Contents

[CRT Delta Extraction Tool 2](#_Toc42516831)

[Description: 2](#_Toc42516832)

[Interface: 2](#_Toc42516833)

[CRT Repository: 2](#_Toc42516834)

[Behavior: 3](#_Toc42516835)

# CRT Delta Extraction Tool

## **Description:**

Delta Extraction tool is a tool designed to perform 2-way comparison of baseline and the latest versions of AML file to generate output AML file containing only importable different made in the latest version of the file.

The tool goal is to reduce usage of ***Fixes*** folder and make structure/usage of AML-packages folder clearer.

Right now, the tool supports only console line interface which can be used by CRT build procedure (or any other) during import package generation in CI or generation of deliverable package.

## **Interface:**

You can run application by passing the following command line options each of which contains its own set of options:

* Repository - contains options to build delta between 2 commits in Git
  + '--PathToRepository', '-**p**' - path to folder in Git repository which is used to build delta
  + '--FromCommit', '-**f**' - initial commit to build delta
  + '--ToCommit', '-**t**' - last commit to build delta (will be used Id of the 'HEAD' commit of the current branch if this option was not passed)
* FileSystem - contains options to compare files in file system  
  \*(NOT SUPPORTED. Reserved for the future when this feature will be implemented)\*
  + '--PathToBaselineFolder', '-**b**' - path to folder which contains original version of files
  + '--PathToModifiedFolder', '-**m**' - path to folder which contains modified files
* For each of these groups exists the following common options:
  + '--OutputFolder', '-**o**' - folder to save delta
  + '--ComparisonMode', '-**c**' - comparison mode of baseline and latest version to get delta with added/modified or deleted items currently 2 modes are supported:
    - direct - use to get delta containing Added/Modified/Renamed changes
    - reverse - use to get delta containing Deleted changes
* '--SkipTypes', '-**s**' - Types of 'Item' nodes, separated by comma, which will be skipped during delta extraction. So, they will contain all the child nodes in result AML.

Example of using:

* DeltaExtractionCmd.exe Repository -p "C:\TestProject\AML-packages" -f f70e741a59aa401d03e93a207ad0d3ca8cd60168 -t a702525e88e78636a6e29c49ba7d15378dccb124 --OutputFolder "D:\DeltaExtractionResult" -s "Workflow Map Activity,Field"

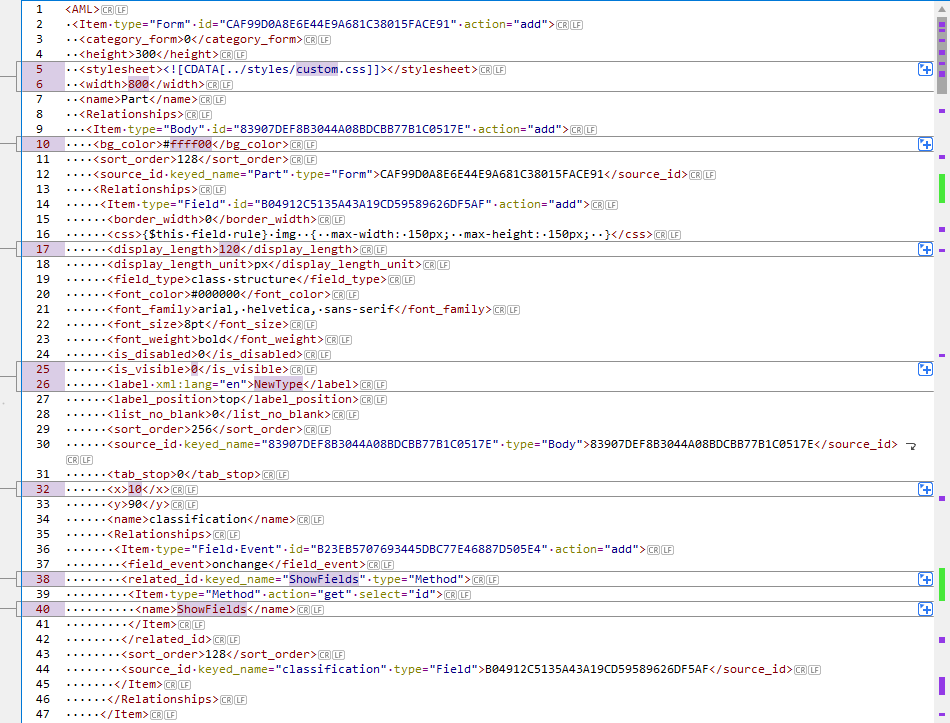
## **CRT Repository:**

To enable DeltaExtraction tool in CRT builds set "Use.Delta.Extraction.Tool" property value to "true" in "AutomatedProcedures\Default.Settings.include" file.

During development process DeltaExtraction tool can be used in 2 cases:

1. **Getting of direct delta (containing Added/Modified/Renamed changes). Occurs automatically during CRT build procedure.**  
   The result will be the same build archive, but AML’s in ‘AML-packages’ folder of this archive will not contain full AML structure, but only changes done between HEAD and baseline commit (Last.Commit.Before.Current.Release property in ..\AutomatedProcedures\Default.Settings.include).
2. **Getting of reverse delta (containing Deleted changes) should be run manually. ExtractDeletedDelta.bat is used for that purpose.**  
   The result is a ‘package’ containing AML’s for items which were deleted. The ‘package’ repeats ‘AML-packages’ folder structure and is placed in following folder: ..\AutomatedProceduresOutput\TemporaryDeletedItems.   
   By default, reverse delta includes changes done between HEAD and baseline commit (Last.Commit.Before.Current.Release). But it also could be calculated for user specified number of last commits – once ExtractDeletedDelta.bat is run, user is asked to enter number of commits. In case user push Enter button without specifying any number, default settings will be used.   
   After getting of reverse delta user should check and move AML’s to appropriate folder (1-BeforeAmlPackagesImport or 2-AfterAmlPackagesImport) and rename them accordingly.

## **Behavior:**

* Delta Extraction tool analyzes 2 version of the AML file: baseline one and the latest one;
  + The output AML file is based on the latest version where we remove all equal-to-baseline nodes;
* It does "intellectual" search of items in the file. Which means order of items and indents do not affect the output content, however sequence of output items depends on the input;
  + Uniqueness of an item is defined by the following parameters: element-name + type-attribute + id-attribute + where-attribute + action-attribute + (in case of a relationship item) sort-order-property;
  + It ignores and leave in the output whole items with types mentioned in --SkipTypes input argument;
* The tool bypasses all nodes in the latest document and correlates nodes in the latest version of the file with nodes in the baseline version of the file;
  + Nodes are corrected in case uniqueness values are equal + uniqueness values for all parents are also equal;
* When the tool compares 2 correlated nodes it uses the following logic: we consider 2 correlated nodes equal in case all attributes and their values are the same + content of the nodes is equal;
  + Content of the node might be a text (in case it is a property) or other node(s) (in case it is a property item, Relationships node or etc.);
* In case a corrected node is equal - the tool removes it from the output content (there is an exception which will be mentioned later);
* In case node is modified or it contains modified children - the tool doesn't remove this node;
  + However, all non-modified children are removed anyway;
* Exception:
  + In case there is an item with **"get"** action in the structure it is removed from the output **only** in case all its children are equal. In case **at least one** of its children is different the **whole "get"** item is remained in the output.
* Example:
* Input (baseline version on the left, the latest version on the right)  
    
  
* Output:  
  