|  |
| --- |
|  |
| GFI_logo1 |

**DDIC Risk Assesment**

**EXTGFI0188**

*Technical Analysis*

INDEX

1. Document revision history 3

1. REFERENCES 3

2. PURPOSE 3

3. BENEFITS 3

4. SAP artifacts 3

4.1 Tables 4

4.2 ABAP Objects 4

4.2.1 Classes 5

4.2.1.1 Attributes 5

4.2.1.2 Methods 7

4.2.1.3 DDIC Risk Categorisation 8

4.2.2 Purpose: Interfaces 9

4.2.3 Function Modules 9

4.2.3.1 ZFM\_DDIC\_ANALYSIS 10

4.2.3.2 ZFM\_CRITICAL\_BATCH\_ANALYSIS 11

4.3 Annexes 12

4.3.1 PULL\_TECHNICAL output example 12

4.3.2 PULL\_DB2 output example 12

4.3.3 PULL\_FUNCTIONAL output example 12

# Document revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changed By** | **Reviewed By** | **Comments** |
| 1.0 | 26/05/2019 | GFI | Adrián Romero-Dapena | Definition of Class and rest of complementary objects |
|  |  |  |  |  |
|  |  |  |  |  |

1. REFERENCES

**Business Owners contacted:** Stuart Dinnen, Klara Falcone.

1. PURPOSE

The purpose of this development is to build a class which can produce an automated DDIC risk assessment on its own. This class will be able to compare the DDIC objects in a list of Transport Requests and return a table with all the technical details for its evaluation by the caller.

The objective is to build a class which can be integrated in reports or other processes so its outputs can be useful not only for a unique report, but for a set of tools.

1. BENEFITS

The automation of the DDIC risk assessment would aim at standarising the way the risk level for DDIC objects is dictated.

By doing this assessments manually we take the risk of missing dangerous objects from being anaysed or being analysed incorrectly with the potential issue that this could produce in the target system when getting deployed.

1. SAP artifacts

New objects which are going to be delivered:

* In SOS:
  + Class: ZCL\_DDIC\_ANALYSIS
  + Interface: ZIF\_DDIC\_ANALYSIS
* In UAD/CAD:
  + Funcion Modules:
    - ZFM\_DDIC\_ANALYSIS
    - ZFM\_CRITICAL\_BATCH\_ANALYSIS
  + DDIC Objects:
    - Table Types:
      * ZTTY\_DDIC\_ANALYSIS
      * ZTTY\_USAGE
      * ZTTY\_DD01V
      * ZTTY\_DD03V
      * ZTTY\_DD04V
      * ZTTY\_DD40V
      * ZTTY\_DD12V
    - Structures:
      * ZTY\_DDIC\_ANALYSIS

* 1. Tables

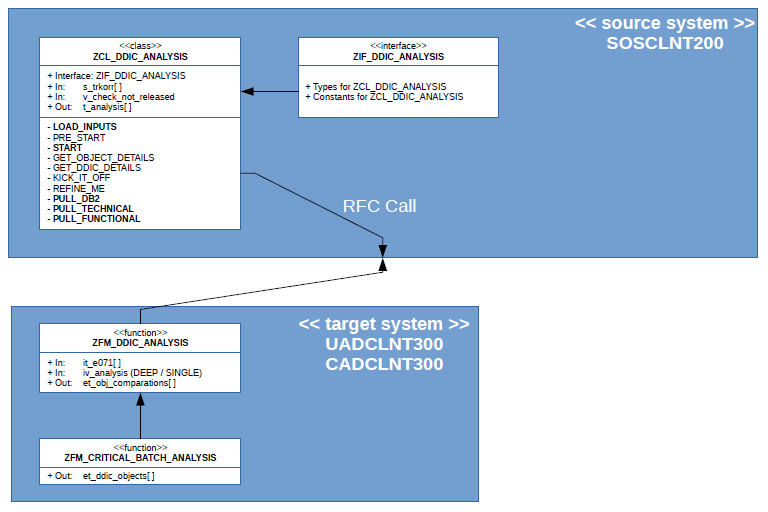
No database tables will be needed for this automation.

* 1. ABAP Objects

Below, there is an UML chart to relate the main objects of this development. The main object is the class ZCL\_DDIC\_ANALYSIS which makes use of the interface ZIF\_DDIC\_ANALYSIS for all the Types and Constants which are used among the Public and Private methods.

The class calls through RFC connection a FM in the target systems to pull the raw DDIC details that will be later categorised.

The main FM ZFM\_DDIC\_ANALYSIS calls within the source system another FM ZFM\_CRITICAL\_BATCH\_ANALYSIS that serves the purpose of providing the Critical Objects used by batches (going only one level in depth).



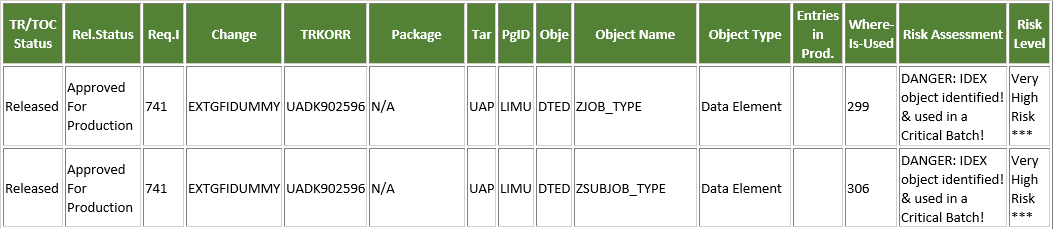
* + 1. Classes

Class ZCL\_DDIC\_ANALYSIS will be using the following objects:

* Interface: ZIF\_DDIC\_ANALYSIS
* Funcion Module: ZFM\_DDIC\_ANALYSIS (RFC enable)
  + - 1. Attributes

The set of public attributes of the class is as it follows:

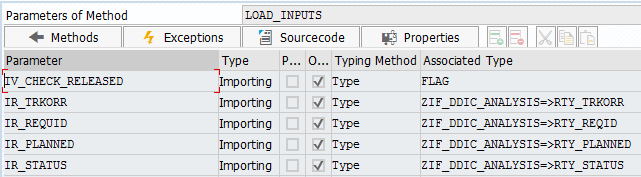
* **Inputs:**
  + S\_TRKORR:
    - Type: Range of TRKORR (ZIF\_DDIC\_ANALYSIS=>RTY\_TRKORR)
    - Purpose: List of TRs/TOCs to be analysed
  + V\_CHECK\_NOT\_RELEASED:
    - Type: Boolean
    - Purpose: Activate the analysis of non-released TRs/TOCs.
      * When True: Analyses the sub-tasks of TRs/TOCs not released yet of the original list of TRs/TOCs
      * When False: It only analyses the provided list of TRs/TOCs
* **Outputs:**
  + T\_ANALYSIS:
    - Type: Table (ZIF\_DDIC\_ANALYSIS=>TTY\_OUTPUT)
    - Purpose:
      * Return to the caller the analysis of the list of TRs/TOCs provided.
      * Other details will be provided along with Risk Assessment & Risk Level, such as the ones in the screenshot below:



* **Middle-Attributes**: (Not meant to be consumed by caller, but still Public in case of need)
  + T\_SUPPORTED[ ]:
    - Type: Internal Table
    - Populated: *<< private method >> \_get\_supported\_trs( )*
    - Purpose: Provides patterns for the supported TRs
  + T\_RFCDEST[ ]:
    - Type: Internal Table
    - Populated*: << private method >> \_get\_destinations( )*
    - Purpose: Get a list of RFC destinations
  + T\_E070[ ]:
    - Type: Internal Table
    - Populated: *<< private method >> \_fetch\_e070\_data( )*
    - Purpose: for the list of TRs/TOCs provided, gets only existing ones in target systems and fetches as well its Sub-Tasks in case main TRs/TOCs are not released yet.
  + T\_E071[ ]:
    - Type: Internal Table
    - Populated: *<< private method >> \_fetch\_e071\_data( )*
    - Purpose: Hold all objects coming from TRs/TOCs or their Sub-Tasks
  + T\_DDIC\_DATA[ ]:
    - Type: Internal Table
    - Populated: *<< function >> ZFM\_DDIC\_ANALYSIS*
    - Purpose: Hold the raw DDIC analysis data coming from source systems
  + T\_HPALM\_DATA [ ]:
    - Type: Internal Table
    - Populated: *<< public method >> pre\_start( )*
    - Purpose: Hold the HPALM details of table ZXX\_HPALM\_DATA for the list of TRs/TOCs provided by caller
  + T\_TR\_DEF\_CR [ ]:
    - Type: Internal Table
    - Populated: *<< public method >> pre\_start( )*
    - Purpose: Hold the TR details of table ZXX\_TR\_DEF\_CR for the list of TRs/TOCs provided by caller
      1. Methods

In the methods section we will be only covering the public methods which are open to the caller.

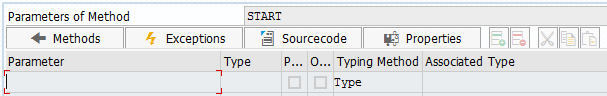
* **Input Methods:**
  + LOAD\_INPUTS:
    - Purpose: provide the search criteria to obtain the list of TRs to be analysed or the final list itself directly
    - Internal calls: It calls internally PRE\_START method
    - Signature:



* + - Example of use: Method Signature has only importing parameters (5)

o\_analysis->load\_inputs**(**ir\_trkorr  **=**s\_trkorr[]  
                         ir\_requid  **=**s\_reqid[]  
                         ir\_planned **=**s\_datum[]  
                         ir\_status  **=**s\_status[] **).**

* **Operations methods:**
  + START:
    - Purpose:
      * To be activated by caller
      * It starts the gathering of the details of the Middle-Attributes
      * The end ouput will be to populate all necessary details to kick off the analysis itself later on
    - Internal calls: It calls internally the following public methods
      * GET\_OBJECT\_DETAILS
      * GET\_DDIC\_DETAILS
      * KICK\_IT\_OFF
      * REFINE\_ME
    - Signature: No parameter. Activation method. They should have to be previously loaded with LOAD\_INPUTS method



* + - Example of use:

o\_analysis->start**( ).**

* **Public Internal Methods**
  + GET\_OBJECT\_DETAILS
    - Purpose: Fetch details related to TRs/TOCs like E070 / E071 details
  + GET\_DDIC\_DETAILS
    - Purpose: Fetch the raw DDIC details to be used as input for the assessment
  + KICK\_IT\_OFF:
    - Purpose: Produce the raw assessment analysis without refinements (sort, presentation, … )
  + REFINE\_ME:
    - Purpose: Produce the final assessment analysis completely refined (sorted, HPALM details added, only showing TRs/TOCs from input (not Sub-Tasks), adding Package information, Adding information on whether TRs/TOCs were released… )

* **Output methods:**
  + PULL\_TECHNICAL:
    - Purpose:
      * Provides a table with a full set of TRs/TOCs with all the technical details of the DDIC Risk Assessment to caller.
      * Annex 4.3.1 for sample of output
  + PULL\_DB2:
    - Purpose:
      * Provides a table with less information that the PULL\_TECHNICAL output since the outcome of this returning method is thought specially for the DB2 Team communications
      * Annex 4.3.2 for sample of output
  + PULL\_FUNCTIONAL:
    - Purpose:
      * Provides an output based on HPALM Changes / Requirement IDs instead of the ones before (based on the TRs/TOCs). The output will be a list of Changes / Requirment IDs related to the list of TRs / TOCs with a rough categorisation of the highest DDIC risk found on the change.
      * Annex 4.3.3 for sample of output
      1. DDIC Risk Categorisation

There are going to be 4 categories for the DDIC Risk Assessment:

1. **Very High Risk:**
   * Any IDEX object (They are only found in UAD)
   * Any Critical Batch Object
2. **High Risk:**
   * Indexes: Whether they are new or modifications
   * New Key Fields in table
   * Changes of order in the middle of the structure of a [Table / Table Type / Structure]
     1. Swapping the current order without adding / removing fields
     2. Adding new field in middle of the structure
     3. Removing existing field from the middle of the structure
   * Some types of changes on data elements of the fields:
     1. Change of Type [Ex. From CHAR to INT]
     2. Change of Length when Development Length is shorter than Production Length
3. **Low Risk:**
   * Any brand new object (Except if it is an Index object, used in a Critical Batch or IDEX)
   * Any fields added at the end of a [Table / Table Type / Structure]
   * Change of Length of Data Element when the change is an increase of the length:
     1. When Development Length is larger than Production Length
4. **Out of scope:**
   * Any DDIC objects below will not be analysed in depth, since they are not risky at all in any regard when it refers to Transport actions, although SAP may consider the object, still, as DDIC:
     + TOBJ: Maintenance and Transport Object
     + ENQD: Lock Object Definition
     + TABU: Table Contents

The DDIC Risk Level can be highlighted with the addition of [\*\*\*] pattern at the end of the categorisation.

When this happens, means that the object has trespassed one of the following thresholds:

* **Entries in Production:** 
  + Number of entries in production of a table
  + Even if the object is categorised as “Low Risk” if the volume of records of the table in production is really large it can be a conflictive transport
* **References of use in Development:**
  + Number of references of use of any object in Development System
  + Even if the object is categorised as “Low Risk” in the first instance, if it is being used in a lot of other objects in case of error during the transport the impact can be higher.

These two thresholds can be set in table TVARVC by changing the following variants through tcode STVARV:

* Z\_DDIC\_ANALYSIS\_NUM\_ENTRY\_PROD
* Z\_DDIC\_ANALYSIS\_WHERE\_USE\_REF
  + 1. Purpose: Interfaces

Interface object ZIF\_DDIC\_ANALYSIS has been created to provide support to class ZCL\_DDIC\_ANALYSIS on the type declarations and other constants used throughout the whole logic of the development.

This will help also the caller to obtain the needed types to consume any of the outputs produced by the class without need of creating permanent DDIC objects

The interface will act as a connector of all the methods, since the importing, exporting, changing and returning parameters of Public & Private methods are based on these types.

* + 1. Function Modules

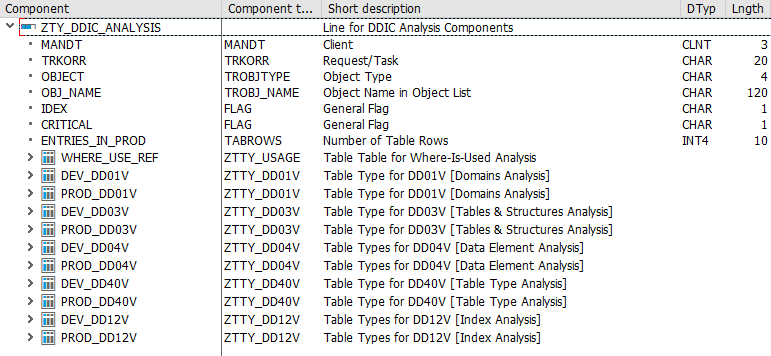
Two function modules have been created to help class ZCL\_DDIC\_ANALYSIS to obtain the raw DDIC details of each DDIC object found in the list of TRs/TOCs provided by the caller.

* + - 1. ZFM\_DDIC\_ANALYSIS

The purpose is to have a table as output with a deep structrure embedded on it.

This means that each row of the output table produced by FM ZFM\_DDIC\_ANALYSIS will contain several deep elements on some of its fields.

See below a the distribution of the structure of the output table produced by ZFM\_DDIC\_ANALYSIS:



Each line will be related to an unique relationship OBJECT-TRKORR. This means that an object could be analysed several times if present in different TRs/TOCs which have been provided as inputs to the caller class.

The explanation of each field is as it follows:

* MANDT: Mandate of current system
* TRKORR: TR/TOC/Sub-Task to which the DDIC object belongs to
* OBJECT: Type of object
* OBJ\_NAME: Name of the object
* IDEX: Boolean. True if OBJ\_NAME is an IDEX object.
* CRITICAL: Boolean. True if OBJ\_NAME is used in a Critical Report
* ENTRIES\_IN\_PROD: Number of entries of OBJ\_NAME in Production system (only applyable to Table objects)
* WHERE\_USE\_REF: Table with the reference of use of OBJ\_NAME in Development System
* \*\_DD01V: Development & Production details of Domains objects
* \*\_DD03V: Development & Production details of Table & Structure objects
* \*\_DD04V: Development & Production details of Data Element objects
* \*\_DD40V: Development & Production details of Table Type objects
* \*\_DD12V: Development & Production details of Index objects

To obtain the details of field CRITICAL we check the OBJ\_NAME against the outputs of the FM ZFM\_CRITICAL\_OBJECTS\_ANALYSIS.

* + - * 1. Logic of the Function Module

The logic of the FM is to pick up all the details from the main standard tables of the system that hold the details of the different types of DDIC objects.

For each object, the FM will identify the type of object that we are currently handling (Data Element, Domain, Table, … ) and depending on the type it will pick up the details of the corresponding standard table of Development & Production systems for the last version of the object.

Once done this, the details will be added to the structure of the output and this row appended to the final output table.

As consequence, the FM will produce a compilation of raw details to be analysed by the caller class.

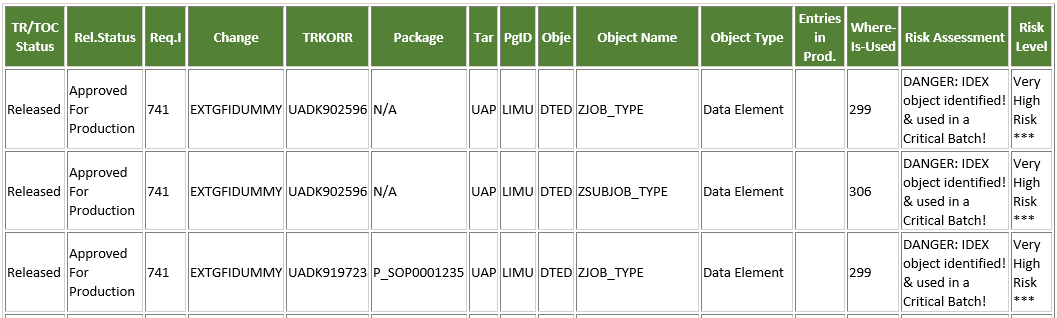
There are two levels of analysis: Deep and Single

* Deep analysis: It will gather the references of use and it will also look for any details in tables, table types, structures and other deep object for the details of their basic elements (A data element, for example, might make references to a domain).
  + - 1. ZFM\_CRITICAL\_BATCH\_ANALYSIS

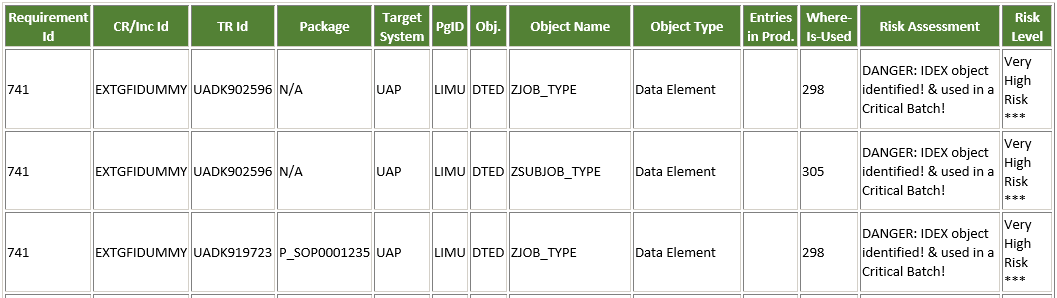
This FM does not have any importing parameter. It simply fetches from Production any DDIC object used directly in a report that belongs to a Critical Batch Execution.

The output is a list of objects that we can compare against a list of candidates to be analysed.

* 1. Annexes
     1. PULL\_TECHNICAL output example



* + 1. PULL\_DB2 output example



* + 1. PULL\_FUNCTIONAL output example

