2147483647))
IN	separator	VARCHAR(1)
IN	qualifier	VARCHAR(1)
OUT	result	PIPE (CSValue VARCHAR(32768))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	csvLine	‘string1, ”string2”, “string “”’quoted”’” 3”, string4’
IN	separator	‘,’
IN	qualifier	‘”’
OUT	result	{string1, string2, string “quoted” 3, string4}

removeDoubleQuotes

Remove the double quotes from a string. Can be useful in circumstances when building a CIS resource path and some of the text contains double quotes. Doubled quotes will be replaced with a single double quote.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inString	LONGVARCHAR
OUT	outString	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inString	‘This string “contains” examples of ””’double”’” quoted text.’
OUT	outString	‘This string contains examples of “double” quoted text.’

removeSingleQuotes

Remove the single quotes from a string. Can be useful in circumstances when building dynamic SQL and some of the text contains single quotes.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inString	LONGVARCHAR
OUT	outString	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inString	"This string 'contains' examples of ''double'' quoted text."
OUT	outString	"This string contains examples of 'double' quoted text."

RegexPatterns

This procedure contains useful and commonly used regular expression patterns that can be used for validating string values (using `string/TextUtils/RegexFind()` or `string/TextUtils/RegexPosition()`) or (for those patterns with matching groups defined) for parsing string values (using `string/TextUtils/RegexGetGroups()`) and pulling out parsed components. See `string/examples/ParseUSPhoneNumber()` for example usage.

1. **Parameters: none**

2. **Examples: none**

3. **Available Patterns:**

Pattern	Purpose	Matching Groups
NUMBER_INTEGER	Validates that a string is a valid integer number	N/A
NUMBER_DECIMAL	Validates that a string is a valid decimal number	N/A
NUMBER_SCIENTIFIC	Validates that a string is a valid number expressed using scientific notation.	N/A
SQL_DATA_TYPE	Validates that a string is a valid CIS SQL data type	N/A
SQL_DATA_TYPE_NAMES	Validates that a string is a valid CIS SQL data type name	N/A
SQL_DATA_TYPE_NAMES_W_SCALE	Validates that a string is a valid CIS SQL data type name that allows the definition of scale	N/A
SQL_DATA_TYPE_NAMES_W_SCALE_AND_PRECISION	Validates that a string is a valid CIS SQL data type name that allows the definition of scale and precision.	N/A
US_PHONE_NUMBER	Validates or parses a U.S./Canada phone number	1 – Country code (NULL if missing)

Pattern	Purpose	Matching Groups
		2 – Area code 3 – Central office code 4 – Subscriber number
XML_DATE_TIME	Validates or parses an XML dateTime string	1 – Date component 2 – Time component 3 – Time zone (NULL if missing)

splitByDelimiter

Split a string by a defined delimiter and return the results in a vector of varchar(255) strings. See also string/TextUtils/RegexSplit.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inString	LONGVARCHAR
IN	inDelimiter any single delimiter character. Values: Common separators include: ' ' or '|' or ','	CHAR(1)
IN	debug - Y/N or T/F	CHAR(1)
OUT	outString	VECTOR(VARCHAR(255))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inString	'This is a delimited string'
IN	inDelimiter	' '
IN	debug	N
OUT	outString	{'This','is','a','delimited','string'}

TextUtils

This section describes the custom java procedure 'TextUtils' which contains several text manipulation utilities.

TextUtils/Blob2Varchar (Custom Function)

Converts a BLOB data type to a VARCHAR. CIS does not natively support this conversion (it *does* natively support CLOB to VARCHAR, however.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	BlobVal	BLOB
OUT	result	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	BlobVal	<blob value>
OUT	result	<blob value as VARCHAR>

TextUtils/CCNumberFormatter (Custom Function)

Provides basic check and standard formatting of a Credit Card number. Validates the length of the supplied numeric field, tries matching it to one of the Visa, MC, AmEx or Discover card patterns, and performs Luhn's validation on the number.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inCCNumber	VARCHAR(2147483647)
OUT	outCCNumber	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inCCNumber	‘5412345678901232’
OUT	outCCNumber	‘5412 3456 7890 1232’

TextUtils/CSVFromCISQuery (Custom Function)

Converts a result set from a CIS query into a CSV string. The inputs “separator_character” and “qualifier_character” should be either a single character or NULL. The input “create_column_headers” indicates whether to include column names as the first row. It should be either “true” or “false”.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	query_string	VARCHAR(2147483647)

Direction	Parameter Name	Parameter Type
IN	separator_character	VARCHAR(2147483647)
IN	qualifier_character	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	'SELECT * FROM /shared/examples/ds_orders/shippingmethods'
IN	separator_character	','
IN	qualifier_character	'"'
IN	create_column_headers	'true'
OUT	result	'ShippingMethodID, ShippingMethod 1, UPS Ground …'

TextUtils/CSVFromCISQueryToFile

Similar to `TextUtils/CSVFromCISQuery`, this CJP converts a result set from a CIS query into a CSV string, however the result is then dumped to a file on the CIS host filesystem. The inputs "separator_character" and "qualifier_character" should be either a single character or NULL. The input "create_column_headers" indicates whether to include column names as the first row. It should be either "true" or "false". The "total_columns" field indicates the expected number of columns in the result and is used as a validation check. The "append" input field indicates whether to append to a file if it already exists (0 = "do not append", 1 = "append".) An integer is returned indicating success (0) or failure (1).

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	query_string	VARCHAR(2147483647)
IN	separator_character	VARCHAR(2147483647)
IN	qualifier_character	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
IN	total_columns	INTEGER
IN	file_path	VARCHAR(2147483647)

Direction	Parameter Name	Parameter Type
IN	append	SMALLINT
OUT	result	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	'SELECT * FROM /shared/examples/ds_orders/shippingmethods'
IN	separator_character	','
IN	qualifier_character	'"'
IN	create_column_headers	'true'
IN	total_columns	2
IN	file_path	'C:\shippingmethods.csv'
IN	append	0
OUT	result	0

TextUtils/FixedFromCISQuery (Custom Function)

Converts a result set from a CIS query into a fixed-width formatted string. The input "format_string" indicates the format of each fixed width row. The format is a pipe separated list of integers indicating the width of each column (col1_Size|col2_Size|...|coln_Size). The input "create_column_headers" indicates whether to include column names as the first row. It should be either "true" or "false". The "total_columns" field indicates the expected number of columns in the result and is used as a validation check.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	query_string	VARCHAR(2147483647)
IN	format_string	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
IN	total_columns	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	'SELECT * FROM /shared/examples/ds_orders/shippingmethods'
IN	format_string	'2|20'
IN	create_column_headers	'true'
IN	total_columns	2
OUT	result	'ShShippingMethod 1 UPS Ground …'

TextUtils/FixedFromCISQueryToFile

Similar to `TextUtils/FixedFromCISQuery`, this CJP converts a result set from a CIS query into a fixed-width formatted string, however the result is then dumped to a file on the CIS host filesystem. The input "format_string" indicates the format of each fixed width row. The format is a pipe separated list of integers indicating the width of each column (col1_Size|col2_Size|...|coln_Size). The input "create_column_headers" indicates whether to include column names as the first row. It should be either "true" or "false". The "total_columns" field indicates the expected number of columns in the result and is used as a validation check. The "append" input field indicates whether to append to a file if it already exists (0 = "do not append", 1 = "append".) An integer is returned indicating success (0) or failure (1).

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	query_string	VARCHAR(2147483647)
IN	format_string	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
IN	total_columns	INTEGER
IN	file_path	VARCHAR(2147483647)
IN	append	SMALLINT
OUT	result	INTEGER

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	'SELECT * FROM /shared/examples/ds_orders/shippingmethods'

Direction	Parameter Name	Parameter Value
IN	format_string	‘2|20’
IN	create_column_headers	‘true’
IN	total_columns	2
IN	file_path	‘C:\shippingmethods.txt’
IN	append	0
OUT	result	0

TextUtils/FormatXML (Custom Function)

Takes an XML string (that has no spaces or newlines) as input and formats the XML, effectively "pretty printing" it.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	UnformattedXML	VARCHAR(2147483647)
OUT	FormatedXML	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	UnformattedXML	'<a><b>b text</b><c>c text</c></a>'

OUT	FormattedXML	'<a>
		    <b>b text</b>
		    <c>c text</c>
		</a>'

TextUtils/GenerateGUID

Generates a random GUID value.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
OUT	result	'42c135c5-945c-40e1-a5a6-67f385e0cea5'

TextUtils/HexToRaw (Custom Function)

This procedure converts a hexadecimal string into a binary array value.

NOTE - Calls to this procedure can be pushed to Oracle data sources by adding some custom code to the Oracle capabilities file. Simply edit the `$CIS_HOME/conf/adapters/system/oracle_<ver>_<type>_driver/oracle_<ver>_<type>_driver_values.xml` by adding the following lines just before the closing "</common:attributes>" line. CIS may need to be restarted after making this change.

```
    <ns726:attribute
xmlns:ns726="http://www.compositesw.com/services/system/util/common">
        <ns726:name>/custom/HexToRaw(@null)</ns726:name>
        <ns726:type>STRING</ns726:type>
        <ns726:value>HexToRaw($1)</ns726:value>
        <ns726:configID>HexToRaw(~string)</ns726:configID>
    </ns726:attribute>
    <ns725:attribute
xmlns:ns725="http://www.compositesw.com/services/system/util/common">
        <ns725:name>/custom/RawToHex(@null)</ns725:name>
        <ns725:type>STRING</ns725:type>
        <ns725:value>RawToHex($1)</ns725:value>
        <ns725:configID>RawToHex(~binary)</ns725:configID>
    </ns725:attribute>
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	hexVal	LONGVARCHAR
OUT	rawVal	LONGVARBINARY

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	hexVal	'1f2e3d4c'
OUT	rawVal	1F2E3D4C

TextUtils/LocalCurrencyFormatter (Custom Function)

Converts a decimal into a localized formatted currency string. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	DOUBLE
IN	inFractionLength	INTEGER
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	12345.6789
IN	inFractionLength	2
IN	ISO639LangCode	‘EN’
IN	ISO3166CountryCode	‘US’
OUT	outValue	‘$12,345.68’

TextUtils/LocalCurrencyParser (Custom Function)

Convert a localized formatted currency string to a decimal. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR

Direction	Parameter Name	Parameter Type
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	DOUBLE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	'$12,345.68'
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'
OUT	outValue	12345.68

TextUtils/LocalDateFormatter (Custom Function)

Convert a date into a localized formatted date string. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	DATE
IN	inStyle 'SHORT', 'MEDIUM', 'LONG', or 'FULL'	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	inValue	2010-08-10
IN	inStyle	‘FULL’
IN	ISO639LangCode	‘EN’
IN	ISO3166CountryCode	‘US’
OUT	outValue	‘Tuesday, August 10, 2010’

TextUtils/LocalDateParser (Custom Function)

Convert a localized formatted date string to a date. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	VARCHAR
IN	inStyle ‘SHORT’, ‘MEDIUM’, ‘LONG’, or ‘FULL’	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	DATE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	‘Tuesday, August 10, 2010’
IN	inStyle	‘FULL’
IN	ISO639LangCode	‘EN’
IN	ISO3166CountryCode	‘US’
OUT	outValue	2010-08-10

TextUtils/LocalNumberFormatter (Custom Function)

Convert a decimal into a localized formatted numeric string. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	DOUBLE
IN	inFractionLength	INTEGER
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	12345.6789
IN	inFractionLength	2
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'
OUT	outValue	'12,345.68'

TextUtils/LocalNumberParser (Custom Function)

Convert a localized formatted numeric string to a decimal. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR

Direction	Parameter Name	Parameter Type
OUT	outValue	DOUBLE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	'12,345.68'
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'
OUT	outValue	12345.68

TextUtils/LocalTimeFormatter (Custom Function)

Convert a date into a localized formatted time string. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	TIME
IN	inStyle 'SHORT', 'MEDIUM', 'LONG', or 'FULL'	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	12:34:56
IN	inStyle	'FULL'
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'

Direction	Parameter Name	Parameter Value
OUT	outValue	‘12:34:56 PM PST’

TextUtils/LocalTimeParser (Custom Function)

Convert a localized formatted time string to a time. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	VARCHAR
IN	inStyle ‘SHORT’, ‘MEDIUM’, ‘LONG’, or ‘FULL’	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	TIME

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	‘12:34:56 PM PST’
IN	inStyle	‘FULL’
IN	ISO639LangCode	‘EN’
IN	ISO3166CountryCode	‘US’
OUT	outValue	12:34:56

TextUtils/LocalTimestampFormatter (Custom Function)

Convert a date into a localized formatted time string. Country code is optional (pass in a NULL.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	TIMESTAMP
IN	inDateStyle	VARCHAR

Direction	Parameter Name	Parameter Type
	'SHORT', 'MEDIUM', 'LONG', or 'FULL'	
IN	inTimeStyle 'SHORT', 'MEDIUM', 'LONG', or 'FULL'	VARCHAR
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	2010-08-10 12:34:56
IN	inDateStyle	'FULL'
IN	inTimeStyle	'FULL'
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'
OUT	outValue	'Tuesday, August 10, 2010 12:34:56 PM PDT'

TextUtils/LocalTimestampParser (Custom Function)

Convert a localized formatted timestamp string to a timestamp. Country code is optional (pass in a NULL.) See also `time/extractTimestamp`.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inValue	VARCHAR
IN	inDateStyle 'SHORT', 'MEDIUM', 'LONG', or 'FULL'	VARCHAR
IN	inTimeStyle 'SHORT', 'MEDIUM', 'LONG', or 'FULL'	VARCHAR

Direction	Parameter Name	Parameter Type
IN	ISO639LangCode (see http://www.loc.gov/standards/iso639-2/php/English_list.php)	VARCHAR
IN	ISO3166CountryCode (see http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm)	VARCHAR
OUT	outValue	TIMESTAMP

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inValue	'Tuesday, August 10, 2010 12:34:56 PM PDT'
IN	inDateStyle	'FULL'
IN	inTimeStyle	'FULL'
IN	ISO639LangCode	'EN'
IN	ISO3166CountryCode	'US'
OUT	outValue	2010-08-10 12:34:56

TextUtils/PhoneNumberFormatter (Custom Function)

Provides standard formatting of phone numbers. If format string is not specified as an input (null or blank), a number will be formatted using the '%.3s-%.3s-%.4s' format (i.e. 999-999-9999.) See http://download.oracle.com/javase/6/docs/api/java/util/Formatter.html for details on formatting syntax.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inPhoneNumber	VARCHAR(2147483647)
IN	inOutputFormat	VARCHAR(2147483647)
OUT	outPhoneNumber	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	inPhoneNumber	'(732) 236-5438'
IN	inOutputFormat	NULL
OUT	outPhoneNumber	'732-236-5438'

TextUtils/RawToHex (Custom Function)

This procedure converts a binary array value into a hexadecimal string.

NOTE - Calls to this procedure can be pushed to Oracle data sources by adding some custom code to the Oracle capabilities file. Simply edit the `$CIS_HOME/conf/adapters/system/oracle_<ver>_<type>_driver/oracle_<ver>_<type>_driver_values.xml` by adding the following lines just before the closing "</common:attributes>" line. CIS may need to be restarted after making this change.

```
    <ns726:attribute
xmlns:ns726="http://www.compositesw.com/services/system/util/common">
        <ns726:name>/custom/HexToRaw(@null)</ns726:name>
        <ns726:type>STRING</ns726:type>
        <ns726:value>HexToRaw($1)</ns726:value>
        <ns726:configID>HexToRaw(~string)</ns726:configID>
    </ns726:attribute>
    <ns725:attribute
xmlns:ns725="http://www.compositesw.com/services/system/util/common">
        <ns725:name>/custom/RawToHex(@null)</ns725:name>
        <ns725:type>STRING</ns725:type>
        <ns725:value>RawToHex($1)</ns725:value>
        <ns725:configID>RawToHex(~binary)</ns725:configID>
    </ns725:attribute>
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	rawVal	LONGVARBINARY
OUT	hexVal	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	rawVal	1F2E3D4C
OUT	hexVal	'1f2e3d4c'

TextUtils/RegexCount (Custom Function)

Count all occurrences of a regular expression match in a VARCHAR and returns the count of the match (similar to the SQL POSITION function, positions start at 1 with 0 indicating a match was not found.) If a NULL value is passed in as the value of any of the inputs, a NULL is returned.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.5 in CIS 4.0.1) See the javadoc for java.util.regex.Pattern for details on what is supported.

See the performance note for RegexFind above for details on how RegEx patterns are cached.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	LONGVARCHAR
IN	Regular Expression	LONGVARCHAR
OUT	result	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	'abaabaaabaaaa'
IN	Regular Expression	'ba+' (matches at least one 'a')
OUT	result	3

TextUtils/RegexFind (Custom Function)

Finds an occurrence of a regular expression match in a VARCHAR and returns the match. The value of the occurrence input value determines which occurrence to return (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.) If no match is found, then a NULL is returned. If a NULL value is passed in as the value of any of the inputs, a NULL is returned. Zero may not be used as a value for an occurrence.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.6 in CIS 5.1.0) See the javadoc for java.util.regex.Pattern for details on what is supported.

Performance note: Instead of compiling a new regular expression pattern every time one of the RegEx CJP's is called, the CJP looks up the pattern in a pattern cache to see if it hasn't already been compiled. This greatly enhances performance when the same pattern is used repeatedly (i.e. as a function call on a result set column.) The cache is capped at 256 patterns (with the least recently used pattern being replaced when a new pattern is compiled) so that CIS's memory management system is not impacted by having a lot of compiled patterns taking up memory.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	VARCHAR
IN	Regular Expression	VARCHAR
IN	Occurrence	INTEGER
OUT	result	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	'abaabaaabaaaa'
IN	Regular Expression	'a+' (matches at least one 'a')
IN	Occurrence	3
OUT	result	'aaa'

TextUtils/RegexGetGroups

Similar to RegexFind, RegexGetGroups finds an occurrence of a regular expression match in a VARCHAR and returns the matched groups (parenthesized groupings) as rows in a cursor. Group 0 is traditionally the entire matched expression and will always be returned if the regular expression matches. The value of the occurrence input value determines which occurrence to return (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.) If no match is found, then an empty result set is returned. If a NULL value is passed in as the value of any of the inputs, an empty result set is returned. Zero may not be used as a value for an occurrence.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.6 in CIS 5.1.0) See the javadoc for java.util.regex.Pattern for details on what is supported.

See the performance note for RegexFind above for details on how RegEx patterns are cached.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	VARCHAR
IN	Regular Expression	VARCHAR
IN	Occurrence	INTEGER
OUT	result	CURSOR(groupNumber INTEGER,

Direction	Parameter Name	Parameter Type
		matchedGroup VARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	‘(650) 227-8200’
IN	Regular Expression	‘\(?(\d{3})\)?[\s\-]*(\d{3})[\s\-]* (\d{4})’ Parses a U.S. phone number in several formats.
IN	Occurrence	1
OUT	result	(0, ‘(650) 227-8200’), (1, ‘650’), (2, ‘227’), (3, ‘8200’)

TextUtils/RegexPosition (Custom Function)

Finds an occurrence of a regular expression match in a VARCHAR and returns the position of the match (similar to the SQL POSITION function, positions start at 1 with 0 indicating a match was not found.) The value of the occurrence input value determines which occurrence to return (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.) If a NULL value is passed in as the value of any of the inputs, a NULL is returned. Zero may not be used as a value for an occurrence.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.6 in CIS 5.1.0) See the javadoc for java.util.regex.Pattern for details on what is supported.

See the performance note for RegexFind above for details on how RegEx patterns are cached.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	VARCHAR
IN	Regular Expression	VARCHAR
IN	Occurrence	INTEGER
OUT	result	INTEGER

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	'abaabaaabaaaa'
IN	Regular Expression	'a+' (matches at least one 'a')
IN	Occurrence	3
OUT	result	6

TextUtils/RegexReplace (Custom Function)

Finds an occurrence of a regular expression match in a VARCHAR and replaces the match with the replacement text input value. The value of the occurrence input value determines which occurrence to replace (numbered starting at 1 from left to right. Use negative values to number occurrences from right to left.) Zero may be used as a value for an occurrence and indicates that ALL matches should be replaced. If no match is found, then the original input text is returned. If a NULL value is passed in as the value of any of the inputs, the original input text is returned.

The regular expression language used is what is supported by the JDK used by CIS (currently 1.6 in CIS 5.1.0) See the javadoc for java.util.regex.Pattern for details on what is supported. Also see the javadoc for java.util.regex.Matcher (specifically for the appendReplacement() method) for detail on how to include grouped (as distinguished from "matched") text in the replacement text.

See the performance note for RegexFind above for details on how RegEx patterns are cached.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	VARCHAR
IN	Regular Expression	VARCHAR
IN	Replacement Text	VARCHAR
IN	Occurrence	INTEGER
OUT	result	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	'abaabaaabaaaa'
IN	Regular Expression	'a+' (matches at least one 'a')
IN	Replacement Text	'i$0' (puts an 'i' in front of the matched text.)
IN	Occurrence	0 (all occurrences)

Direction	Parameter Name	Parameter Value
OUT	result	‘iabiaabiaaabiaaaa’

TextUtils/RegexSplit

This function uses Java's String.split() method to split a string using a regular expression and a limit.

Paraphrased from String's javadoc:

The cursor returned by this function contains each substring of this string that is terminated by another substring that matches the given expression or is terminated by the end of the string. The substrings in the cursor are in the order in which they occur in this string. If the expression does not match any part of the input then the resulting cursor has just one row, namely this string.

The limit parameter controls the number of times the pattern is applied and therefore affects the length of the resulting cursor. If the limit n is greater than zero then the pattern will be applied at most n - 1 times, the cursor's cardinality will be no greater than n, and the cursor's last row will contain all input beyond the last matched delimiter. If n is non-positive then the pattern will be applied as many times as possible and the cursor can have any number of rows. If n is zero then the pattern will be applied as many times as possible, the cursor can have any number of rows, and trailing empty strings will be discarded.

The string 'boo:and:foo', for example, yields the following results with these parameters:

Regex	Limit	Result
:	2	'boo', 'and:foo'
:	5	'boo', 'and', 'foo'
:	-2	'boo', 'and', 'foo'
o	5	'b', '', ':and:f', '', ''
o	-2	'b', '', ':and:f', '', ''
o	0	'b', '', ':and:f'

The regular expression language used is what is supported by the JDK used by CIS (currently 1.6 in CIS 5.1.0) See the javadoc for java.util.regex.Pattern for details on what is supported.

See also `string/splitByDelimiter()`.

See the performance note for RegexFind above for details on how RegEx patterns are cached.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Input Text	VARCHAR
IN	Regular Expression	VARCHAR
IN	Limit	INTEGER
OUT	result	CURSOR(splitElement VARCHAR)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	Input Text	'abaacaaabaaaa'
IN	Regular Expression	'[bc]' (delimit using the characters 'b' or 'c')
IN	Limit	0
OUT	result	('a'), ('aa'), ('aaa') , ('aaaa')

TextUtils/SSNumberFormatter (Custom Function)

Provides standard formatting of a Social Security number.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inSSNumber	VARCHAR(2147483647)
OUT	outSSNumber	VARCHAR(2147483647)

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inSSNumber	'111223456'
OUT	outSSNumber	'111-22-3456'

19 How To Use ‘Templates’ Procedures

Introduction

This section will show how to use the Templates procedures.

procedureTemplate

This entire procedure is a template that shows the best practice for documenting a CIS Procedure. This is a procedure description that describes the purpose of this procedure.

1. **Parameters: none**

2. **Examples: none**

20 How To Use 'Time' Procedures

Introduction

This section will show how to use the 'Time' manipulation procedures.

ADD_MONTHS (Custom Function)

This procedure takes an input timestamp and adds (or subtracts) an input number of months to it. If any of the inputs are null a null will be returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	startTimestamp	TIMESTAMP
IN	numMonths	INTEGER
OUT	result	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	startTimestamp	'2011-03-01 00:00:00'
IN	numMonths	-2
OUT	result	'2011-01-01 00:00:00'

DefaultValues

This procedure contains a vector of valid date, time and timestamp formats which is used by `time/extractDate()`, `time/extractTime()`, and `time/extractTimestamp()` to extract a date, time or timestamp from a string. Add additional formats to the vector as needed. Place the most common formats towards the top of the vector so that the extraction procedure works as efficiently as it can.

1. **Parameters: none**

2. **Examples: none**

extractDate (Custom Function)

This procedure takes an input string that is expected to contain a date value. It compares this string against a (non-exhaustive) set of date formats, and extracts the date value. The time formats supported in this procedure are defined as constants in `time/DefaultValues()`.

Exceptions: InvalidDateException

If isMandatory=0 and no format could be found then return null.

If isMandatory=1 and no format could be found then an exception is thrown.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inDateStr	VARCHAR(255)
IN	isMandatory	SMALLINT
IN	debug Y/N or T/F	CHAR(1)
OUT	extractDate	DATE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inDateStr	Aug 7, 2010
IN	isMandatory 0 means not mandatory, 1 means mandatory.	0
IN	debug	Y
OUT	extractDate If isMandatory=0 and no format could be found then return null. If isMandatory=1 and no format could be found then an exception is thrown.	2010-07-27

extractTime (Custom Function)

This procedure takes an input string that is expected to contain a time value. It compares this string against a (non-exhaustive) set of time formats, and extracts the time value. The time formats supported in this procedure are defined as constants in `time/DefaultValues()`.

Exceptions: InvalidTimeException

If isMandatory=0 and no format could be found then return null.

If isMandatory=1 and no format could be found then an exception is thrown.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inTimeStr	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	isMandatory	SMALLINT
IN	debug Y/N or T/F	CHAR(1)
OUT	extractTime	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inTimeStr	'12:00:00'
IN	isMandatory 0 means not mandatory, 1 means mandatory.	0
IN	debug	Y
OUT	extractTimeIf isMandatory=0 and no format could be found then return null. If isMandatory=1 and no format could be found then an exception is thrown.	12:00:00

extractTimestamp (Custom Function)

Extract a timestamp from a string using a vector of default timestamp formats found in `time/DefaultValues()`. Throws an exception (InvalidTimestampException) if no valid timestamp could be extracted. See also `string/TextUtils/LocalTimestampParser()`.

Exceptions: InvalidTimestampException

If isMandatory=0 and no format could be found then return null.

If isMandatory=1 and no format could be found then an exception is thrown.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inTimestampStr	VARCHAR(255)
IN	isMandatory	SMALLINT
IN	debug Y/N or T/F	CHAR(1)
OUT	extractTimestamp	TIMESTAMP

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inTimestampStr	'Aug 7, 2010 12:00:00.101'
IN	isMandatory 0 means not mandatory, 1 means mandatory.	0
IN	debug	Y
OUT	extractTimestamp If isMandatory=0 and no format could be found then return null. If isMandatory=1 and no format could be found then an exception is thrown.	2010-08-07 12:00:00.101

getCurrentTimestamp (Custom Function)

Get the current timestamp. Added this procedure getCurrentTimestamp to workaround a bug introduced by 7.0.3 server when setting CURRENT_TIMESTAMP within the context of a procedure. Since the invocation of this procedure put the CURRENT_TIMESTAMP in a separate context from the invoking procedure, it would return the exact current timestamp.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	currentTS	TIMESTAMP

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	currentTS	2018-02-01 12:00:00

getTimestampInterval (Custom Function)

Add a value of "hours" to a given timestamp value to calculate time.

3. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inInterval measured in hours	INTEGER
IN	inTimestamp	TIMESTAMP

Direction	Parameter Name	Parameter Type
OUT	endDatetime	TIMESTAMP

4. **Examples:**

4.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inInterval	25
IN	inTimestamp	2010-07-10 00:00:00
OUT	endDatetime	2010-07-11 01:00:00

intervalDay2Seconds (Custom Function)

Converts an INTERVAL DAY data type to an equivalent number of seconds. If either input is NULL a NULL will be returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inIntervalDay – An INTERVAL DAY value.	INTERVAL DAY TO SECOND
OUT	result – The number of seconds (including any fractional component) in the input interval.	DOUBLE

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inIntervalDay	0 00:05:00
OUT	result	300.0

period2IntervalDay (Custom Function)

Converts a specified period to an INTERVAL DAY data type. If either input is NULL a NULL will be returned.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	amount - The number of units (specified by periodName)	INTEGER
IN	periodName - The unit of measure for the "amount" input: second, minute, hour, day, week, month (31 days), year	VARCHAR(20)

Direction	Parameter Name	Parameter Type
OUT	result - An interval of the specified period	INTERVAL DAY(9) TO SECOND

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	amount	1
IN	periodName	‘week’
OUT	result	7 00:00:00

DateUtils

This section describes the use of the custom java procedure (DateUtils) which are used for various date manipulations.

BigintToTimestamp (Custom Function)

This procedure converts a long integer (in milliseconds since The Epoch, otherwise known as midnight on January 1, 1970 GMT) to a TIMESTAMP. Negative long integer values are allowed.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inBigint	BIGINT
OUT	result	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inBigint	0
OUT	Result	'1969-12-31 16:00:00' (if server is in PST timezone.)

DateUtils/DateAddDate (Custom Function)

Returns a new Date value based on adding a datePart to the specified Date. It is leap year aware. Valid values for datePart are ‘second’, ‘minute’, ‘hour’, ‘day’, ‘week’, ‘month’ and ‘year’ (these not case sensitive.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datePart	VARCHAR
IN	dateLength	INTEGER
IN	startDate	DATE
OUT	endDate	DATE

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	datePart	‘day’
IN	dateLength	29
IN	startDate	‘2008-02-01’
OUT	endDate	‘2008-03-01’

DateUtils/DateAddTimestamp (Custom Function)

Returns a new Timestamp value based on adding a datePart to the specified Timestamp. It is leap year aware. Valid values for datePart are ‘second’, ‘minute’, ‘hour’, ‘day’, ‘week’, ‘month’ and ‘year’ (these not case sensitive.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datePart	VARCHAR
IN	dateLength	INTEGER
IN	startDate	TIMESTAMP
OUT	endDate	TIMESTAMP

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	datePart	‘second’
IN	dateLength	2505600
IN	startDate	‘2008-02-01 00:00:00’
OUT	endDate	‘2008-03-01 00:00:00’

DateUtils/DateDiffDate (Custom Function)

Returns the difference between two dates in the specified unit of measure. It is leap year aware. Valid values for datePart are ‘second’, ‘minute’, ‘hour’, ‘day’, ‘week’, ‘month’ and ‘year’ (these not case sensitive.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datePart	VARCHAR
IN	startDate	DATE
IN	endDate	DATE
OUT	dateLength	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	datePart	‘day’
IN	startDate	‘2008-02-01’
IN	endDate	‘2008-03-01’
OUT	dateLength	29

DateUtils/DateDiffTimestamp (Custom Function)

Returns the difference between two timestamps in the specified unit of measure. It is leap year aware. Valid values for datePart are ‘second’, ‘minute’, ‘hour’, ‘day’, ‘week’, ‘month’ and ‘year’ (these not case sensitive.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	datePart	VARCHAR
IN	startTimestamp	TIMESTAMP
IN	endTimestamp	TIMESTAMP
OUT	dateLength	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	datePart	‘second’
IN	startTimestamp	‘2008-02-01 00:00:00’
IN	endTimestamp	‘2008-03-01 00:00:00’
OUT	dateLength	2505600

DateUtils/GetServerTimezone (Custom Function)

Returns the time zone this instance of CIS is running in. Various display types are available:

Display Type	Description
‘ID’	The time zone ID (according to Java, i.e. "America/Los_Angeles")
‘SHORT_NAME’	The name of the time zone in short format (i.e. "PDT")
‘LONG_NAME’	The name of the time zone in long format (i.e. "Pacific Daylight Time")
‘OFFSET’	The number of milliseconds to add to GMT time to get the current time zone's time (i.e. "-28800000")
'XML'	The offset in hours and minutes from GMT (i.e. "-08:00") that is expected for an XML dateTime or time field.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	displayType	VARCHAR
OUT	outValue	VARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	displayType	‘SHORT_NAME’
OUT	extractTimestamp	‘PDT’

TimestampToBigint (Custom Function)

This procedure converts a TIMESTAMP to a long integer (in milliseconds since The Epoch, otherwise known as midnight on January 1, 1970 GMT.) Timestamp values from before The Epoch are allowed.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inTimestamp	TIMESTAMP
OUT	result	BIGINT

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inBigint	'1969-12-31 16:00:00' (if server is in PST timezone.)
OUT	Result	0

DateUtils/TZConverter (Custom Function)

Converts a timestamp from one time zone to another. Valid time zone input values can be found in the "Info" tab of this CJP.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	sourceTimestamp	TIMESTAMP
IN	fromTimeZone	VARCHAR
IN	toTimeZone	VARCHAR
OUT	targetTimestamp	TIMESTAMP

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	sourceTimestamp	‘2010-01-01 00:00:00’
IN	fromTimeZone	‘UTC’
IN	toTimeZone	‘PST8PDT’
OUT	targetTimeZone	‘2009-12-31 16:00:00’

21 How To Use ‘Upgrade’ Procedures

Introduction

This section describes the routines using the “upgrade” procedures. These procedures and views are designed to assist with major upgrades of CIS.

getDatabaseTests

This view returns default regression testing entries that can be used with JMeter regression suite for upgrade testing. It returns queries for all published views for all virtual databases, except the 'system' or 'examples' databases.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	N/A	CURSOR ("Skip Execution" CHAR(2), "Test Name" VARCHAR(21), "Test Query" LONGVARCHAR, "Test Plan" VARCHAR(260), "Service Name" VARCHAR(255))

getServiceTests

This view returns default regression testing entries that can be used with JMeter regression suite for upgrade testing. It returns queries for all published operations for all virtual web services, except the 'admin' or 'util' services.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	N/A	CURSOR ("Skip Execution" CHAR(2), "Test Name" VARCHAR(21), "Operation Path" LONGVARCHAR, "Test Plan" VARCHAR(260), "Service Name" VARCHAR(255))

updateCacheConfigTables

This procedure will update the cache config tables - cache_status and cache_tracking - to allow a new server to use existing cache data as part of a CIS server version upgrade. This can be done as part of a migration from CIS 6.2 to CIS 7.0, and allow cache data to be preserved after the upgrade.

NOTE: BEFORE RUNNING THIS PROCEDURE, ENSURE THAT CACHING IS DISABLED ON THE OLD SERVER AND NEW SERVER.

NOTE: THIS PROCEDURE IS TO BE RUN ON THE TARGET SITE FOR UPGRADE - I.E. IF YOU ARE UPGRADING FROM CIS 6.2 TO 7.0, EXECUTE THE PROCEDURE ON THE CIS 7.0 SERVER, PROVIDING THE CIS 6.2 SERVER'S SERVER_ID AS INPUT.

NOTE: AFTER RUNNING THIS PROCEDURE, RE-ENABLE CACHING ON THE TARGET SITE FOR UPGRADE ONLY.

NOTE: ALSO RECOMMEND SETTING CACHE STATUS SYNC INTERVAL TO 60 SECONDS ON SOURCE AND TARGET SERVER BEFORE RUNNING.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**performDelete** – Indicates whether to delete existing entries for this server.	BIT
IN	**performInsert** – Indicates whether to insert new entries for this server.	BIT
IN	**previousServerID** – Server ID of the previous CIS server that is being upgraded	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	performDelete	1
IN	performInsert	1
IN	previousServerID	'cgoodric-vm-win7x64-9400-945983814'

helpers

This section describes the auxiliary procedures for documentation.

helpers/configuredCaches

This view returns all distinct caching target data sources

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	N/A	CURSOR (DATASOURCE_PATH VARCHAR(4096), STATUS_PATH VARCHAR(4096),

Direction	Parameter Name	Parameter Type
		TRACKING_PATH VARCHAR(4096))

helpers/findCaches

This procedure finds cached resources configured in CIS managed cache_status tables.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
OUT	result	CURSOR (RESOURCE_PATH VARCHAR(4096), DATASOURCE_PATH VARCHAR(4096), STATUS_PATH VARCHAR(4096), TRACKING_PATH VARCHAR(4096))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
OUT	result	('/shared/examples/ds_orders/tutorial/orders', '/shared/examples/ds_orders', '/shared/examples/ds_orders/tutorial/cache_status', '/shared/examples/ds_orders/tutorial/cache_tracking'), …

helpers/returnColumnOrderingString

This procedure generates an ORDER BY string with the numCols numbered columns. It is useful for generating an ORDER BY clause for use with automated regression testing.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	**numCols** – The number of columns to include in the generated ORDER BY string.	INTEGER
OUT	**orderByString** – The ORDER BY string result	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

Direction	Parameter Name	Parameter Value
IN	numCols	10
OUT	orderByString	'ORDER BY 1, 2, 3, 4, 5, 6, 7, 8, 9, 10'

22 How To Use ‘XML’ Procedures

Introduction

This section will show how to use the ‘XML’ manipulation procedures.

castXMLTextNodeAsVarchar (Custom Function)

This script converts XML text node to varchar. Does appropriate conversion from XML schema dateTime or time to ANSI format (and even adjusts timezone to server's timezone if timezone provided.)

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	xmlValue	XML
OUT	result	/lib/util/System.Text (VARCHAR(2147483647))

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	XML(‘this is a test’)
OUT	outStr	’this is a test’

2.2. **Assumptions: CIS instance in PDT timezone.**

Direction	Parameter Name	Parameter Value
IN	inStr	XML(‘2010-12-15T12:50:00-08:00’)
OUT	outStr	’2010-12-15 12:50:00’

CreateXmlString2CursorXForm

This script can be used to create XSLT transforms that take a VARCHAR (with XML content) as input instead of an XML data source. The iXSLT input variable takes an XSLT document that produces the same XML as the standard XSLT transform does. In fact, the easiest way to create the XSLT is probably to use an XSLT modeler to create a transformation against a sample XML flat file then copy and paste the XSLT code.

The iColumnDefList input variable takes a vector of column definition rows (see the ColumnDefRow and ColumnDefList type definitions below.) This describes the transformation's output cursor. See `xml/examples/createProcedureFromProductCatalogXForm()` for an example of how this script is used.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	iXFormProcName	/lib/resource/ResourceDefs.ResourceName
IN	iParentFolder	/lib/resource/ResourceDefs.ResourcePath
IN	iXSLT	XML
IN	iColumnDefList	ColumnDefList

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	iXFormProcName	‘ProductCatalogTransform’
IN	iParentFolder	’/shared/ASAssets/Utilities/"xml"/examples’
IN	iXSLT	XSLT script not shown here. See examples folder.
IN	iColumnDefList	VECTOR[('ProductID', 'INTEGER'), ('ProductName', 'VARCHAR(32768)'), ('CategoryID', 'INTEGER'), ('CategoryName', 'VARCHAR(32768)'), ('ProductDescription', 'VARCHAR(32768)'), ('SerialNumber', 'VARCHAR(32768)'), ('UnitPrice', 'DECIMAL(32,2)'), ('ReorderLevel', 'INTEGER'), ('LastReorder', 'TIMESTAMP'), ('LeadTime', 'VARCHAR(32768)')]

escapeXML (Custom Function)

Change xml tags to their corresponding escape sequences.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inStr	LONGVARCHAR

Direction	Parameter Name	Parameter Type
OUT	outStr	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	'<tag>somevalue</tag>'
OUT	outStr	'&lt;tag&gt;somevalue&lt;/tag&gt;'

getNodeFromXML

Given a qualified XPath, extract the node tree from the incoming XML.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug - Y or T = debugging turned on, N or F = debugging turned off	CHAR(1)
IN	namespaces - any string with the valid namespaces for the incoming XML	LONGVARCHAR
IN	inXpath - xpath statement used to extract the value at that location in the XML	LONGVARCHAR
IN	iteration - 0=get all iterations, 1=get 1st occurrence only	INTEGER
IN	inXml - any valid XML document	XML
OUT	outValue - text value from the XML document as directed by the XPath. Empty String returned if node not found.	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	namespaces	'xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"'
IN	inXpath	'/Book/sam:Name/@sam:isbn'
IN	iteration	1

Direction	Parameter Name	Parameter Value
IN	inXml	'<Book xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0"> <sam:Name sam:isbn="12-3456-123">Test</sam:Name> <Chapter>Test Data</Chapter> </Book>'
OUT	outValue	12-3456-123

getValueFromXML (Custom Function)

Given an XPath, extract the value from any XML document.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	debug	CHAR(1)
IN	namespaces	LONGVARCHAR
IN	inXpath	LONGVARCHAR
IN	inXml	XML
OUT	outValue	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	namespaces	'xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"'
IN	inXpath	'/Book/sam:Name/@sam:isbn'
IN	inXml	'<Book xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0"> <sam:Name sam:isbn="12-3456-123">Test</sam:Name> <Chapter>Test Data</Chapter> </Book>'
OUT	outValue	'12-3456-123'

parseAndModifyXML

Parse and XML document given a vector of XPath statements and modify the values and reconstruct the XML. IMPORTANT: When updating an element with a list of attributes, you must supply all of the original attributes in order to retain what you had. This routine overwrites the original element so be careful.

Use cases supported:

- Use Case 1: Modify the attribute for an element
- Use Case 2: Modify the value of the the element with attributes
- Use Case 3: Modify the element with no attributes
- Use Case 4: Modify the element with no attributes and set it to null
- Use Case 5: Retain the original attributes and modify the element with attributes and set it to null

Limitations: This method does not modify an iteration of the same node.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Debug - Y/N or T/F	CHAR(1)
IN	namespaces	LONGVARCHAR
IN	ElemAttrVector	VECTOR (/shared/ASAssets/Utilities/TypeDefinitions.ElemAttrValuesType)
IN	inXml	XML
OUT	outXml	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	‘N’
IN	namespaces	'xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"'
IN	ElemAttrVector	{

Direction	Parameter Name	Parameter Value
		-- Use Case 1: Modify the attribute for an element ('sam:Name','sam:isbn','00-0000-000'), -- Use Case 2: Modify the value of the the element with attributes ('sam:Name',NULL,'Joe Author'), -- Use Case 3: Modify the element with no attributes ('Chapter',NULL,'Chapter 1'), -- Use Case 4: Modify the element with no attributes and set it to null ('Paragraph',NULL,NULL), -- Use Case 5: Retain the attribute ('sam:Author','sam:firstname','Joe'), -- Use Case 5: Retain the attribute ('sam:Author','sam:lastname','Author'), -- Use Case 5: Modify the element with attributes and set it to null ('sam:Author',NULL,NULL) }
IN	inXml	'<Book xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0"> <sam:Name sam:isbn="12-3456-123"/> <Chapter>Test Data</Chapter> <Paragraph>Paragraph 1</Paragraph> </Book> '
OUT	outXml	'<Book xmlns:sam="http://www.compositesw.com/samples/mynamespace/v1.0"> <sam:Name sam:isbn="00-0000-000">Joe Author</sam:Name> <Chapter>Chapter 1</Chapter> <Paragraph xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/> </Book>'

pruneXML

Prune out empty XML nodes except those in an retainXPathVector list.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	Debug - Y/N or T/F	CHAR(1)
IN	namespaces - any string with the valid	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	namespaces for the incoming XML	
IN	retainXPathVector - contains a vector list of XPath statements for the incoming XML that insure those nodes are retained in the XML if they are empty. The invoking of this procedure must define and populate the vector. The invoker will determine the XPath statement by consulting the XML Schema for required elements and the deterimine from their application which nodes have the possibility of being empty. Those are the candidates for this list. Values: e.g. {'/rootnode/XYZ/A/@foo', '/rootnode/XYZ/C'}	VECTOR(varchar(2048))
IN	inXml – any valid XML document	XML
OUT	outXml – any valid XML document	XML

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	debug	'N'
IN	namespaces	'xmlns:tns="http://mynamespace" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"'
IN	retainXPathVector	{'/rootnode/XYZ/A/@foo','/rootnode/XYZ/C'}
IN	inXml	'<rootnode xmlns:tns="http://mynamespace"> <XYZ abc="def" ghi=""> <A foo="">aa</A> <B>bbb</B> <C/> </XYZ> </rootnode>'
OUT	outXml	'<rootnode xmlns:tns="http://mynamespace"> <XYZ abc="def"> <A foo="">aa</A> <B>bbb</B> <C/> </XYZ> </rootnode>'

reverseXML (Custom Function)

This provides a way to reverse the direction of an XML document so that CIS thinks it is calling a data source. This is the only way that you can change an XSLT data source into a transformation. CIS does not allow you to simply build a pure XML transformation. It always thinks you are transforming an XML file.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inXml	XML
OUT	outXml	XML

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inXml	<tag>somevalue</tag>
OUT	outXml	<tag>somevalue</tag>

stripInvalidXMLChars (Custom Function)

Strip the invalid characters from an XML document. This procedure is still under construction

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inXml	XML
OUT	outXml	XML

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inXml	
OUT	outXml	

unescapeXML (Custom Function)

Change xml entities to their corresponding individual characters.

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	inStr	LONGVARCHAR
OUT	outStr	LONGVARCHAR

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inStr	'&quot;A &amp; B &lt; C &amp; D&quot;'
OUT	outStr	'"A & B < C & D"'

XMLUtils

This section describes the use of the custom java procedure (DateUtils) which are used for various date manipulations.

XMLUtils/CSVFromXMLToFile

Similar to `TextUtils/CSVFromCISQueryToFile`, this CJP converts a result set in XML format into a CSV string and dumps the result to a file on the CIS host filesystem. The inputs "separator_character" and "qualifier_character" should be either a single character or NULL. The input "create_column_headers" indicates whether to include column names as the first row. It should be either "true" or "false". The "total_columns" field indicates the expected number of columns in the result and is used as a validation check. The "append" input field indicates whether to append to a file if it already exists (0 = "do not append", 1 = "append".) An integer is returned indicating success (0) or failure (1).

Note: For best results, the XML string should be formatted with repeated rows containing all expected columns in each row. Deviation from this pattern may result in unexpected behavior. The specific XML node names and number of columns do not matter as long as it follows the example pattern shown below:

```
<?xml version="1.0"?>
<p1:Customer xmlns:p1="http://www.compositesw.com/ps/FileProcessor">
     <row>
          <customerID>1</customerID>
          <companyName>Composite Software</companyName>
          <contactFirstName>John</contactFirstName>
          <contactLastName>Doe</contactLastName>
          <billingAddress>1234 First Avenue NE</billingAddress>
          <city>Reston</city>
          <stateOrProvince>VA</stateOrProvince>
          <postalCode>22190</postalCode>
```

```
                <countryRegion>USA</countryRegion>
                <contactTitle>Mr</contactTitle>
                <phoneNumber>(703)111-2222</phoneNumber>
                <faxNumber>(703)111-3333</faxNumber>
        </row>
        <row>
                <customerID>2</customerID>
                <companyName>Company 2</companyName>
                <contactFirstName>Jane</contactFirstName>
                <contactLastName>Doe</contactLastName>
                <billingAddress>5678 Second Street NW</billingAddress>
                <city>Washington</city>
                <stateOrProvince>DC</stateOrProvince>
                <postalCode>10002</postalCode>
                <countryRegion>US</countryRegion>
                <contactTitle>Mrs</contactTitle>
                <phoneNumber>202-111-2222</phoneNumber>
                <faxNumber>202-111-3333</faxNumber>
        </row>
</p1:Customer>
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	xml_string	VARCHAR(2147483647)
IN	separator_character	VARCHAR(2147483647)
IN	qualifier_character	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
IN	total_columns	INTEGER
IN	file_path	VARCHAR(2147483647)
IN	append	SMALLINT
OUT	result	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	See example XML above
IN	separator_character	‘,’
IN	qualifier_character	‘”’
IN	create_column_headers	‘true’

Direction	Parameter Name	Parameter Value
IN	total_columns	2
IN	file_path	'C:\customer.csv'
IN	append	0
OUT	result	0

XMLUtils/DeleteElement (Custom Function)

Removes an element (including it's children) from an XML structure.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inXML	VARCHAR(2147483647)
IN	inElementName	VARCHAR(2147483647)
IN	inElementNamespace	VARCHAR(2147483647)
IN	occurrence	INTEGER
OUT	result	VARCHAR(2147483647)

2. **Examples:**

   2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

IN	inXML	'<xmldoc xmlns="uri:mynamespace">
		    <parent>
		        <child>
		            <grandchild/>
		        </child>
		    </parent>
		</xmldoc>

IN	inElementName	'child'
IN	inElementNamespace	'uri:mynamespace'
IN	occurrence	1
OUT	result	'<xmldoc xmlns="uri:mynamespace"> <parent/> </xmldoc>'

XMLUtils/DeleteElementSpareChildren (Custom Function)

Removes an element from an XML structure. The element's children become children of the element's parent.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inXML	VARCHAR(2147483647)
IN	inElementName	VARCHAR(2147483647)
IN	inElementNamespace	VARCHAR(2147483647)
IN	occurrence	INTEGER
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value

IN	inXML	'<xmldoc xmlns="uri:mynamespace">
		    <parent>
		        <child>
		            <grandchild/>
		        </child>
		    </parent>
		</xmldoc>'

IN	inElementName	'child'
IN	inElementNamespace	'uri:mynamespace'
IN	occurrence	1

OUT	result	'<xmldoc xmlns="uri:mynamespace">
		    <parent>
		        <grandchild/>
		    </parent>
		</xmldoc>'

XMLUtils/FixedFromXMLToFile

Similar to `TextUtils/FixedFromCISQueryToFile`, this CJP converts a result set from a result set in XML format into a fixed-width formatted string then the result is dumped to a file on the CIS host filesystem. The input "format_string" indicates the format of each fixed width row.

The format is a pipe separated list of integers indicating the width of each column (col1_Size|col2_Size|...|coln_Size). The input "create_column_headers" indicates whether to include column names as the first row. It should be either "true" or "false". The "total_columns" field indicates the expected number of columns in the result and is used as a validation check. The "append" input field indicates whether to append to a file if it already exists (0 = "do not append", 1 = "append".) An integer is returned indicating success (0) or failure (1).

Note: For best results, the XML string should be formatted with repeated rows containing all expected columns in each row. Deviation from this pattern may result in unexpected behavior. The specific XML node names and number of columns do not matter as long as it follows the example pattern shown below:

```
<?xml version="1.0"?>
<p1:Customer xmlns:p1="http://www.compositesw.com/ps/FileProcessor">
     <row>
          <customerID>1</customerID>
          <companyName>Composite Software</companyName>
          <contactFirstName>John</contactFirstName>
          <contactLastName>Doe</contactLastName>
          <billingAddress>1234 First Avenue NE</billingAddress>
          <city>Reston</city>
          <stateOrProvince>VA</stateOrProvince>
          <postalCode>22190</postalCode>
          <countryRegion>USA</countryRegion>
          <contactTitle>Mr</contactTitle>
          <phoneNumber>(703)111-2222</phoneNumber>
          <faxNumber>(703)111-3333</faxNumber>
     </row>
     <row>
          <customerID>2</customerID>
          <companyName>Company 2</companyName>
          <contactFirstName>Jane</contactFirstName>
          <contactLastName>Doe</contactLastName>
          <billingAddress>5678 Second Street NW</billingAddress>
          <city>Washington</city>
          <stateOrProvince>DC</stateOrProvince>
          <postalCode>10002</postalCode>
          <countryRegion>US</countryRegion>
          <contactTitle>Mrs</contactTitle>
          <phoneNumber>202-111-2222</phoneNumber>
          <faxNumber>202-111-3333</faxNumber>
     </row>
</p1:Customer>
```

1. **Parameters:**

Direction	Parameter Name	Parameter Type

Direction	Parameter Name	Parameter Type
IN	xml_string	VARCHAR(2147483647)
IN	format_string	VARCHAR(2147483647)
IN	create_column_headers	VARCHAR(2147483647)
IN	total_columns	INTEGER
IN	file_path	VARCHAR(2147483647)
IN	append	SMALLINT
OUT	result	INTEGER

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	query_string	See XML example above
IN	format_string	‘5|40|20|40|40|40|5|10|2|10|20|20’
IN	create_column_headers	‘true’
IN	total_columns	2
IN	file_path	‘C:\customer.txt’
IN	append	0
OUT	result	0

XMLUtils/HTMLtoXML

This procedure converts HTML into XHTML (XML). HTML has a looser tagging syntax than XML so this procedure uses the JTidy library (http://jtidy.sourceforge.net) to clean up the HTML to conform to the XHTML standard.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	inHTML	LONGVARCHAR
OUT	outXML	LONGVARCHAR

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inHTML	‘<html><body><table><tr><td></table></body></html>’

Direction	Parameter Name	Parameter Value
OUT	outXML	‘<html> <head> <META http-equiv="Content-Type" content="text/html; charset=UTF-8"> <meta name="generator" content="HTML Tidy for Java (vers. 2009-12-01), see jtidy.sourceforge.net"> <title></title> </head> <body> <table> <tr> <td></td> </tr> </table> </body> </html>’

XMLUtils/InsertElementDemoteChildren (Custom Function)

Inserts an element from into an XML structure. The parent element’s children become children of the new element.

1. **Parameters:**

Direction	Parameter Name	Parameter Type
IN	XML	VARCHAR(2147483647)
IN	parentElementName	VARCHAR(2147483647)
IN	parentElementNamespace	VARCHAR(2147483647)
IN	occurrence	INTEGER
IN	ElementName	VARCHAR(2147483647)
IN	ElementNamespace	VARCHAR(2147483647)
OUT	result	VARCHAR(2147483647)

2. **Examples:**

2.1. **Assumptions: none**

Direction	Parameter Name	Parameter Value
IN	inXML	‘<xmldoc xmlns=”uri:mynamespace”> <parent> <grandchild/> </parent>

Direction	Parameter Name	Parameter Value
		</xmldoc>’
IN	parentElementName	‘parent’
IN	parentElementNamespace	‘uri:mynamespace’
IN	occurrence	1
IN	ElementName	‘child’
IN	ElementNamespace	‘uri:mynamespace’

OUT	result	‘<xmldoc xmlns=”uri:mynamespace”>
		     <parent>
		         <child>
		             <grandchild/>
		         </child>
		     </parent>
		 </xmldoc>

23 How To Submit New Procedures

Introduction

This section will provide guidelines for submitting new procedures to the consolidated CIS development utilities library.

Documentation

More than anything else, the procedures in this library need to be documented. SQL Scripts need to have documentation similar to the following in both a header comment and also in the "Annotations" field in the "Info" tab. With CJP's, the code is not visible to the CIS developer, but the source code should also have comments with the same information and also have documentation in each CJP's "Annotation" field. (Having documentation in the "Annotation" field keeps things consistent regardless of whether the procedure is an SQL Script or a CJP.)

Recommended documentation format:

```
/*
Description:
  Description of the procedure

  Usage note: Describe any requirements (such as "calling user must have ACCESS_TOOLS
right") or other things to be aware of.

Inputs:
  myInput1 - Describe the input(s), whether or not it is optional (along with what to
pass to indicate that the input should be ignored, usually NULL), and what will be used
as a default.

Outputs:
  myOutput1 - Describe the output(s)

Exceptions:
  myException1 - Describe any explicitly raised exception(s) (no need to detail every
possible raised exception, just the ones explicitly raised in the code.)

Modified Date:   Modified By:     CSW Version:                    Reason:
mm/dd/yyyy       (Author's Name)  (Lowest supported CIS version)(Reason for change)

Example:
Modified Date:   Modified By:             CSW Version:     Reason:
05/01/2013       Mike Tinius              6.2.3            Created new
08/01/2013       Mike Tinius              6.2.4            Fixed format issue
*/
```

The procedure found in `templates/procedureTemplate()` has comments and annotations already set up for this. Please copy and use this as a basis for all development utilities scripts. (A similar CJP template needs to be set up as well.)

As much as possible, we should provide examples of how utilities are used so that other developers can get a better understanding of how each utility works. Please put such examples in the /shared/ASAssets/Utilities/examples folder.

Regression Test Cases

Along with the new procedure and documentation, a set of regression test cases (sample inputs and expected outputs) will help us automate the process of QA testing new releases and allow us to more easily see if new changes will impact previously known good test cases. Please submit as many of these as possible.

Source Code Control

The master image of the Utilities distribution is currently on http://github.com/TibcoSoftware/ASAssets_Utilities. All official distributions of the Utilities are created from this instance.

CJP source code will be checked into the JavaSource. CIS resources will be exported using the VCS system and checked into the DVSource folder.

Peer Review

Peer review provides a mechanism to reduce duplication of effort and also a way to enforce consistency across all the utilities.

Before being included in the development utilities library, new submissions will be peer reviewed by one or more of the development utilities team. Subsequent changes can be checked in by the original developer but will again be peer reviewed. This peer review will also be conducted whenever a member of the development utilities team submits new or updated procedures.

Team Members

The following are members of the development utilities team:

- Michael Tinius (mtinius@tibco.com)