Overview v4.0

This KSA provides a guideline to use the methods in the Customer Intelligence 360 Utilities Package. This package contains some common functions required to work with the 360 API. These functions will aide developers to achieve the desired output with much lesser lines of code and lesser time for development.

Included files

- CI360Utilities.sas7bdat
  - Compiled sas utilities package
- CI360Utilities.sas
  - SAS program containing the DS2 code to create the package
- SASCI360Parm.sas
  - Contains the environment parameters needed to call CI360Utilities.sas or CI360Utilities.sas7bdat once configured for your environment

External Methods Overview

Method	Description
SetupProxy	This function configures the main HTTP methods used by all other methods to use a proxy server
DiscoverExport	This function requests CI30 to export either the Discover Detail or DBT data marts or both together from 360.
ExportData	This function requests CI360 to export requested data.
ExportDiscoverDBT	This function requests CI360 to export Discover DBT or Reporting data.
ExportDiscoverDetail	This function requests CI360 to export Discover Detail data.
ExportIdentityData	This function requests CI360 to export Discover Identity data.
ImportData	This function requests CI360 to import and process the data file that was uploaded in the temporary cloud location. Call can default to UpdateMode UPSERT or can specify UpdateMode.
RequestUploadPath	This function requests CI 360 for a temporary location in the cloud to upload a data file as a descriptor or an External Event.

Implementation Tasks

1. Save CI360Utilities

When implementing a program that will use one or more of the methods contained in this package, you will need to do one or more of the following:

- Save CI360Utilities.sas7bdat in a directory assessible from the programs that will call the methods OR
- Save CI360Utilities.sas in a directory assessible from the programs that will call the methods
  Note: The SASSRV or other user executing the stored process must have write access to the OS directory where this file lives if it will keep the statement to update the package when the version has been updated.

If the CI360Utilties package will be referenced across many individual programs, a common location accessible to these dependent programs and stored procedures is recommended.
Example: /sas/contexts/common/CI360Utilities

2. Create API User

It is required that you create an API user.

Create one by logging into 360, then navigate to General Settings->External->API Users. Please note the API Username and Secret for use in the API calls.

3. Create External Application

It is required that you create an External Application.

Create one by logging into 360, then navigate to General Settings->External->Applications. Please note the Application ID as you will provide it as an environment parameter.

4. Create General Access Point

In order to use some of the methods, you must first create an Access Point.

Create one by logging into 360, then navigate to General Settings->External->Access. Make sure to associate it with the appropriate ID's. Please note the Tenant ID and Client Secret as you will provide them as environment parameters.

5. Edit Environment Parameters

Environment parameters are available in SASCI360ParmFile.sas This SAS program contains macrovar assignment statements to assign site and tenant specific details and credentials for authorization. These macrovars must be assigned per execution program. Edit this file and save in the same directory as your CI360Utilities if you are using common credentials or assign and store with your calling program.

Name	Description	Sample Value
CI360_ENV	SAS AWS hostname suffix	use.ci360.sas.com
API_USER	API username defined in 360 UI	API-<moniker>-data
API_SECRET	API User secret from 360 UI	ABCDEFGHIJK
TENANT_ID	Tenant ID	da1a105f5300013b4dde1b18
AGENT	The name of the general access point defined in 360 UI	ExternalEvents
AGENT_SECRET	The client secret associated with the general access point from 360 UI	aBcdeFghij…
APPLICATION_ID	The external application ID defined in 360 UI	ExternalEvents
CUST_PROXY_URL	The URL of the proxy server (http://host:port) that the requests must use	http://inetgw.unx.sas.com
CUST_PROXY_USER	Username to access the proxy server (optional)	
CUST_PROXY_PWD	Password to access the proxy server (optional)	
SAS_UTILITY_LIBRARY	Libname given during code execution (optional)	CI_LIB
SAS_UTILITY_PATH	OS directory where DS2 code file lives	/sas/contexts/direct/jobs/common
SAS_UTILITY_FILENAME	SAS program filename for SASCI360Utilities.sas	SASCI360Utilities.sas

6. Calling the Utilities

To use any of these utilities you will need to include a libname assignment for the SAS library with your CI360Utilities package

```
/* Name the SAS CI360Utilities Library */
libname &sas_utility_library. BASE "&sas_utility_libpath.";
```

Alternately, you can include the .sas program and create the package if not found

```
%include "&sas_utility_libpath./&sas_utility_filename.";

%CreatePackage();
```

Detailed examples of how to use SAS code to leverage to each of these utilities is provided within each method detailed section. For each method:

- Declare an object named ci360 to initialize the CI 360 Utilities package.
- Use the appropriate constructor to initialize the Utilities package depending on your method requirements.
- Call the method with the required arguments

```
/* ************************************************************ */
/* EXAMPLE DS2 METHOD CALL for USER BASED API's
/* ************************************************************ */
PROC DS2;
   data work.Result (OVERWRITE=YES);
      declare package &sas_utility_library..CI360Utilities ci360(%tslit(&CI360_ENV),
          %tslit(&API_USER), %tslit(&API_SECRET), &SAS_LOG_LEVEL); /* declare package */
      declare int rc; /* declare return variable */

      method run();/* call the method to run using ci360.method name */
         rc = ci360.ImportData(&DESCRIPTOR_NAME, &DESCRIPTOR_NAME,
              &UPLOAD_URL, 1, &IMPORT_WAIT_MINS);
      end;
   enddata;
RUN;
QUIT;

/* ************************************************************ */
/* EXAMPLE DS2 METHOD CALL for AGENT BASED API'S
/* ************************************************************ */
PROC DS2;
   data work.Result (OVERWRITE=YES);
      declare package &sas_utility_library..CI360Utilities ci360(%tslit(&CI360_ENV),
          %tslit(&TENANT_ID), %tslit(&ACCESS_POINT), %tslit(&ACCESS_POINT_SECRET),
          &SAS_LOG_LEVEL); /* declare package */
      declare int rc; /* declare return variable */

      method run();/* call the method to run using ci360.method name */
         url = ci360.RequestUploadPath('bulkEventsFileLocation');
      end;
   enddata;
RUN;
QUIT;

/* ************************************************************ */
/* EXAMPLE DS2 METHOD CALL for AGENT BASED API'S USING A PROXY
/* ************************************************************ */
PROC DS2;
   data work.Result (OVERWRITE=YES);
      declare package &sas_utility_library..CI360Utilities ci360(%tslit(&CI360_ENV),
          %tslit(&TENANT_ID), %tslit(&ACCESS_POINT), %tslit(&ACCESS_POINT_SECRET),
          &SAS_LOG_LEVEL); /* declare package */
      declare int rc; /* declare return variable */

      method run();/* call the method to run using ci360.method name */
          ci360.SetupProxy(%tslit(&CUST_PROXY_URL), %tslit(&CUST_PROXY_USER),
   %tslit(&CUST_PROXY_PWD));
         url = ci360.RequestUploadPath('bulkEventsFileLocation');
```

```
        end;
    enddata;
RUN;
QUIT;
```

7. Versioning

As updates to this package are made, the version will be incremented, or a new major version will be assigned. Methods exist to return both Version and MajorVersion properties.

- Version changes -e.g., 3.5 release update from prior version 3.4:
  - Limited to only those that occur internally
  - Additional functionality that will not impact consumer scripts using previously published version of the CI360Utilities package
- Major version changes – e.g., 4.0 release update from prior version 3.4:
  - Used when any consumer script would require a specific version – e.g., because of a signature change
  - Fix for a serious defect without which existing programs would not work

Example of how to incorporate a version check in consumer programs:

```
%global rc;
%let min_sas_utility_version = 3.4;

/* check version */
PROC DS2;
    data work.VersionCheck (OVERWRITE=YES);
       declare package &sas_utility_library..CI360Utilities ci360(%tslit(&CI360_ENV),
       %tslit(&API_USER), %tslit(&API_SECRET), &SAS_LOG_LEVEL); /* declare package */
       declare varchar(10) version;

       method run();
          version = ci360.Version();
       end;
    enddata;
RUN;
QUIT;

PROC SQL noprint;
    SELECT version
    INTO   :version separated by ''
    FROM   work.VersionCheck;
QUIT;

%let rc = %eval((&SQLOBS > 0) and (&version >= &min_sas_utility_version));

%if &rc = 1 %then %do; /* RC = 1 when returned version is >= min version */
    /* Insert your code here */
%end;
```

8. Assumptions/Troubleshooting

For Customer environments with encoding defaulting to “LATIN9”, using a log level of 4 may result in a transcoding error. It is recommended to validate the system default encoding and use a log level of 3 (or lower), if system encoding is “LATIN9”.

SetupProxy

SetupProxy(strProxyURL, strProxyUser, strProxyPassword);

This function configures the 360 Utility package to use a proxy server.

- Use the ci360 object to invoke the SetupProxy method in the Utilities package prior to calling any other method.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strProxyURL	varchar(512)	URL host and port of the proxy server	"ExportFile"
strProxyUser	varchar(128)	Username to authenticate with the proxy	scott
strProxyPassword	varchar(128)	password	tiger

Return Variables

None

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
   method run();/* call the method to run using ci360.method name */

            ci360.SetupProxy(%tslit(&CUST_PROXY_URL),
%tslit(&CUST_PROXY_USER), %tslit(&CUST_PROXY_PWD));

            url = ci360.RequestUploadPath('bulkEventsFileLocation',
%tslit(&APPLICATION_ID.));

   end;
```

DiscoverExport

DiscoverExport(strExportName, blnDetails, blnDBTs, tsFrom, tsTo) returns Result;

This function requests CI30 to export either the Discover Detail or DBT data marts or both together from 360.

- Use the ci360 object to invoke the **DiscoverExport** method in the Utilities package.
- Upon execution, the function would return an integer with the number of files exported.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
blnDetails	int	Set to 1 if you want to download Detail Mart.	1 or 0
blnDBTs	int	Set to 1 if you want to download DBT Mart.	1 or 0
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.DiscoverExport(%tslit(&EXPORT_NAME),1,0, %tslit(&TIME_FROM),
%tslit(&TIME_TO));
```

ExportData

ExportData(strExportName, strDescriptorID, tsFrom, tsTo, intWaitMins) returns Result;

This function requests CI 360 to export requested data.

- NOTE: this method will be deprecated in 2020. Use with caution.
- Use the ci360 object to invoke the ExportData method in the Utilities package.
- Upon execution, the function would return an integer with the number of files exported

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
strDescriptorID	varchar(2048)	ID of the Export Descriptor	"a2e9bcfc-ed5a-4294-a30c-6c6ed117634a"
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z
inWaitMins	int	Number of minutes to wait before checking whether the Export file is ready	Any integer like 1, 2, 3

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportData(%tslit(&EXPORT_NAME), %tslit(&DESCRIPTOR_ID),
%tslit(&TIME_FROM), %tslit(&TIME_TO), 1);
```

ExportDiscoverDBT

ExportDiscoverDBT(strExportName, tsFrom, tsTo) returns Result;

This function requests CI360 to export Discover DBT or Reporting data.

- Use the ci360 object to invoke the **`ExportDiscoverDBT`** `method in the Utilities package.`
- `Upon execution, the function would return an integer with the number of files exported.`

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportDiscoverDBT(%tslit(&EXPORT_NAME));
```

ExportDiscoverDBT with Independent Boolean

ExportDiscoverDBT(strExportName, tsFrom, tsTo, blnIndependent) returns Result;

This function requests CI360 to export Discover DBT or Reporting data.

- Use the ci360 object to invoke the **`ExportDiscoverDBT`** `method in the Utilities package.`
- `Upon execution, the function would return an integer with the number of files exported.`

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z
blnIndependent	Int	States whether the download is being run independently or as part of a larger request	1=independent

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportDiscoverDBT(%tslit(&EXPORT_NAME), %tslit(&TIME_FROM),
%tslit(&TIME_TO), 1);
```

ExportDiscoverDetail

ExportDiscoverDetail(strExportName, tsFrom, tsTo) returns Result;

This function requests CI360 to export Discover Detail data.

- Use the ci360 object to invoke the ExportDiscoverDetail method in the Utilities package.
- Upon execution, the function would return an integer with the number of files exported.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportDiscoverDetail(%tslit(&EXPORT_NAME));
```

ExportDiscoverDetail with Independent Boolean

ExportDiscoverDetail(strExportName, tsFrom, tsTo, blnIndependent) returns Result;

This function requests CI360 to export Discover Detail data.

- Use the ci360 object to invoke the ExportDiscoverDetail method in the Utilities package.
- Upon execution, the function would return an integer with the number of files exported.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"
tsFrom	timestamp	The start time (in UTC format) of data that you want to export	2020-01-01T00:10:10Z
tsTo	timestamp	The end time (in UTC format) for data that you want to export	2020-01-03T00:10:10Z
blnIndependent	Int	States whether the download is being run independently or as part of a larger request	1=independent

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating the number of files exported.	O=None >0=Number of files exported

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportDiscoverDetail(%tslit(&EXPORT_NAME), %tslit(&TIME_FROM),
%tslit(&TIME_TO), 1);
```

ExportIdentityData;

ExportIdentityData(strExportName) returns Result;

This function requests CI 360 to export requested data.

- Use the ci360 object to invoke the ExportIdentityData method in the Utilities package.
- Upon execution, the function would return an integer with the number of files exported.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strExportName	varchar(256)	Name of the Export File	"ExportFile"

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating the number of files exported.	O=None

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
result = ci360.ExportIdentityData(%tslit(&EXPORT_NAME));
```

ImportData Upsert Only UpdateMode

ImportData(strUploadName, strDescriptor, strLocation, blnHeaderRow, intWaitMins) returns Result;

This function requests CI360 to import and process the data file that was uploaded in the temporary cloud location with UpdateMode UPSERT function.

- Use the ci360 object to invoke the ImportData method in the Utilities package. Default update mode in this function is "Upsert".
- Upon execution, the function would return an integer whether the import was successful or not.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strUploadName	varchar(256)	Name of the File to Upload	"FileForUpload"
strDescriptor	varchar(2048)	Name of the Descriptor to where the file needs to be uploaded	"DescriptorABC"
strLocation	varchar(2048)	The temporary signed URL that is generated when you request the file transfer location in the RequestUploadPath function	A signedURL in the format: "https://<*temporary URL*>"
blnHeaderRow	Int	Set to true if the first row contains header information instead of a data record.	1 = header row present 0 = no header row
inWaitMins	Int	Number of minutes to wait for the function to keep checking for the status of the Import.	Any integer like 1, 2, 3

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating whether the Import was successful or not.	1 = Success or 0 = Fail

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
rc = ci360.ImportData(%tslit(&descriptor_name), %tslit(&descriptor_name),
              %tslit(&upload_url), 1, 60);
```

ImportData with UpdateMode Argument

ImportData(strUploadName, strDescriptor, strLocation, strUpdateMode, blnHeaderRow, intWaitMins) returns Result;

This function requests CI360 to import and process the data file that was uploaded in the temporary cloud location with with user defined UpdateMode (REPLACE or UPSERT).

- Use the ci360 object to invoke the ImportData method in the Utilities package.
- Upon execution, the function would return an integer whether the import was successful or not.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strUploadName	varchar(256)	Name of the File to Upload	"FileForUpload"
strDescriptor	varchar(2048)	Name of the Descriptor to where the file needs to be uploaded	"DescriptorABC"
strLocation	varchar(2048)	The temporary signed URL that is generated when you request the file transfer location in the RequestUploadPath function	A signedURL in the for mat: "https://<*temporary URL*>"
strUpdateMode	varchar(36)	**replace**: This mode replaces all the data from previous uploads to this descriptor. Replace mode is valid for data that uses these data descriptor types: • deleteList • importedList • lookup • transient • transientDeviceIdList **upsert**: This mode updates any existing records and appends new records. The upsert mode updates values based on checking data items that have the key parameter set. Upsert mode is valid for data that uses these data descriptor types: • customer • contact_preference • offers	"replace" or "upsert"
blnHeaderRow	Int	Set to true if the first row contains header information instead of a data record.	1 = header row present 0 = no header row
inWaitMins	Int	Number of minutes to wait for the function to keep checking for the status of the Import.	Any integer like 1, 2, 3

Return Variables

Variables	DataType	Details	Sample Value(s)
Result	Int	Returns an integer value stating whether the Import was successful or not.	1 = Success or 0 = Fail

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
rc = ci360.ImportData(%tslit(&descriptor_name), %tslit(&descriptor_name),
             %tslit(&upload_url), %tslit(&update_mode), 1, &import_wait_mins);
```

RequestUploadPath with Parameterized Method

RequestUploadPath(strMethod) returns URL;

This function requests CI 360 for a temporary location in the cloud to upload a data file.

- Use the ci360 object to invoke the RequestUploadPath method in the Utilities package.
- The strMethod parameter should be defaulted to "fileTransferLocation".
- Upon execution, the function would return the temporary file transfer location to the string variable url.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strMethod	varchar(100)	Name of the Export File	"fileTransferLocation"

Return Variables

Variables	DataType	Details	Sample Value(s)
URL	varchar(2048)	Returns a signed URL of the File Transfer location.	A signedURL in the format: "https://<*temporary URL*>"

Sample Calling Program

```
/* ************************************************************ */
/* SAMPLE
/* ************************************************************ */
url = ci360.RequestUploadPath(fileTransferLocation);
```

RequestUploadPath with Parameterized Method

RequestUploadPath(strMethod, strApplicationID) returns URL;

This function requests CI 360 for a temporary location in the cloud to upload the External Events data file.

- Use the ci360 object to invoke the RequestUploadPath method in the Utilities package.
- Pass '`bulkEventsFileLocation`' value to the strMethod parameter.
- Requires agent and agent secret in the environment parameters file.
- Pass applicationID from your 360 UI to the strApplicationID parameter.
- Upon execution, the function would return the signed S3 temporary file transfer location to the assigning string variable URL.

Required Arguments

Parameter	DataType	Details	Sample Value(s)
strMethod	varchar(2048)	Name of the Export File	"ExportFile"
strApplicationID	Varchar(100)	Application ID from the 360 UI	"ExternalEvent"

Return Variables

Variables	DataType	Details	Sample Value(s)
URL	varchar(2048)	Returns a signed URL of the File Transfer location.	A signedURL in the format: "https://<*temporary URL*>"

Sample Calling Program

```
url = ci360.RequestUploadPath('bulkEventsFileLocation' ,%tslit(&Application_ID.));
```