Altova EBA Add-in for Excel, Version 2024r2
Enterprise Edition

EBA
ALTOVA®
European Banking Authority
(EBA) XBRL add-in for Excel

User & Reference Manual

Altova EBA Add-in for Excel, Version 2024r2
Enterprise Edition
User & Reference Manual

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2024

Table of Contents

1 Introduction

The **Altova® European Banking Authority (EBA) XBRL Add-in for Excel** (or EBA Add-in for short) enables your organization to prepare XBRL reports that conform to the EBA (European Banking Authority) XBRL taxonomy or to related country-specific XBRL taxonomies.

The EBA XBRL Add-in allows you to do the following:

- Enter XBRL data in Microsoft Excel, using a predefined template spreadsheet which maps to the XBRL taxonomy
- Validate the report data directly from Excel to ensure it conforms to the XBRL taxonomy
- Export report data from Excel to XBRL format
- Import data from existing XBRL reports into Excel
- Batch convert XBRL files to Excel (`.xlsx`) format (*Enterprise Edition*)

XBRL taxonomies

The list of supported XBRL taxonomies is periodically updated to include newer versions, independently of Altova add-in releases. Country-specific XBRL taxonomies and older versions of the EBA XBRL taxonomy are not installed by default when you install the add-in. You can view, install, upgrade, and uninstall taxonomies on demand, using the XBRL Taxonomy Manager[30].

General taxonomies

The following general, non-country-specific taxonomies are supported:

- EBA (European Banking Authority) XBRL Taxonomy (starting with version 2.0 up to the most recent version)
- SFRDP (Supervisory Financial Reporting data Points) provided by the ECB
- SRB RES REP: Single Resolution Board Resolution Reporting for liability data, critical functions and financial market infrastructures
- SRB SRF EAC: Single Resolution Board Ex-Ante Contributions to the Single Resolution Fund

Country-specific taxonomies

The following country-specific taxonomies are supported:

- ACPR (Banque de France Autorité de Contrôle Prudentiel et de Résolution)
  - LCB-FT: Combating Money Laundering and the Financing of Terrorism
  - RUBA: Unified Reporting Banks and Assimilates
  - COREP: Common Reporting with the modules LCR_CON, LCR_DA, NSFR_CON and ALM_CON
  - CREDITIMMO: Reporting for home loans and the profitability of production and outstanding home loans in France
- BBK (Deutches Bundesbank)
  - EBA-ITS-Reporting extended by the German base taxonomy
- BDP (Banco de Portugal)
  - Financial and Common Reporting framework

- BOE (Bank of England)
    - BANKING: Regulatory reporting for the banking sector
    - STATISTICS: Statistical reporting obligations
- CBI (Central Bank of Ireland)
    - FSP: Fund Service Providers, Investment Firms and other Payment Institutions
- DNB (De Nederlandsche Bank)
    - BISCBS: BIS Consolidated Banking Statistics
    - BSI: Balance Sheet Items Taxonomy
    - DGS: Deposit Guarantee Scheme Reporting Framework
    - FBO: Taxonomy for financial reporting for investment funds and fund managers
    - FEH: Foreign Equity Holdings
    - MESREP: Macroeconomic Statistics Reporting
    - MIR: MFI Interest Rates
    - MSR: Taxonomy for the Monthly Securities Reporting
    - OTC: Derivatives Taxonomy
    - PAY: Payment Statistics Reporting Framework
    - PLA: Profit and Loss Account Taxonomy
    - PPSP: Taxonomy for Payment Processing Service Providers
    - SPV: Special Purpose Vehicles Taxonomy
- NBB (National Bank of Belgium)
    - BANKING: Domain MBS-XBRL reporting: FINREP, TREP

This documentation

This documentation should be read in conjunction with the supporting documents included in the EBA XBRL Taxonomy:

- *Description of DPM formal model*
- *EBA Architecture for XBRL representation of DPM*
- *EBA XBRL Filing Rules*

Last updated: 8 April 2024

1.1 System Requirements

To install and run the add-in, note the following system requirements:

- Windows 10, Windows 11, Windows Server 2016 or newer
- Microsoft Excel 2010 and later
- Visual Studio 2010 Tools for Office Runtime
- .NET Framework 4.8 or later

Important

- The add-in is available for Microsoft Excel 32-bit and 64-bit. Microsoft Excel 64-bit is recommended if you need to load big taxonomies such as COREP CON, COREP IND, FINREP. Otherwise, you may see out-of-memory errors when you attempt to load such taxonomies with Microsoft Excel 32-bit.
- The add-in requires full access to the Excel document in order to create, validate, and export XBRL reports. If your organization enforces Information Rights Management (IRM) using Azure Information Protection or a similar technology, access to the Excel document may be restricted. For information about how to allow code to run behind documents with restricted permissions, see the Microsoft documentation.

1.2 Installation and Licensing

To install the EBA XBRL Add-in, download the executable from the Altova Download Center. Run the executable and follow the steps to complete the installation. You will need to accept the license agreement and privacy policy in order to proceed with the installation. Make sure to download the executable corresponding to your operating system and Excel platform (32-bit or 64-bit). The 32-bit executable can be installed on both 32-bit and 64-bit Windows. However, the 32-bit executable supports only Excel 32-bit. Note that If you have Excel 32-bit and install the 64-bit version of the add-in, you will still be running the 32-bit version.

After the installation, a new tab called **EBA** becomes available in the Excel ribbon. To view the current version of the add-in, open the add-in tab in the Excel ribbon and click **About EBA Add-In**.

Possible installation issues

After the installation of the add-in, when you first launch Excel, the add-in's DLL files are copied to **`C:\Users\<UserName>\AppData\Local\assembly\dl3`**. The Microsoft ASR rules might block the execution of the DLL files from that location and the installation of the add-in. If you experience problems with the installation of the add-in, make sure to check the group policies configured for your system.

Licensing

To be able to use the EBA XBRL Add-in, you need a valid license key code. To purchase a new key code or request a free evaluation from the Altova website, open the add-in ribbon and click **Add-In Activation**. You will see instructions on how to get a new license from Altova or manage an existing license.

After you purchase a license from Altova, follow the same steps as above to open the activation dialog box and upload the license file. Alternatively, you can upload the purchased licenses to Altova LicenseServer running on your organization's network. Altova LicenseServer is a free product that helps organizations manage all Altova licenses in a centralized place. See also License Information 94.

2 Create a New Report

This section will help you get started with an EBA report. The section is organized into the following topics:

- New Report 11
- Enter Data 17
- Enter Data into 3D Tables 21
- Control Accuracy of Cells 23
- Validate Data 24
- Export Data to XBRL 26
- Import Data from XBRL 27
- Batch convert XBRL to Excel 28

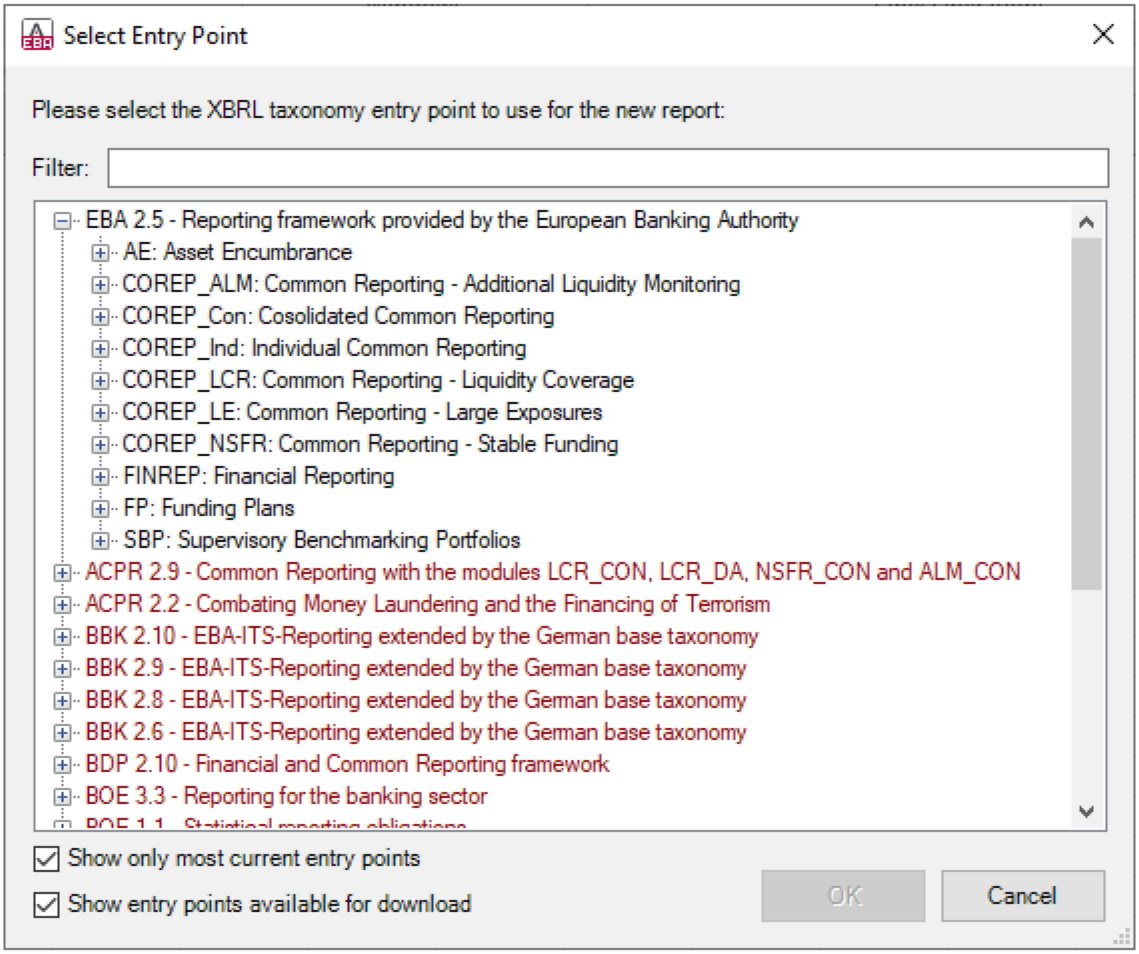
2.1 New Report

The instructions below show you how to prepare a new XBRL report based on the default EBA taxonomy available in EBA XBRL Add-in. This XBRL taxonomy is installed by default on your computer when you install the add-in. Additional taxonomies can be installed separately. For details, see XBRL Taxonomy Manager 30.

New report

To create a new report, take the following steps:

1. Click the add-in tab in the Excel ribbon.
2. Click **Insert New Report**. This will open the **Select Entry Point** dialog.
3. Select the taxonomy entry point corresponding to the report you want to create (*see screenshot below*). Use the filter at the top of the dialog box to filter entry points by keywords. By default, only the most current entry points for the current version of the add-in are shown. To show all the XBRL taxonomies available for download, select the check box *Show entry points available for download*. To show all versions, clear this check box. Entry points shown in red are not installed. To install the respective XBRL taxonomies, select the entry point and click **OK**. This opens the XBRL Taxonomy Manager where you can complete the installation. For more information, see XBRL Taxonomy Manager 30. Because of memory requirements, some entry points cannot be loaded in the 32-bit version of the add-in, in which case they appear as grayed out in the dialog box. To make the loading of such entry points possible, use Excel 64-bit and install the 64-bit version of the add-in.



Select Entry Point

Please select the XBRL taxonomy entry point to use for the new report:

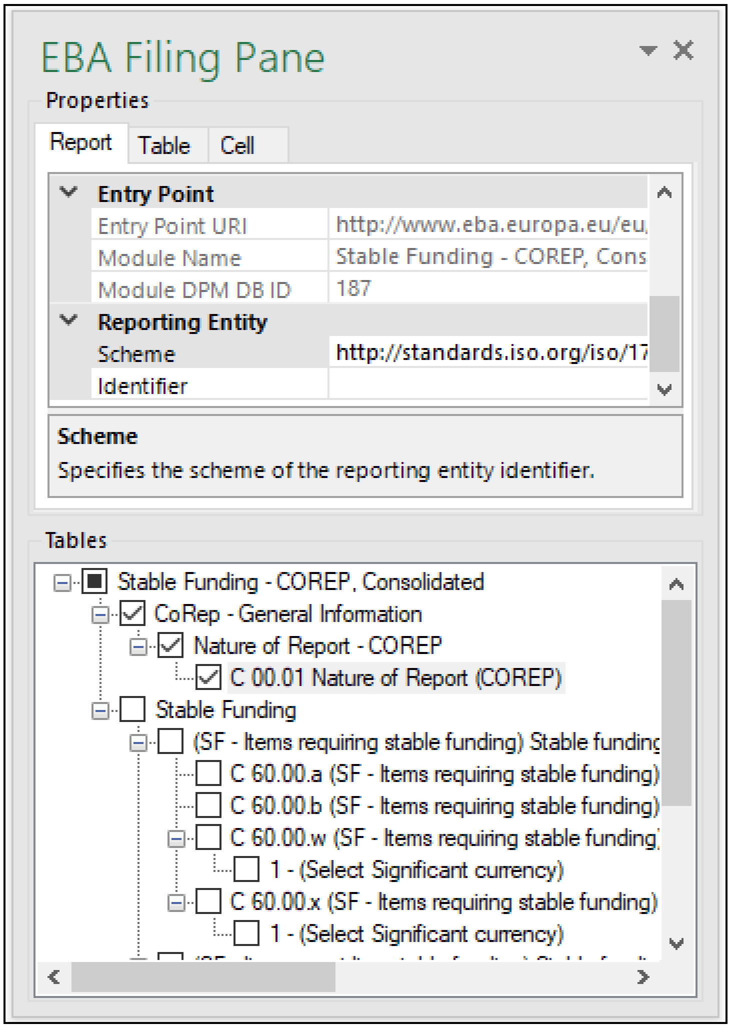
Filter:

- EBA 2.5 - Reporting framework provided by the European Banking Authority
  - AE: Asset Encumbrance
  - COREP_ALM: Common Reporting - Additional Liquidity Monitoring
  - COREP_Con: Cosolidated Common Reporting
  - COREP_Ind: Individual Common Reporting
  - COREP_LCR: Common Reporting - Liquidity Coverage
  - COREP_LE: Common Reporting - Large Exposures
  - COREP_NSFR: Common Reporting - Stable Funding
  - FINREP: Financial Reporting
  - FP: Funding Plans
  - SBP: Supervisory Benchmarking Portfolios
- ACPR 2.9 - Common Reporting with the modules LCR_CON, LCR_DA, NSFR_CON and ALM_CON
- ACPR 2.2 - Combating Money Laundering and the Financing of Terrorism
- BBK 2.10 - EBA-ITS-Reporting extended by the German base taxonomy
- BBK 2.9 - EBA-ITS-Reporting extended by the German base taxonomy
- BBK 2.8 - EBA-ITS-Reporting extended by the German base taxonomy
- BBK 2.6 - EBA-ITS-Reporting extended by the German base taxonomy
- BDP 2.10 - Financial and Common Reporting framework
- BOE 3.3 - Reporting for the banking sector
- BOE 1.1 - Statistical reporting obligations

☑ Show only most current entry points

☑ Show entry points available for download

4. At the next stage, the report tables are loaded into Excel. During this operation, a dialog box informs you about the progress. Once the report tables have finished loading, notice the *Tables* section in the EBA Filing Pane (*see below*).



Entry Point	
Entry Point URI	http://www.eba.europa.eu/eu,
Module Name	Stable Funding - COREP, Cons
Module DPM DB ID	187
Reporting Entity	
Scheme	http://standards.iso.org/iso/17
Identifier	

Scheme
Specifies the scheme of the reporting entity identifier.

- ■ Stable Funding - COREP, Consolidated
  - ☑ CoRep - General Information
    - ☑ Nature of Report - COREP
      - ☑ C 00.01 Nature of Report (COREP)
  - □ Stable Funding
    - □ (SF - Items requiring stable funding) Stable funding
      - □ C 60.00.a (SF - Items requiring stable funding)
      - □ C 60.00.b (SF - Items requiring stable funding)
      - □ C 60.00.w (SF - Items requiring stable funding)
        - □ 1 - (Select Significant currency)
      - □ C 60.00.x (SF - Items requiring stable funding)
        - □ 1 - (Select Significant currency)

5. Select the check boxes next to the tables you want to include in the report. Each included table appears on a new sheet in the Excel book. Please note that tables checked for inclusion will generate a true filing indicator, and tables that are left unchecked will generate a false filing indicator. You can set filing indicators by checking/unchecking the relevant table in the *Tables* section of the pane. Each table that has been checked will be added to the report and set as filed; all the others will not be added to the report.

You can now start entering data in tables, validate it, and export it to XBRL format. See the following topics for more information:

- Enter Data 17
- Control Accuracy of Cells 23
- Validate Data 24
- Export Data to XBRL 26

Update entry point (Enterprise Edition)

If you have already created a report and wish to switch to a different version (newer or older, if necessary) of the same taxonomy, you can do this by means of the **Update Entrypoint** ribbon command. Clicking this command opens the **Select Entry Point** dialog (*see subsection above*), in which you can select and install a different version of the same taxonomy. The currently used taxonomy version is grayed-out in the **Select Entry Point** dialog.

When you select a different taxonomy version you want to install, this taxonomy version will open in a new workbook. Importantly, the facts from the original workbook that fully correspond to the concepts in the new entry point will be copied to the new workbook. If, for some reasons, the facts are not available in the new entry point (e.g., the data types do not match in the original and newly installed entry points), you can copy these facts manually from the original workbook. Preserving the original workbook helps prevent data loss.

All the facts that cannot be inserted into the newly installed entry point will be shown in the Validation Report. This information will appear simultaneously in the validation reports of the original and newly installed entry points. The validation report will contain links to the facts that cannot be imported into the new report. Clicking these links will highlight the corresponding cells in the original report.

2.2 Filing Pane

The EBA Filing Pane enables you to include and exclude tables from the report, view information about each cell, and view and set various report properties. By default, this pane is visible. You can show or hide it by clicking the **Toggle EBA Filing Pane** command in the ribbon.The EBA Filing Pane has two main sections: *Properties* and *Tables* (*see subsections below*).

Properties
The properties displayed in the EBA Filing Pane directly affect the content of the XBRL instance file that will be created when you export the XBRL instance. To view what each property does, click it and observe the description displayed in the gray box under the grid. Grayed out properties are read-only. Otherwise, you can edit a property by typing text or selecting a value, as applicable.

The *Scheme* and *Identifier* properties under *Reporting Entity* are usually provided by the relevant competent authorities.

Even though some property values begin with **`http`** (e.g., *XSD Entry Point URI*), they do not necessarily point to live web resources and thus should not be considered dead links. To resolve entry point URIs, the add-in uses a catalog mechanism that maps URIs to files on the local system. This is largely due to the size of the taxonomies and the fact that they contain thousands of files. Accessing the taxonomy files on the Internet would result in extremely slow performance even if their issuing organizations served them that way.

Properties are grouped into the following three tabs:

- The Report tab displays properties applicable to the entire report: One report corresponds to one Excel workbook.
- The Table tab displays properties of the currently selected table. A table normally corresponds to a single Excel worksheet. Therefore, whenever you click inside a new Excel sheet, the properties are re-drawn to display the new worksheet.
- The Cell tab displays properties of the currently selected cell. Whenever you click a new cell, the cell properties are re-drawn accordingly.

You can set the accuracy-related properties at report, table, or cell level. For more information, see Control Accuracy of Cells 23.

Tables
To include a table in the report, select its corresponding check box in the pane. Each included table appears on a new sheet in the Excel book. To go to a specific sheet, navigate to it using the standard Excel way or click the corresponding table in the EBA Filing Pane. To remove a particular table from the report, clear the check box next to it. Tables that are not selected will not be included in the report.

Some tables support a z-axis (a third dimension). For information about adding a z-axis to a table, see Enter Data into 3D Tables 21.

Each report table displayed in the EBA Filing Pane is XBRL-bound, which means that data you enter directly in the table cells will be reflected in the XBRL instance file when the report is ready. For more information, see Export Data to XBRL 26. While the report data is work in progress, you can save the Excel workbook and reopen it later.

Any sheets that contain tables are bound to the XBRL taxonomy. Therefore, these tables must not be deleted. It is not recommended to rename such sheets. If necessary, you can add new sheets to the workbook.

However, such sheets will not be bound to the XBRL taxonomy. As a result, these sheets will be ignored when you generate the XBRL instance file.

2.3 Enter Data

You can fill a report with data by entering data into cells manually or by pasting values. With some cells, you can select a value from a predefined list, such as countries or currencies. In some report tables, you may need to add new rows or columns. Below you will find information about how to enter data.

Editable vs. non-editable cells

As a general rule, gray cells must not be edited. Only cells that are included in the XBRL-bound area (delimited by the table boundaries) are to be edited. For information about various cells and data to be entered, consult the cell information (properties) displayed in the Cell tab of the EBA Filing Pane.

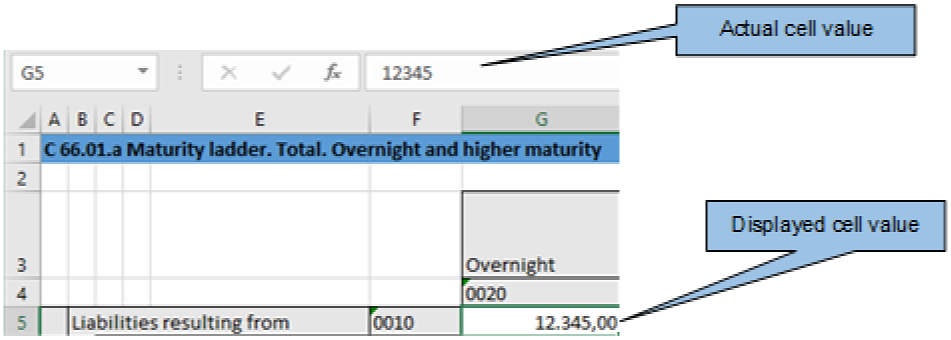
Paste data

If you paste data from multiple columns, the number of pasted columns should correspond to the number of columns in the predefined sheet. If you accidentally paste a larger number of columns or type text outside the default table, unwanted columns may appear outside the XBRL-bound area. To delete the unwanted columns, right-click the cell and select **Delete | Table Columns**. To prevent Excel from adding new columns and rows automatically, go to **File | Options | Proofing | AutoCorrect Options | AutoFormat As You Type | Apply as you work** and clear the *Include new rows and columns in table* check box.

When you paste data, it is recommended to keep only the values but not the formatting. To do this, select the **Paste Values** option when you paste cells or rows.

Actual vs. displayed cell value

While generating the XBRL instance file, the add-in ignores any cell formatting information and exports the *actual* value of the Excel cell. However, bear in mind that the actual value may be different from the value displayed in the cell because of the cell formatting information. You can see the actual value that will be written to the XBRL instance in the formula bar of Excel (*see example below*).



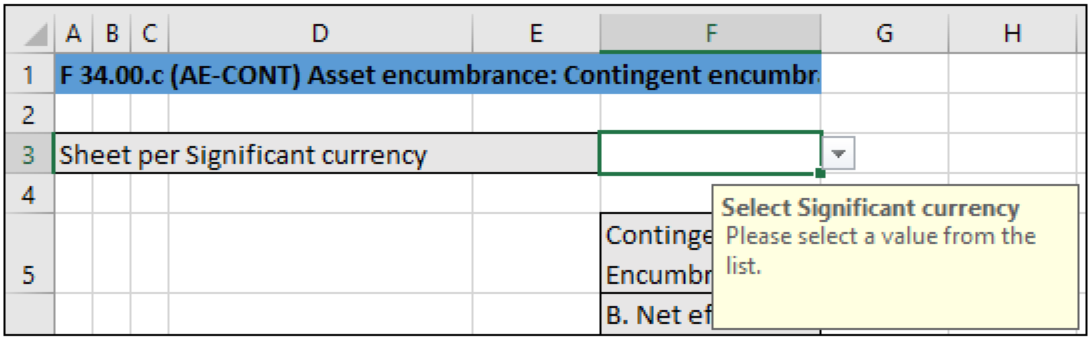
G5 12345

	A	B	C	D	E	F	G
1	C 66.01.a Maturity ladder. Total. Overnight and higher maturity						
2							
3							Overnight
4							0020
5		Liabilities resulting from				0010	12.345,00

In the example above, the value that will be written to the XBRL instance is `12345`. Note that the number accuracy reported in the XBRL instance file also depends on the value you selected for the *Accuracy* properties. For more information about the accuracy of cells, see Control Accuracy of Cells 23.

Enumeration values

Some cells expect a fixed predetermined value (e.g., cells that represent currencies or countries). In this case, the add-in displays a small tooltip when you click the cell. You can choose the required value from the drop-down list:



F 34.00.c (AE-CONT) Asset encumbrance: Contingent encumbr

Sheet per Significant currency

Select Significant currency
Please select a value from the list.

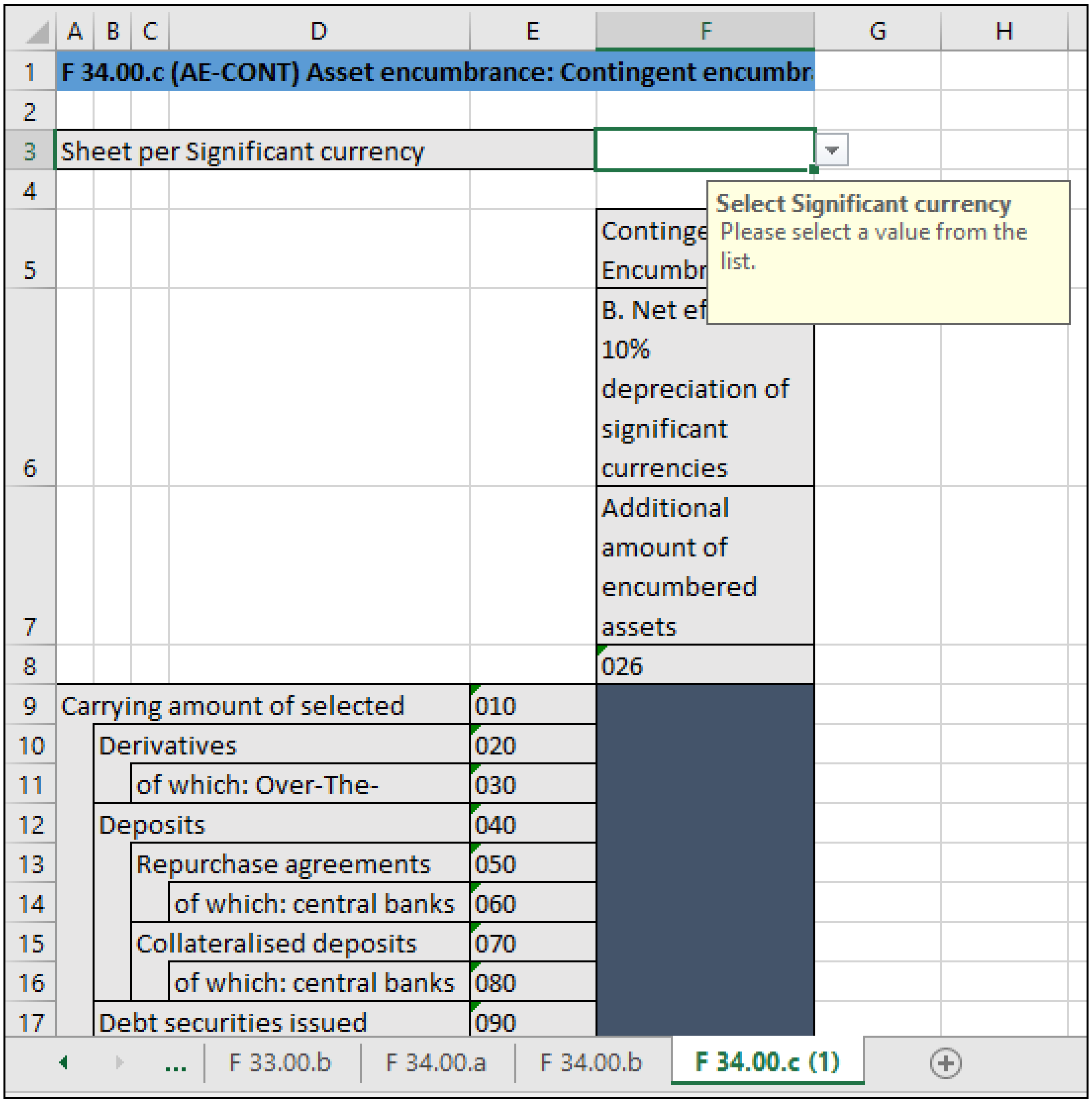
Continge
Encumbr

B. Net ef

To view the full list of all possible values, click the cell and see the cell properties in the Cell tab of the EBA Filing Pane.

Conditional cells

In some tables, you must first fill out a cell value in order to make other cells of the table editable. For example, cell F3 must be filled in before all other cells in the same column can be edited (*see screenshot below*).



F 34.00.c (AE-CONT) Asset encumbrance: Contingent encumbr

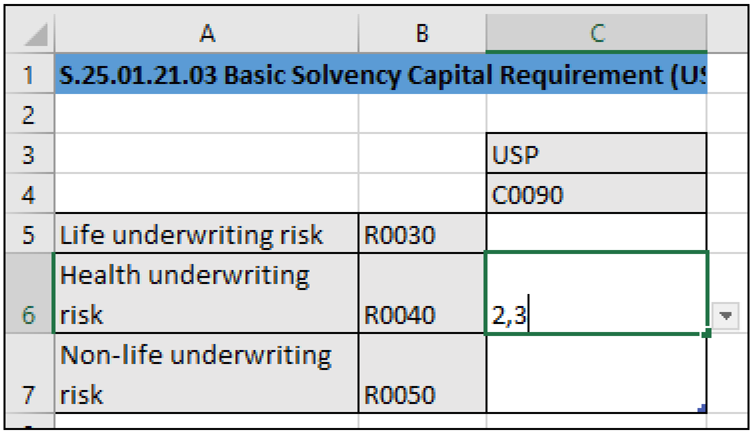
Sheet per Significant currency

Select Significant currency
Please select a value from the list.

		Continge Encumbr
		B. Net ef 10% depreciation of significant currencies
		Additional amount of encumbered assets
		026
Carrying amount of selected	010	
Derivatives	020	
of which: Over-The-	030	
Deposits	040	
Repurchase agreements	050	
of which: central banks	060	
Collateralised deposits	070	
of which: central banks	080	
Debt securities issued	090	

Cells with multiple values

Depending on the XBRL taxonomy, some reports might have facts that represent a list of comma-separated multiple values. As a result, the corresponding cell also requires multiple values to be entered in the same cell. To enter data for such cells, expand the drop-down list and click all the relevant items. Alternatively, you can type all the numeric values, separated by a comma (*see screenshot below*). Remember that you can view all possible values of a cell in the Cell tab of the EBA Filing Pane.



	A	B	C
1	S.25.01.21.03 Basic Solvency Capital Requirement (U		
2			
3			USP
4			C0090
5	Life underwriting risk	R0030	
6	Health underwriting risk	R0040	2,3
7	Non-life underwriting risk	R0050	

After you exit the cell with multiple values, this cell is automatically re-drawn to display all selected values in a readable form (even though you may have entered only numbers).

New rows

With some tables, you may need to create new rows. For example, this is the case for *Table C 10.02* available through the entry point *EBA 2.6 COREP CON*. You can add new rows using the standard Excel commands or shortcuts. Alternatively, you can click the **Add Row** button in the ribbon. For example, to add a new row to *Table C 10.02* of the entry point mentioned above, do one of the following:

- Open the add-in tab in the Excel ribbon and click **Add Row | Insert Row Below**. Note that the commands to insert or delete rows are enabled only if adding new rows is supported for this table.
- Click the rightmost cell of the last row in the table and press **Tab**.
- Right-click a cell in the empty row and select **Insert | Table Row Below** from the context menu.

Note: Any newly added rows must be within the XBRL-bound area of the table, clearly delimited by black lines.

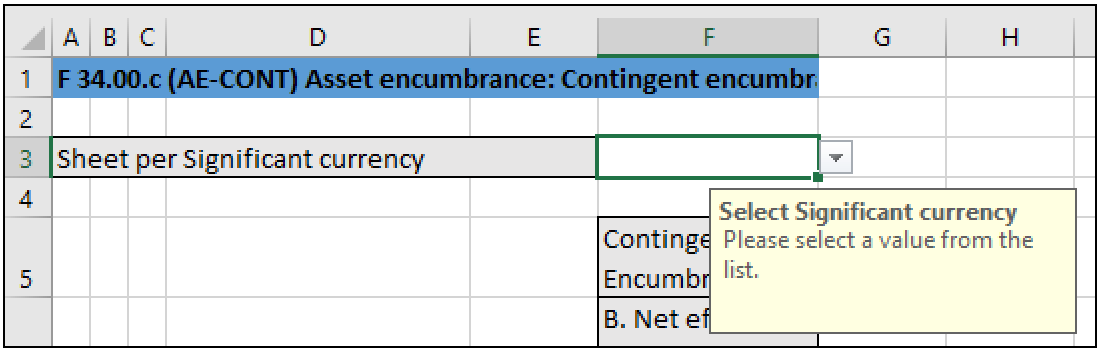
New columns

For some tables, you may need to add extra columns, which means these tables can grow horizontally. You can add new columns to such tables in one of the following ways:

- Open the add-in tab in the Excel ribbon and click **Add Column**. Note that this command is enabled only if adding new columns is supported for this table.
- Click the **Add** button that appears next to the rightmost column of a table.
- Right-click a table cell and select **Insert | Table Columns to the Right** from the context menu.

2.4 Enter Data into 3D Tables

Most of the report tables have only two dimensions: the x-axis (columns) and the y-axis (rows). However, there are some tables where you may need to enter data into a third dimension (the z-axis): e.g., *Table F 34.00.c (AE CON)* available through the entry point *AE CON: Asset Encumbrance, Consolidated*. This table may need an additional sheet for each currency. As shown below, cell F3 is a drop-down list from which you can select a currency.

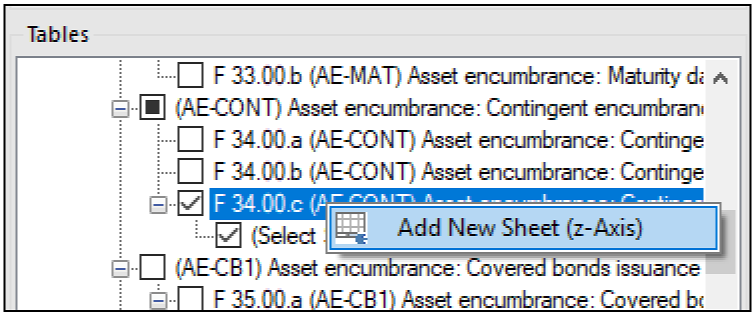


	A	B	C	D	E	F	G	H
1	F 34.00.c (AE-CONT) Asset encumbrance: Contingent encumbr							
2								
3	Sheet per Significant currency							
4								
5						Continge Encumbr		
						B. Net ef		

Select Significant currency
Please select a value from the list.

In cases such as the one above, you can add a new sheet along the z-axis (third dimension) of the table as follows:

1. Click the add-in tab in the Excel ribbon.
2. Click the **Add Sheet (z-Axis)** button. Note that the commands to insert or delete new z-axis sheets are enabled only if adding the z-axis is supported for this table by the taxonomy. Alternatively, right-click the table in the EBA Filing Pane and select **Add New Sheet (z-Axis)** from the context menu (*see screenshot below*). This creates a new sheet that displays the third dimension of the table (z-axis). The sheet representing the z-axis always has an indicative name that resembles the original table.

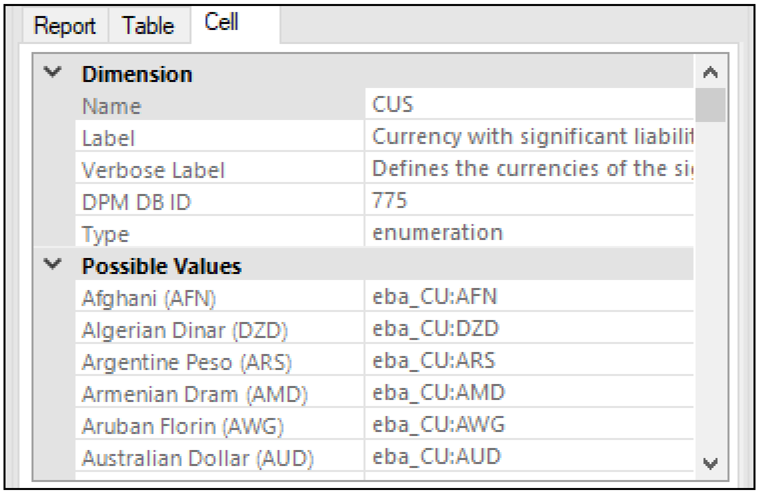


Tables

- □ F 33.00.b (AE-MAT) Asset encumbrance: Maturity da
- ■ (AE-CONT) Asset encumbrance: Contingent encumbran
  - □ F 34.00.a (AE-CONT) Asset encumbrance: Continge
  - □ F 34.00.b (AE-CONT) Asset encumbrance: Continge
  - ☑ F 34.00.c (AE-CONT) Asset encumbrance: Continge
    - ☑ (Select
- □ (AE-CB1) Asset encumbrance: Covered bonds issuance
  - □ F 35.00.a (AE-CB1) Asset encumbrance: Covered bo

Data from the third dimension (z-axis) of a table is displayed as new sheets in Excel. Therefore, three-dimensional tables span across more than one sheet. This is an exception to the rule that one Excel sheet corresponds to one table in an XBRL report. In the XBRL instance, data that belongs to the z-axis will be correctly reported as part of the same table.

When you click a cell that represents the z-axis, all the possible values for the drop-down list are displayed in the Cell tab of the EBA Filing Pane (*see screenshot below*).

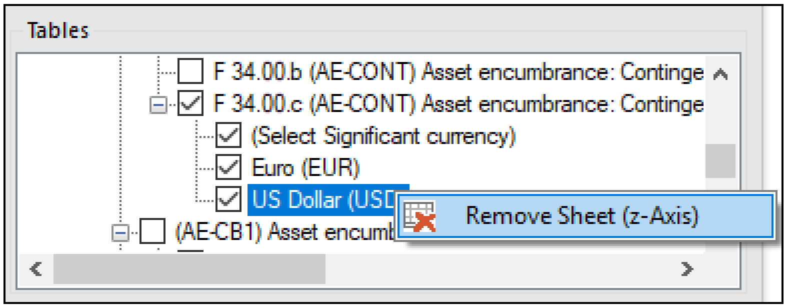


Dimension	
Name	CUS
Label	Currency with significant liabilit
Verbose Label	Defines the currencies of the si
DPM DB ID	775
Type	enumeration
Possible Values	
Afghani (AFN)	eba_CU:AFN
Algerian Dinar (DZD)	eba_CU:DZD
Argentine Peso (ARS)	eba_CU:ARS
Armenian Dram (AMD)	eba_CU:AMD
Aruban Florin (AWG)	eba_CU:AWG
Australian Dollar (AUD)	eba_CU:AUD

Delete z-Axis sheets

You can delete sheets that contain data from the third dimension (z-axis) as follows:

1. Select the relevant sheet or click the corresponding entry in the *Tables* section of the EBA Filing Pane.
2. Click **Delete Sheet (z-Axis)** in the EBA tab. Alternatively, right-click the relevant table in the EBA Filing Pane and select **Delete Sheet (z-Axis)** from the context menu (*see screenshot below*).



Tables

- □ F 34.00.b (AE-CONT) Asset encumbrance: Continge
- ☑ F 34.00.c (AE-CONT) Asset encumbrance: Continge
  - ☑ (Select Significant currency)
  - ☑ Euro (EUR)
  - ☑ US Dollar (USD
- □ (AE-CB1) Asset encumb

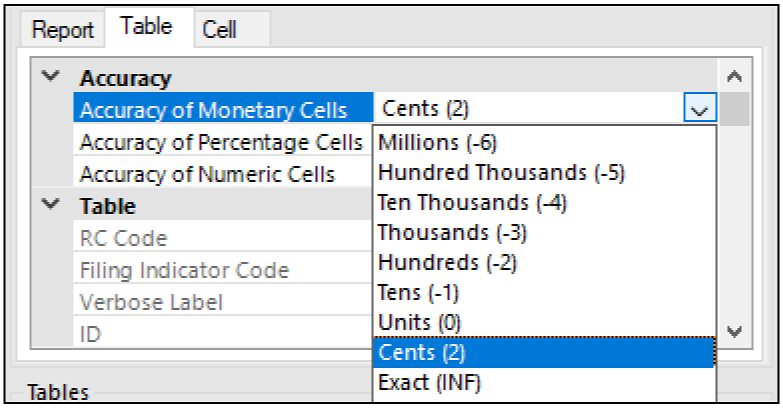
2.5 Control Accuracy of Cells

You can control the accuracy of monetary and other numeric values in the XBRL report by setting the accuracy of monetary cells, numeric cells, and percentage cells. These properties are available in the EBA Filing Pane in the *Properties* section.

The accuracy parameter communicates the level of precision of reported facts and indicates the number of decimal places to which the reported fact is accurate. The level of accuracy is communicated through the `decimals` attribute. The `decimals` attribute can be an integer or an INF value. The INF value represents the exact value of a reported fact. In the screenshot below, the accuracy value indicated in the parentheses corresponds to the value of the `decimals` attribute in the XBRL instance file.

For monetary and numeric cells, the `decimals` value can be positive or negative. A positive value *N* specifies the accuracy of up to *N* digits to the right of the decimal separator. For example, the value 2 specifies the accuracy of monetary cells to be in cents. A negative value *N* specifies the accuracy of up to *N* digits to the left of the decimal separator. For example, the value -3 specifies the accuracy to be up to thousands, while the value -6 specifies the accuracy to be up to millions. To find out more about accuracy, see the [XBRL 2.1 Recommendation].

You can set the accuracy-related properties at report, table, or cell level (*see screenshot below*). If you set accuracy at multiple levels, the more specific property always overrides the more general one. For example, the accuracy set at cell level takes priority over the accuracy set at table level.



Accuracy of Monetary Cells: Cents (2)
Accuracy of Percentage Cells
Accuracy of Numeric Cells

Accuracy of monetary cells
The property *Accuracy of Monetary Cells* applies to numeric cells that represent a monetary value. The option you choose specifies the accuracy of the number, relative to the decimal point. By default, this property is set to *Cents (2)*, which means that the number is accurate up to 2 places to the right of the decimal point.

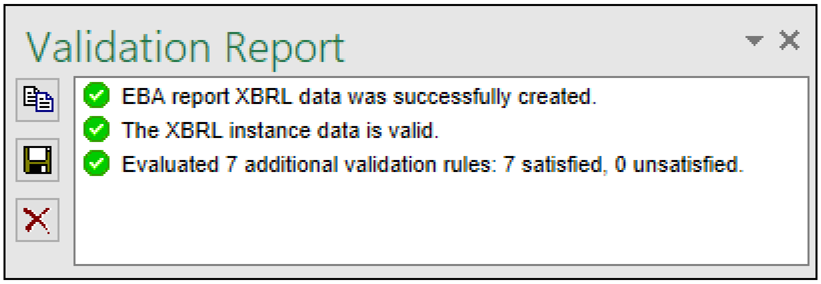
Accuracy of percentage cells
The property *Accuracy of Percentage Cells* applies to values that represent a percentage. The option you choose specifies the accuracy of the percentage number, relative to the decimal point. By default, this property is set to *Basis Points (4)*, which means the percentage number is accurate up to 4 places to the right of the decimal point.

Accuracy of numeric cells
The property *Accuracy of Numeric Cells* applies to numeric values that are neither monetary nor percentage values. The option you choose specifies the accuracy of the number, relative to the decimal point. By default, this property is set to *Exact (INF)*.

2.6 Validate Data

Validation ensures that the XBRL data you are filing conforms to the XBRL specification. It is essential that you validate your report data before exporting it. To validate data, click the **Validate** button in the add-in tab of the Excel ribbon. When the validation of the instance finishes, a validation report similar to the one below is displayed.

When the validation fails, the Validation Report window may display links to the cell where the error occurred. To quickly find a cell with the error, click the underlined text, and the cursor will be positioned automatically on the required cell. Note that there are cases where multiple cells are involved in a single validation check. In such cases, clicking on the error link will select one of the affected cells.



✓ EBA report XBRL data was successfully created.
✓ The XBRL instance data is valid.
✓ Evaluated 7 additional validation rules: 7 satisfied, 0 unsatisfied.

Validation results

The validation result can contain any of the following validation messages:

Message type	Meaning
✓	The instance data is valid.
⚠	The instance data is valid but has inconsistencies or warnings.
⊗	The instance data is not valid.

Information messages, warnings, and errors

The **Validation Report** dialog box may additionally display any of the following message types: information messages, warnings, and errors.

Message type	Meaning
ℹ	This is an information message. Information messages do not make the XBRL instance invalid.
⚠	This is a warning message or an inconsistency. Warnings and inconsistencies do not make the XBRL instance invalid.
❗	Indicates an error. If there are validation errors, the XBRL instance is not valid. In this case, you will need to edit the report data to correct each error before exporting to XBRL.

Message type	Meaning
	During validation, the add-in checks XBRL formula assertions and reports them as errors. If you are using Altova RaptorXML+XBRL Server for validation, XBRL formula assertions may be optionally configured not to be reported as errors.

Note: By default, the add-in treats invalid cell values as errors. If necessary, you can configure the add-in to treat invalid cell values as warnings instead. For more information, see Settings 51.

Copy, save, clear

To copy the contents of the validation report to the clipboard, click 📋 and paste it into a target file (e.g., an email). Alternatively, right-click inside the Validation Report window and select **Copy All Messages** from the context menu.

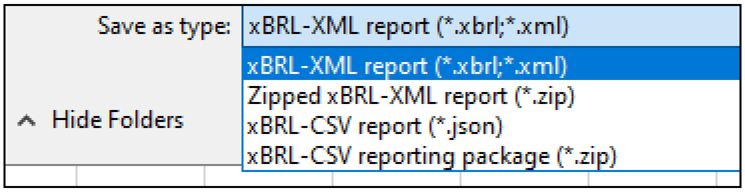
To save the validation report as text or HTML, click 💾. Alternatively, right-click inside the Validation Report window and select **Save Validation Report** from the context menu.

To clear the validation report, click ✕. Alternatively, right-click inside the Validation Report window and select **Clear** from the context menu.

2.7 Export Data to XBRL

Once your report is ready and valid, you can generate an XBRL instance file. To do this, click **Export to XBRL** in the add-in ribbon. By default, instance files are saved with a `.xbrl` extension. If you need the exported file to have another extension (for example, `.xml`), type the file extension in the **Export** dialog box.

You can export your file in the formats shown in the screenshot below:



Save as type: xBRL-XML report (*.xbrl;*.xml)

- xBRL-XML report (*.xbrl;*.xml)
- Zipped xBRL-XML report (*.zip)
- xBRL-CSV report (*.json)
- xBRL-CSV reporting package (*.zip)

During the export, data is automatically validated. Any errors, inconsistencies and warnings are reported on the screen after the export finishes. For more information, see Validate Data[24].

Note: Cell values that are not valid, that is, cells that do not conform to the data type of the underlying XBRL concept, prevent the report from being exported.

For tips on how to avoid data formatting errors, see Enter Data[17]. Note, however, that not all XBRL validation errors might be related to incorrect formatting. Some errors might occur because entered data does not meet the XBRL validation rules applicable to the report you are filing.

2.8 Import Data from XBRL

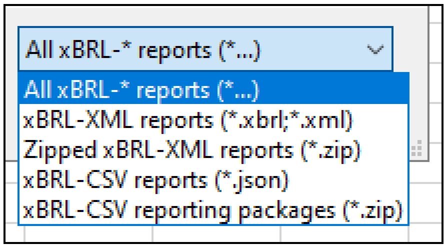
You can import data from existing instances of XBRL reports into Excel (typically, files with a **`.xbrl`** extension). For the import to be successful, the imported instances must be valid XBRL reports. They may be reports you have previously generated using the **EBA XBRL Add-in** or reports that you received from other parties.

To import an existing XBRL instance file into Excel, follow the instructions below:

1. In the Excel ribbon, click the add-in tab.
2. Click **Import from XBRL** and browse for the XBRL instance file.

Note: If a report is already open in Excel, the **Import from XBRL** button is disabled. To enable the command, save and close the current report (workbook) and create a new workbook.

The formats that can be imported are illustrated in the screenshot below:



All xBRL-* reports (*...)

All xBRL-* reports (*...)
xBRL-XML reports (*.xbrl;*.xml)
Zipped xBRL-XML reports (*.zip)
xBRL-CSV reports (*.json)
xBRL-CSV reporting packages (*.zip)

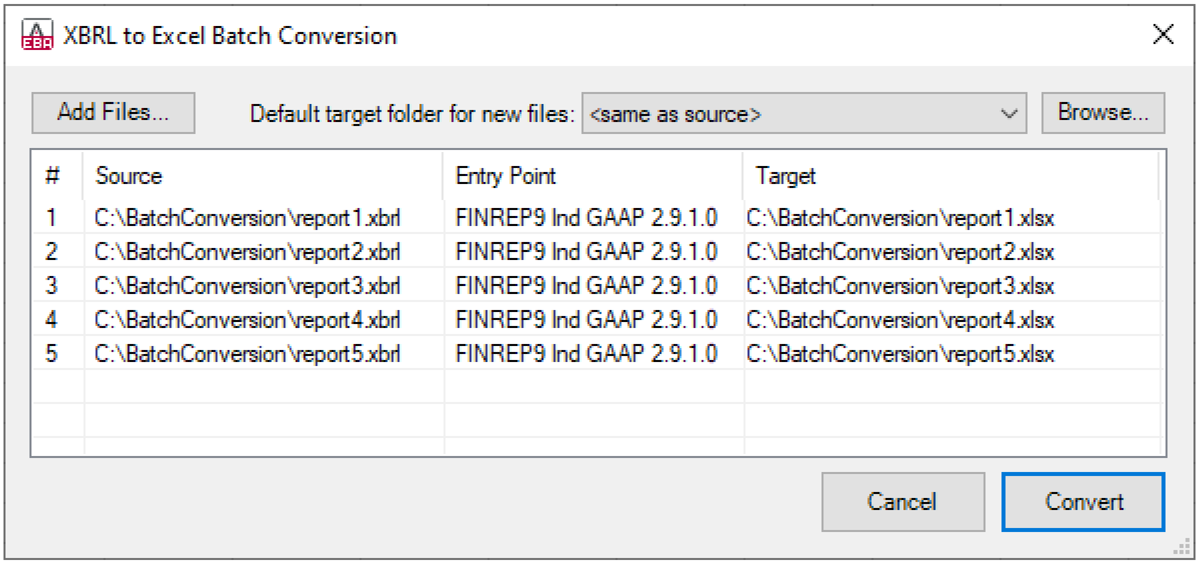
During the import operation, the **Importing XBRL report** dialog box informs you about the progress. While the report data is loaded into Excel, it is automatically validated. The dialog box notifies you about potential warnings, inconsistencies, and/or errors. For more information, see Validate Data 24.

Note: During the import, the add-in validates XBRL formula assertions. The report will be imported even if it contains unsatisfied assertions.

2.9 Batch Convert XBRL to Excel

The **Batch Conversion** command in the Excel ribbon enables you to convert multiple XBRL instance files to Excel format. The result would be the same as if you imported[27] multiple XBRL instance files and then saved them to Excel format. The main advantage is that the conversion is processed as a batch. To perform a batch conversion, add all the required files to a batch as follows:

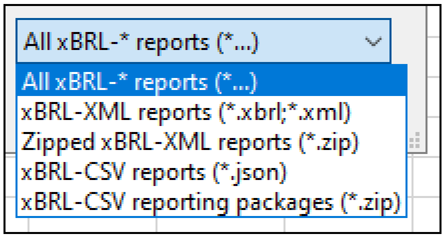
1. Click the add-in tab in the Excel ribbon.
2. Click the **Batch Conversion** button.
3. Click **Add Files**. Alternatively, right-click the grid and select **Add Files** from the context menu.



Default target folder for new files: <same as source>

#	Source	Entry Point	Target
1	C:\BatchConversion\report1.xbrl	FINREP9 Ind GAAP 2.9.1.0	C:\BatchConversion\report1.xlsx
2	C:\BatchConversion\report2.xbrl	FINREP9 Ind GAAP 2.9.1.0	C:\BatchConversion\report2.xlsx
3	C:\BatchConversion\report3.xbrl	FINREP9 Ind GAAP 2.9.1.0	C:\BatchConversion\report3.xlsx
4	C:\BatchConversion\report4.xbrl	FINREP9 Ind GAAP 2.9.1.0	C:\BatchConversion\report4.xlsx
5	C:\BatchConversion\report5.xbrl	FINREP9 Ind GAAP 2.9.1.0	C:\BatchConversion\report5.xlsx

The formats available for batch conversion are shown in the screenshot below:


All xBRL-* reports (*...)
All xBRL-* reports (*...)
xBRL-XML reports (*.xbrl;*.xml)
Zipped xBRL-XML reports (*.zip)
xBRL-CSV reports (*.json)
xBRL-CSV reporting packages (*.zip)

Additional options in the Batch Conversion dialog box

The conversion dialog box allows you to perform the following additional tasks:

1. To add files from additional source locations to the same batch, click **Add Files**.
2. Whenever you add new files to the batch, their default target folder is the same as the source folder. If you want to assign a specific target folder to all new files by default, select the relevant folder from the list called *Default target folder for new files*. To add new entries to the list, click **Browse** and choose a folder. By default, the *Default target folder for new files* option affects new files that you add to the batch. However, if you change this option, and if the files already exist in the batch (in the grid), you

will be informed that the change will affect all the new files added to the batch. Click **Yes** if the target folder of existing files should also be changed.

3. You can choose to save all converted files to the same target folder or set a different target folder for each file. To change the target folder of specific files, select the relevant files in the grid, right-click the grid, and select **Set target folder** from the context menu.
4. You can rename the target files. First, select the files in the grid. Then right-click the grid and select **Rename** from the context menu (or press **F2**). You can change both the file name and the file path. However, if you change the path, make sure that it exists.
5. To remove files from the batch, select them, right-click the grid, and select **Remove** from the context menu (or press the **Del** key).

Select multiple files

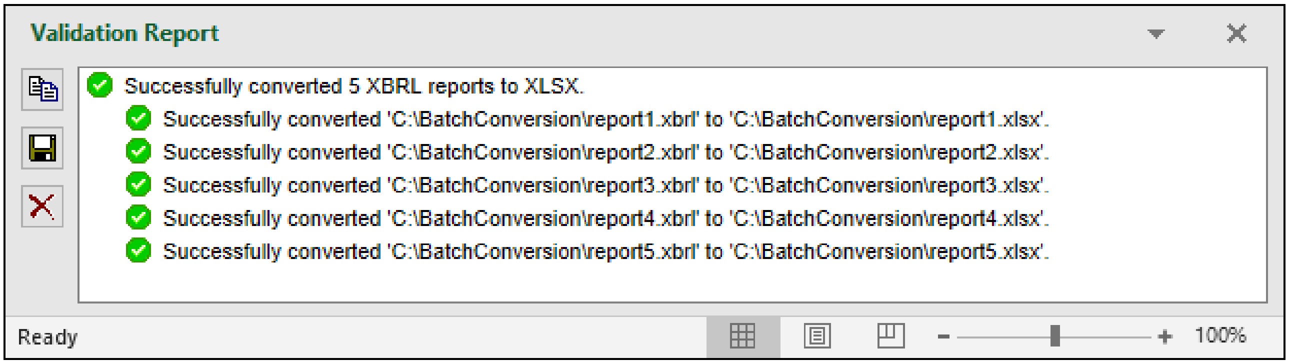
To select multiple files from the grid, use the standard Windows key combinations, for example:

- While holding the **Ctrl** key pressed, click to select the files of interest.
- Click an empty area in the dialog box and drag the cursor over the files that you want to select (rectangular selection).
- Press **Ctrl+A** to select all the files in the grid.

Once the batch is ready, click **Convert** to process all the files in it.

Validation report

The outcome of the conversion is reported in the Validation Report window (*see screenshot below*). For more information about validation messages, see Validate Data 24.

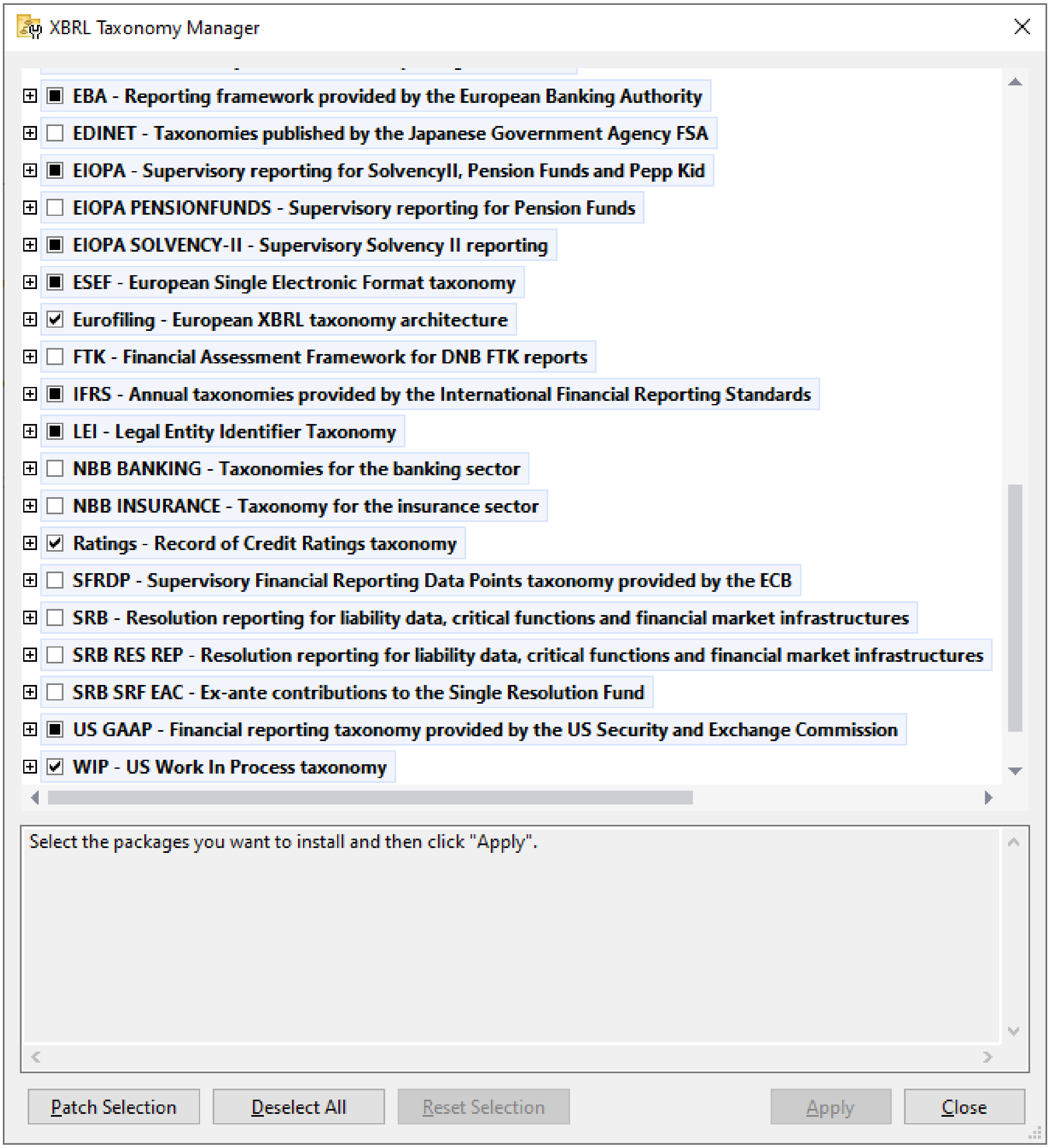


Successfully converted 5 XBRL reports to XLSX.
Successfully converted 'C:\BatchConversion\report1.xbrl' to 'C:\BatchConversion\report1.xlsx'.
Successfully converted 'C:\BatchConversion\report2.xbrl' to 'C:\BatchConversion\report2.xlsx'.
Successfully converted 'C:\BatchConversion\report3.xbrl' to 'C:\BatchConversion\report3.xlsx'.
Successfully converted 'C:\BatchConversion\report4.xbrl' to 'C:\BatchConversion\report4.xlsx'.
Successfully converted 'C:\BatchConversion\report5.xbrl' to 'C:\BatchConversion\report5.xlsx'.

3 Taxonomy Manager

XBRL Taxonomy Manager is an Altova tool that provides a centralized way to install and manage XBRL taxonomies for use across all Altova's XBRL-enabled applications, including EBA XBRL Add-in.

- On Windows, Taxonomy Manager has a graphical user interface (*screenshot below*) and is also available at the command line. (Altova's desktop applications are available on Windows only; *see list below*.)
- On Linux and macOS, Taxonomy Manager is available at the command line only. (Altova's server applications are available on Windows, Linux, and macOS; *see list below*.)



- ⊞ ■ EBA - Reporting framework provided by the European Banking Authority
- ⊞ □ EDINET - Taxonomies published by the Japanese Government Agency FSA
- ⊞ ■ EIOPA - Supervisory reporting for SolvencyII, Pension Funds and Pepp Kid
- ⊞ □ EIOPA PENSIONFUNDS - Supervisory reporting for Pension Funds
- ⊞ ■ EIOPA SOLVENCY-II - Supervisory Solvency II reporting
- ⊞ ■ ESEF - European Single Electronic Format taxonomy
- ⊞ ☑ Eurofiling - European XBRL taxonomy architecture
- ⊞ □ FTK - Financial Assessment Framework for DNB FTK reports
- ⊞ ■ IFRS - Annual taxonomies provided by the International Financial Reporting Standards
- ⊞ ■ LEI - Legal Entity Identifier Taxonomy
- ⊞ □ NBB BANKING - Taxonomies for the banking sector
- ⊞ □ NBB INSURANCE - Taxonomy for the insurance sector
- ⊞ ☑ Ratings - Record of Credit Ratings taxonomy
- ⊞ □ SFRDP - Supervisory Financial Reporting Data Points taxonomy provided by the ECB
- ⊞ □ SRB - Resolution reporting for liability data, critical functions and financial market infrastructures
- ⊞ □ SRB RES REP - Resolution reporting for liability data, critical functions and financial market infrastructures
- ⊞ □ SRB SRF EAC - Ex-ante contributions to the Single Resolution Fund
- ⊞ ■ US GAAP - Financial reporting taxonomy provided by the US Security and Exchange Commission
- ⊞ ☑ WIP - US Work In Process taxonomy

Select the packages you want to install and then click "Apply".

Altova's XBRL-enabled applications

Desktop applications (Windows only)	Server applications (Windows, Linux, macOS)
Altova XBRL Add-ins for Excel (EBA, ESEF, Solvency II, WIP)	MapForce Server (Standard and Advanced Editions)

MapForce Enterprise Edition	RaptorXML+XBRL Server
StyleVision Enterprise Edition	StyleVision Server
XMLSpy Enterprise Edition	

Installation and de-installation of Taxonomy Manager

Taxonomy Manager is installed automatically when you first install a new version of Altova Mission Kit Enterprise Edition or of any of Altova's XBRL-enabled applications (*see table above*).

Likewise, it is removed automatically when you uninstall the last Altova XBRL-enabled application from your computer.

Taxonomy Manager features

Taxonomy Manager provides the following features:

- Shows XBRL taxonomies installed on your computer and checks whether new versions are available for download.
- Downloads newer versions of XBRL taxonomies independently of the Altova product release cycle. (Altova stores taxonomies online, and you can download them via Taxonomy Manager.)
- Install or uninstall any of the multiple versions of a given taxonomy (or all versions if necessary).
- An XBRL taxonomy may have dependencies on other taxonomies. When you install or uninstall a particular taxonomy, Taxonomy Manager informs you about dependent taxonomies and will automatically install or remove them as well.
- Taxonomy Manager uses the XML catalog mechanism to map schema references to local files. In the case of large XBRL taxonomies, processing will therefore be faster than if the taxonomies were at a remote location.
- All major taxonomies are available via Taxonomy Manager and are regularly updated for the latest versions. This provides you with a convenient single resource for managing all your taxonomies and making them readily available to all of Altova's XBRL-enabled applications.
- Changes made in Taxonomy Manager take effect for all Altova products installed on that machine.

How it works

Altova stores all XBRL taxonomies used in Altova products online. This repository is updated when new versions of the taxonomies are released. Taxonomy Manager displays information about the latest available taxonomies when invoked in both its GUI form as well as on the CLI. You can then install, upgrade or uninstall taxonomies via Taxonomy Manager.

Taxonomy Manager also installs taxonomies in one other way. At the Altova website (https://www.altova.com/taxonomy-manager) you can select a taxonomy and its dependent taxonomies that you want to install. The website will prepare a file of type **`.altova_taxonomies`** for download that contains information about your taxonomy selection. When you double-click this file or pass it to Taxonomy Manager via the CLI as an argument of the **`install`** 44 command, Taxonomy Manager will install the taxonomies you selected.

Local cache: tracking your taxonomies

All information about installed taxonomies is tracked in a centralized cache directory on your computer, located here:

Windows	`C:\ProgramData\Altova\pkgs\.cache` or `<USER-HOME>\Documents\Altova\pkgs\.cache` (the path depends on the directory you have selected for the *Taxonomies Folder* field in the **Settings** dialog)
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

This cache directory is updated regularly with the latest status of taxonomies at Altova's online storage. These updates are carried out at the following times:

- Every time you start Taxonomy Manager.
- When you start EBA XBRL Add-in for the first time on a given calendar day.
- If EBA XBRL Add-in is open for more than 24 hours, the cache is updated every 24 hours.
- You can also update the cache by running the `update` 47 command at the command line interface.

The cache therefore enables Taxonomy Manager to continuously track your installed taxonomies against the taxonomies available online at the Altova website.

Do not modify the cache manually!

The local cache directory is maintained automatically based on the taxonomies you install and uninstall. It should not be altered or deleted manually. If you ever need to reset Taxonomy Manager to its original "pristine" state, then, on the command line interface (CLI): (i) run the `reset` 45 command, and (ii) run the `initialize` 43 command. (Alternatively, run the **`reset`** command with the **`--i`** option.)

HTTP proxy

You can use an HTTP proxy for Taxonomy Manager connections. The Windows system's proxy settings will be used.

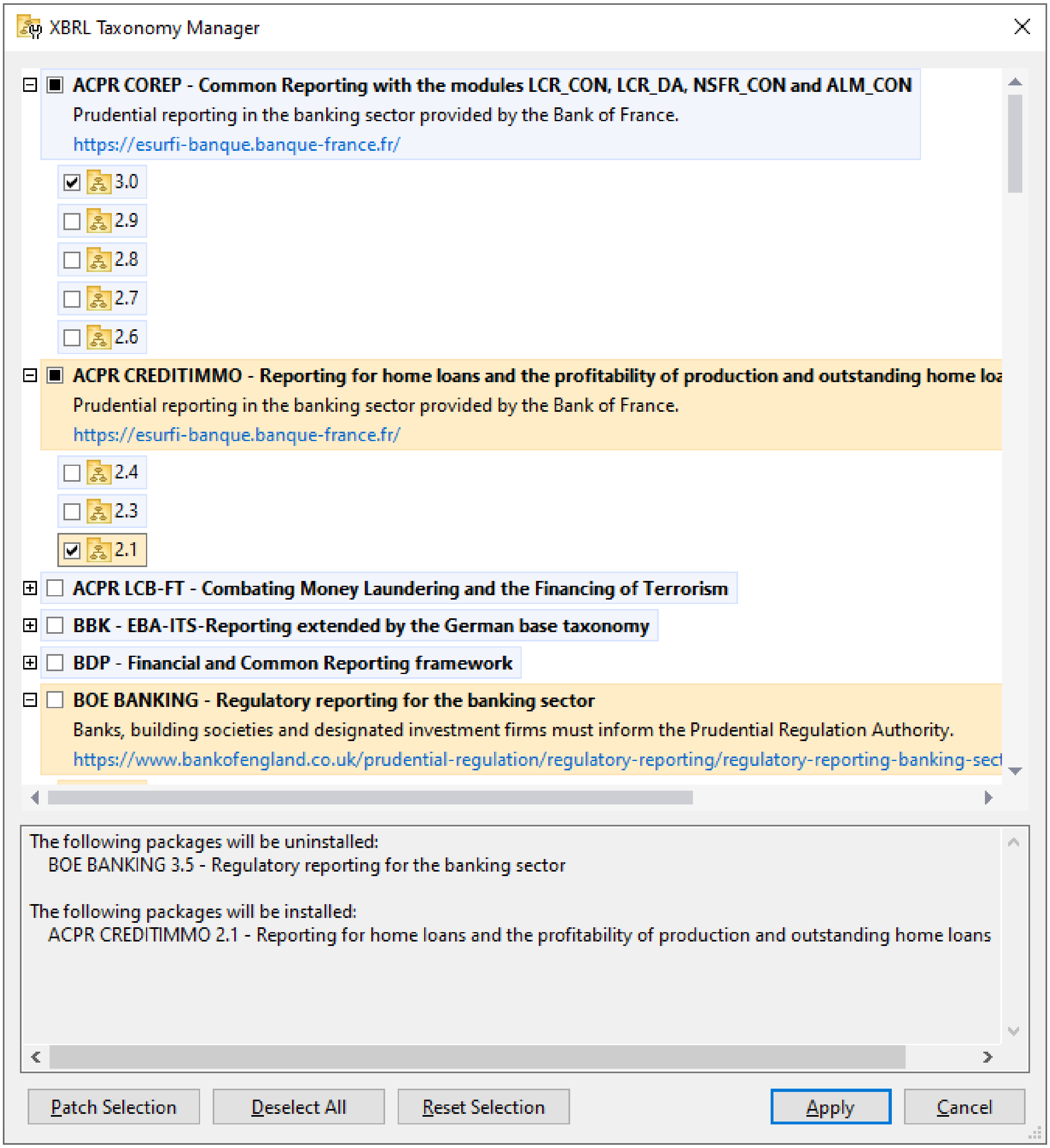
3.1 Run Taxonomy Manager

Graphical User Interface

You can access the GUI of Taxonomy Manager in any of the following ways:

- *During the installation of EBA XBRL Add-in:* Towards the end of the installation procedure, select the check box *Invoke Altova Taxonomy Manager* to access the XBRL Taxonomy Manager GUI straight away. This will enable you to install taxonomies during the installation process of your Altova application.
- *After the installation of EBA XBRL Add-in:* After your application has been installed, you can access the Taxonomy Manager GUI at any time, by clicking **Manage Taxonomies** in the add-in ribbon.
- Via the **`.altova_taxonomies`** file downloaded from the Altova's XBRL Taxonomy Download Center: Double-click the downloaded file to run the Taxonomy Manager GUI, which will be set up to install the taxonomies you selected (at the website) for installation. (To open Altova's online XBRL Taxonomy Download Center in your browser and select the taxonomies you want to install, click **Altova on the Web | Taxonomy Download** in the add-in ribbon.)

After the Taxonomy Manager GUI (*screenshot below*) has been opened, already installed taxonomies will be shown selected. If you want to install an additional taxonomy, select it. If you want to uninstall an already installed taxonomy, deselect it. After you have made your selections and/or deselections, you are ready to apply your changes. The taxonomies that will be installed or uninstalled will be highlighted and a message about the upcoming changes will be posted to the Messages pane at the bottom of the Taxonomy Manager window (*see screenshot*).



⊟ ■ **ACPR COREP - Common Reporting with the modules LCR_CON, LCR_DA, NSFR_CON and ALM_CON**
Prudential reporting in the banking sector provided by the Bank of France.
https://esurfi-banque.banque-france.fr/

- ☑ 3.0
- □ 2.9
- □ 2.8
- □ 2.7
- □ 2.6

⊟ ■ **ACPR CREDITIMMO - Reporting for home loans and the profitability of production and outstanding home loa**
Prudential reporting in the banking sector provided by the Bank of France.
https://esurfi-banque.banque-france.fr/

- □ 2.4
- □ 2.3
- ☑ 2.1

⊞ □ **ACPR LCB-FT - Combating Money Laundering and the Financing of Terrorism**

⊞ □ **BBK - EBA-ITS-Reporting extended by the German base taxonomy**

⊞ □ **BDP - Financial and Common Reporting framework**

⊟ □ **BOE BANKING - Regulatory reporting for the banking sector**
Banks, building societies and designated investment firms must inform the Prudential Regulation Authority.
https://www.bankofengland.co.uk/prudential-regulation/regulatory-reporting/regulatory-reporting-banking-sect

The following packages will be uninstalled:
  BOE BANKING 3.5 - Regulatory reporting for the banking sector

The following packages will be installed:
  ACPR CREDITIMMO 2.1 - Reporting for home loans and the profitability of production and outstanding home loans

Command line interface

You can run Taxonomy Manager from a command line interface by sending commands to its executable file, `taxonomymanager.exe`.

The **`taxonomymanager.exe`** file is located in the following folder:

- *On Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *On Linux or macOS (server applications only):* `%INSTALLDIR%/bin`, where `%INSTALLDIR%` is the program's installation directory.

You can then use any of the commands listed in the CLI command reference section.

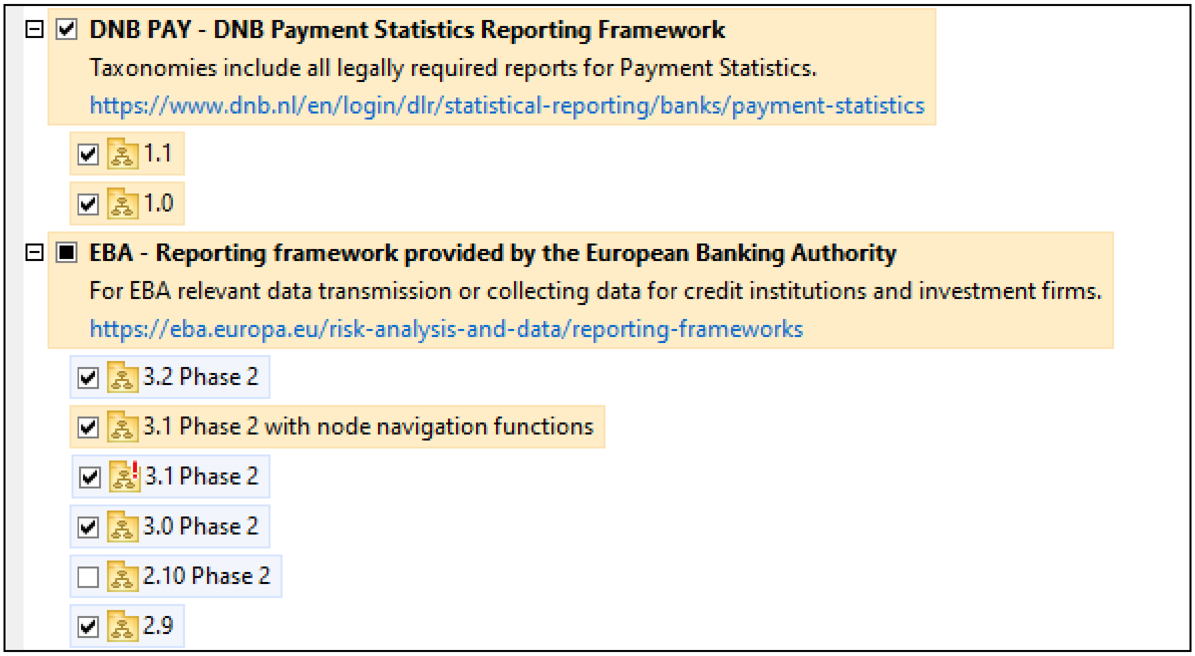
To display help for the commands, run the following:

- *On Windows:* `taxonomymanager.exe --help`
- *On Linux or macOS (server applications only):* `sudo ./taxonomymanager --help`

3.2 Status Categories

Taxonomy Manager categorizes the taxonomies under its management as follows:

- *Installed taxonomies.* These are shown in the GUI with their check boxes selected (*in the screenshot below the checked versions of the DNB and EBA taxonomies are installed taxonomies*). If all the versions of a taxonomy are selected, then the selection mark is a tick. If at least one version is unselected, then the selection mark is a solid colored square. You can deselect an installed taxonomy to **uninstall** it.
- *Uninstalled available taxonomies.* These are shown in the GUI with their check boxes unselected. You can select the taxonomies you want to **install**.



⊟ ☑ **DNB PAY - DNB Payment Statistics Reporting Framework**
Taxonomies include all legally required reports for Payment Statistics.
https://www.dnb.nl/en/login/dlr/statistical-reporting/banks/payment-statistics

☑ 1.1
☑ 1.0

⊟ ■ **EBA - Reporting framework provided by the European Banking Authority**
For EBA relevant data transmission or collecting data for credit institutions and investment firms.
https://eba.europa.eu/risk-analysis-and-data/reporting-frameworks

☑ 3.2 Phase 2
☑ 3.1 Phase 2 with node navigation functions
☑ 3.1 Phase 2
☑ 3.0 Phase 2
□ 2.10 Phase 2
☑ 2.9

- *Upgradeable taxonomies* are those which have been revised by their issuers since they were installed. They are indicated in the GUI by a icon (*see screenshot above*). You can **patch** an installed taxonomy with an available revision.

Points to note

- In the screenshot above, both DNB taxonomies and some of the EBA taxonomies are checked. Those with the blue background are already installed. Those with the yellow background are uninstalled and have been selected for installation. Note that (i) the EBA 2.10 Phase 2 taxonomy is not installed and has not been selected for installation, (ii) the EBA 3.1 Phase 2 taxonomy has been installed, but it has been patched by its issuer since it was installed and the patch has not yet been installed.
- When running Taxonomy Manager from the command line, the `list`[44] command is used with different options to list different categories of taxonomies:

`taxonomymanager.exe list`	Lists all installed and available taxonomies; upgradeables are also indicated

`taxonomymanager.exe list -i`	Lists installed taxonomies only; upgradeables are also indicated
`taxonomymanager.exe list -u`	Lists upgradeable taxonomies

Note: On Linux and macOS, **`use sudo ./taxonomymanager list`**

3.3 Patch or Install a Taxonomy

Patch an installed taxonomy

Occasionally, XBRL taxonomies may receive patches (upgrades or revisions) from their issuers. When Taxonomy Manager detects that patches are available, these are indicated in the taxonomy listings of Taxonomy Manager and you can install the patches quickly.

In the GUI

Patches are indicated by the icon. (*Also see the previous topic about status categories*[37].) If patches are available, the **Patch Selection** button will be enabled. Click it to select and prepare all patches for installation. In the GUI, the icon of each taxonomy that will be patched changes from to , and the Messages pane at the bottom of the dialog lists the patches that will be applied. When you are ready to install the selected patches, click **Apply**. All patches will be applied together. Note that if you deselect a taxonomy marked for patching, you will actually be uninstalling that taxonomy.

On the CLI

To apply a patch at the command line interface:

1. Run the `list -u`[44] command. This lists any taxonomies where patch upgrades are available.
2. Run the `upgrade`[47] command to install all the patches.

Install an available taxonomy

You can install taxonomies using either the Taxonomy Manager GUI or by sending Taxonomy Manager the install instructions via the command line.

Note: If the current taxonomy references other taxonomies, the referenced taxonomies are also installed.

In the GUI

To install taxonomies using the Taxonomy Manager GUI, select the taxonomies you want to install and click **Apply**.

You can also select the taxonomies you want to install at the Altova website and generate a downloadable **`.altova_taxonomies`** file. When you double-click this file, it will open Taxonomy Manager with the taxonomies you wanted pre-selected. All you will now have to do is click **Apply**.

On the CLI

To install taxonomies via the command line, run the `install`[44] command:

```
taxonomymanager.exe install [options] Taxonomy+
```

where **`Taxonomy`** is the taxonomy (or taxonomies) you want to install or a **`.altova_taxonomies`** file. A taxonomy is referenced by an identifier of format **`<name>-<version>`**. (The identifiers of taxonomies are displayed when you run the **`list`**[44] command.) You can enter as many taxonomies as you like. For details, see the description of the `install`[44] command.

Note: On Linux or macOS, use the **`sudo ./taxonomymanager`** command.

Installing a required taxonomy

When you run an XBRL-related command in EBA XBRL Add-in and EBA XBRL Add-in discovers that a taxonomy it needs for executing the command is not present or is incomplete, Taxonomy Manager will display information about the missing taxonomy. You can then directly install any missing taxonomy via Taxonomy Manager.

In the Taxonomy Manager GUI, you can view all previously installed taxonomies at any time by running Taxonomy Manager from **Tools | Taxonomy Manager**.

3.4 Uninstall a Taxonomy, Reset

Uninstall a taxonomy

You can uninstall taxonomies using either the Taxonomy Manager GUI or by sending Taxonomy Manager the uninstall instructions via the command line.

Note: If the taxonomy you want to uninstall references other taxonomies, then the referenced taxonomies are also uninstalled.

In the GUI

To uninstall taxonomies in the Taxonomy Manager GUI, clear their check boxes and click **Apply**. The selected taxonomies and their referenced taxonomies will be uninstalled.

To uninstall all taxonomies, click **Deselect All** and click **Apply**.

On the CLI

To uninstall taxonomies via the command line, run the **`uninstall`** command:

```
taxonomymanager.exe uninstall [options] Taxonomy+
```

where each **`Taxonomy`** argument is a taxonomy you want to uninstall or a **`.altova_taxonomies`** file. A taxonomy is specified by an identifier that has a format of **`<name>-<version>`**. (The identifiers of taxonomies are displayed when you run the **`list`** 44 command.) You can enter as many taxonomies as you like. For details, see the description of the `uninstall` 46 command.

Note: On Linux or macOS, use the **`sudo ./taxonomymanager`** command.

Reset Taxonomy Manager

You can reset Taxonomy Manager.

- In the GUI, click **Reset Selection**. This resets the the GUI to show what taxonomies are currently installed. Any selections or de-selections that the user has made in the current session will be canceled.
- On the CLI, run the `reset` 45 command. This removes all installed taxonomies and the cache directory.

After running this command, make sure to run the `initialize` 43 command in order to recreate the cache directory. Alternatively, run the `reset` 45 command with the `-i` option.

Note that `reset -i` 45 restores the original installation of the product, so it is recommended that you run the `update` 47 command after performing a reset. Alternatively, run the `reset` 45 command with the `-i` and `-u` options.

3.5 Command Line Interface (CLI)

To call Taxonomy Manager at the command line, you need to know the path of the executable. By default, the Taxonomy Manager executable is installed here:

```
C:\ProgramData\Altova\SharedBetweenVersions\TaxonomyManager.exe
```

Note: On Linux and macOS systems, once you have changed the directory to that containing the executable, you can call the executable with **`sudo ./taxonomymanager`**. The prefix **`./`** indicates that the executable is in the current directory. The prefix **`sudo`** indicates that the command must be run with root privileges.

Command line syntax

The general syntax for using the command line is as follows:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

In the listing above, the vertical bar **`|`** separates a set of mutually exclusive items. The square brackets **`[]`** indicate optional items. Essentially, you can type the executable path followed by either `--h`, `--help`, or `--version` options, or by a command. Each command may have options and arguments. The list of commands is described in the following sections.

3.5.1 help

This command provides contextual help about commands pertaining to Taxonomy Manager executable.

Syntax

```
<exec> help [command]
```

Where `[command]` is an optional argument which specifies any valid command name.

Note the following:

- You can invoke help for a command by typing the command followed by **`-h`** or **`--help`**, for example: **`<exec> list -h`**
- If you type **`-h`** or **`--help`** directly after the executable and before a command, you will get general help (not help for the command), for example: **`<exec> -h list`**

Example

The following command displays help about the `list` command:

```
taxonomymanager help list
```

3.5.2 info

This command displays detailed information for each of the taxonomies supplied as a `Taxonomy` argument. This information for each submitted taxonomy includes the title, version, description, publisher, and any dependent taxonomies, as well as whether the taxonomy has been installed or not.

Syntax

```
<exec> info [options] Taxonomy+
```

- The **`Taxonomy`** argument is the name of a taxonomy or a part of a taxonomy's name. (To display a taxonomy's package ID and detailed information about its installation status, you should use the `list` 44 command.)
- Use **`<exec> info -h`** to display help for the command.

Example

The following command displays information about the **`eba-2.10`** and **`us-gaap-2020.0`** taxonomies:

```
taxonomymanager info eba-2.1.0 us-gaap-2020.0
```

3.5.3 initialize

This command initializes the Taxonomy Manager environment. It creates a cache directory where information about all taxonomies is stored. Initialization is performed automatically the first time an XBRL-enabled Altova application is installed. You would not need to run this command under normal circumstances, but you would typically need to run it after executing the **`reset`** command.

Syntax

```
<exec> initialize | init [options]
```

Options

The **`initialize`** command takes the following options:

`--silent, --s`	Display only error messages. The default is **`false`**.
`--verbose, --v`	Display detailed information during execution. The default is **`false`**.
`--help, --h`	Display help for the command.

Example

The following command initializes Taxonomy Manager:

```
taxonomymanager initialize
```

3.5.4 install

This command installs one or more taxonomies.

Syntax

```
<exec> install [options] Taxonomy+
```

To install multiple taxonomies, add the **Taxonomy** argument multiple times.

The **Taxonomy** argument is one of the following:

- A taxonomy identifier (having a format of **<name>-<version>**, for example: **eba-2.10**). To find out the taxonomy identifiers of the taxonomies you want, run the list 44 command. You can also use an abbreviated identifier if it is unique, for example **eba**. If you use an abbreviated identifier, then the latest version of that taxonomy will be installed.
- The path to a **.altova_taxonomies** file downloaded from the Altova website. For information about these files, see *Introduction to TaxonomyManager: How It Works* 30.

Options
The **install** command takes the following options:

`--silent, --s`	Display only error messages. The default is **false**.
`--verbose, --v`	Display detailed information during execution. The default is **false**.
`--help, --h`	Display help for the command.

Example

The following command installs the latest **eba** (European Banking Authority) and **us-gaap** (US Generally Accepted Accounting Principles) taxonomies:

```
taxonomymanager install eba us-gaap
```

3.5.5 list

This command lists taxonomies under the management of Taxonomy Manager. The list displays one of the following

- All available taxonomies
- Taxonomies containing in their name the string submitted as a **Taxonomy** argument
- Only installed taxonomies
- Only taxonomies that can be upgraded

Syntax

```
<exec> list | ls [options] Taxonomy?
```

If no **Taxonomy** argument is submitted, then all available taxonomies are listed. Otherwise, taxonomies are listed as specified by the submitted options (*see example below*). Note that you can submit the **Taxonomy** argument multiple times.

Options
The **list** command takes the following options:

`--installed, --i`	List only installed taxonomies. The default is **false**.
`--upgradeable, --u`	List only taxonomies where upgrades (patches) are available. The default is **false**.
`--help, --h`	Display help for the command.

Examples

- To list all available taxonomies, run: **taxonomymanager list**
- To list installed taxonomies only, run: **taxonomymanager list -i**
- To list taxonomies that contain either "eba" or "us-gaap" in their name, run: **taxonomymanager list eba us-gaap**

3.5.6 reset

This command removes all installed taxonomies and the cache directory. You will be completely resetting your taxonomy environment. After running this command, be sure to run the `initialize` 43 command to recreate the cache directory. Alternatively, run the `reset` command with the `-i` option. Since `reset -i` restores the original installation of the product, we recommend that you run the `update` 47 command after performing a reset and initialization. Alternatively, run the reset command with both the `-i` and `-u` options.

Syntax

```
<exec> reset [options]
```

Options
The **reset** command takes the following options:

`--init, --i`	Initialize Taxonomy Manager after reset. The default is **false**.
`--update, --u`	Updates the list of available taxonomies in the cache. The default is **false**.
`--silent, --s`	Display only error messages. The default is **false**.
`--verbose, --v`	Display detailed information during execution. The default is **false**.
`--help, --h`	Display help for the command.

Examples

- To reset Taxonomy Manager, run: **`taxonomymanager reset`**
- To reset Taxonomy Manager and initialize it, run: **`taxonomymanager reset -i`**
- To reset Taxonomy Manager, initialize it,and update its taxonomy list, run: **`taxonomymanager reset -i -u`**

3.5.7 uninstall

This command uninstalls one or more taxonomies. By default, any taxonomies referenced by the current one are uninstalled as well. To uninstall just the current taxonomy and keep the referenced taxonomies, set the option `--k`.

Syntax

```
<exec> uninstall [options] Taxonomy+
```

To uninstall multiple taxonomies, add the **`Taxonomy`** argument multiple times.

The **`Taxonomy`** argument is one of the following:

- A taxonomy identifier (having a format of **`<name>-<version>`**, for example: **`eba-2.10`**). To find out the taxonomy identifiers of the taxonomies that are installed, run the `list -i` 44 command. You can also use an abbreviated taxonomy name if it is unique, for example **`eba`**. If you use an abbreviated name, then all taxonomies that contain the abbreviation in its name will be uninstalled.
- The path to a **`.altova_taxonomies`** file downloaded from the Altova website. For information about these files, see *Introduction to TaxonomyManager: How It Works* 30.

Options
The **`uninstall`** command takes the following options:

`--keep-references, --k`	Set this option to keep referenced taxonomies. The default is **`false`**.
`--silent, --s`	Display only error messages. The default is **`false`**.
`--verbose, --v`	Display detailed information during execution. The default is **`false`**.
`--help, --h`	Display help for the command.

Example

The following command uninstalls the **`eba-2.10`** and **`us-gaap-2020.0`** taxonomies and their dependencies:

```
taxonomymanager uninstall eba-2.10 us-gaap-2020.0
```

The following command uninstalls the **`eba-2.10`** taxonomy but not the taxonomies it references:

```
taxonomymanager uninstall --k eba-2.10
```

3.5.8 update

This command queries the list of taxonomies available from the online storage and updates the local cache directory. You should not need to run this command unless you have performed a `reset` [45] and `initialize` [43].

Syntax

```
<exec> update [options]
```

Options
The **`update`** command takes the following options:

`--silent, --s`	Display only error messages. The default is **`false`**.
`--verbose, --v`	Display detailed information during execution. The default is **`false`**.
`--help, --h`	Display help for the command.

Example

The following command updates the local cache with the list of latest taxonomies:

```
taxonomymanager update
```

3.5.9 upgrade

This command upgrades all installed taxonomies that can be upgraded to the latest available *patched* version. You can identify upgradeable taxonomies by running the `list -u` [44] command.

Note: The **`upgrade`** command removes a deprecated taxonomy if no newer version is available.

Syntax

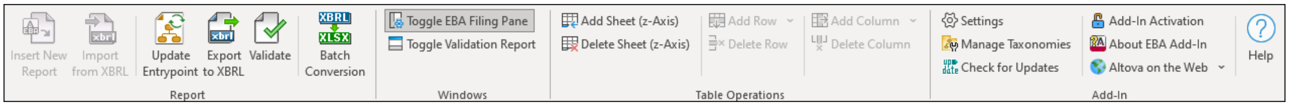
```
<exec> upgrade [options]
```

Options
The **`upgrade`** command takes the following options:

`--silent, --s`	Display only error messages. The default is **`false`**.
`--verbose, --v`	Display detailed information during execution. The default is **`false`**.
`--help, --h`	Display help for the command.

4 Menu Commands

This section describes commands available in the EBA ribbon (*see screenshot below*). The commands are organized into four groups: *Report*, *Windows*, *Table Operations* and *Add-In*. For more information, see the subsections below.



Report

Insert New Report

Creates a new report. This command is disabled if the report sheet has already been inserted into the workbook. For more information, see Create a New Report 11.

Import from XBRL

Imports an XBRL instance file into the current Excel spreadsheet. This command is disabled if the report has already been inserted into the workbook. To enable the command, save and close the current report (workbook) and create a new workbook. For details, see Import Data from XBRL 27.

Update Entrypoint

The **Update Entrypoint** command enables you to switch to a different version of the same taxonomy. Clicking this command opens the **Select Entry Point** dialog, in which you can choose another taxonomy version to download.

Export to XBRL

Exports data from all currently active sheets to an XBRL instance file. For more details, see Export Data to XBRL 26.

Validate

Validates report data against the underlying XBRL taxonomy and displays validation results in the **Validation Report** dialog box. For details, see Validate Data 24.

Batch Conversion (Enterprise Edition)

Converts multiple XBRL instance files to Excel. For more information, see Batch Convert XBRL to Excel 28.

Windows

Toggle Filing Pane

Toggles the Filing Pane on or off. By default, this pane is visible.

Toggle Validation Report

Shows or hides the Validation Report window. See Validate Data[24].

Table Operations

⊟ Add Sheet (z-Axis)

Adds a new sheet which allows entering data in a third dimension. This button is enabled only if a third dimension is allowed for the current table by the XBRL taxonomy. For more information, see Enter Data into 3D Tables[21].

⊟ Delete Sheet (z-Axis)

Removes the previously added z-axis sheet.

⊟ Add Row

Adds a new row above or below the currently selected row. This button is enabled only if extending the current table vertically is supported by the taxonomy.

⊟ Delete Row

Deletes the currently selected row. This button is enabled only if extending the current table vertically is supported by the taxonomy.

⊟ Add Column

Adds a new column to the left or to the right of the currently selected column. This button is enabled only if extending the current table horizontally is supported by the taxonomy.

⊟ Delete Column

Removes the currently selected column. This button is enabled only if extending the current table horizontally is supported by the taxonomy.

Add-In

⊟ Settings

Displays a dialog box where you can view and change the add-in settings.

⊟ Manage Taxonomies

This command opens the XBRL Taxonomy Manager tool, which allows viewing, installing, and uninstalling XBRL taxonomies. See XBRL Taxonomy Manager[30].

⊟ Check for Updates

Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.

⊟ Add-In Activation

Displays the activation status of the add-in and provides options to enter or purchase a license key code.

⊟ About EBA XBRL Add-in

Displays information about the add-in version.

⊟ Altova on the Web

Provides links to the Altova website, including the Online Support Center, XBRL Taxonomy Download Center, training and tutorials.

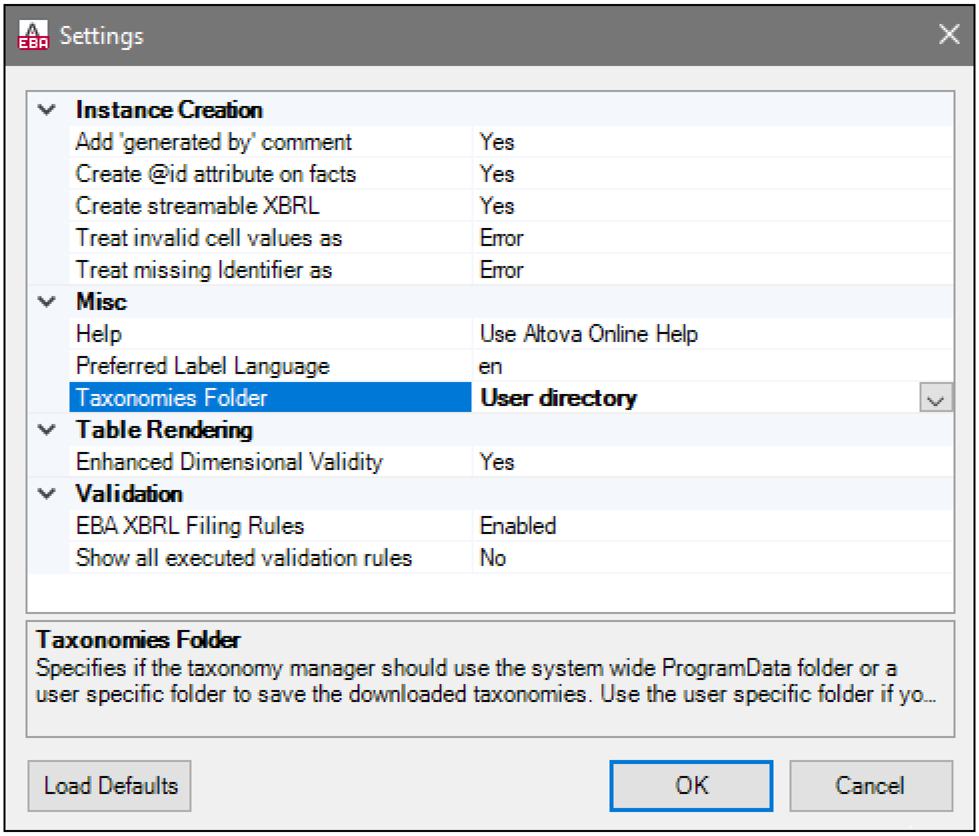
⊟ Help

The **Help** command opens the add-in's Help documentation (its user manual). By default, the Online Help in HTML format on the Altova website will be opened.

If you do not have Internet access or do not want, for some other reason, to access the Online Help, you can use the locally stored version of the user manual. The local version is a PDF file named **`EBA XBRL Add-in.pdf`** that is stored in the application folder (in the Program Files folder). If you want to change the default format of the user manual, you can do this in the *Misc* section of the **Settings** dialog.

5 Settings

The **Settings** dialog (*see screenshot below*) allows you to configure instance creation, Help and language parameters, table rendering and define validation settings. To view or change the add-in settings, click **Settings** in the add-in ribbon.



Instance Creation	
Add 'generated by' comment	Yes
Create @id attribute on facts	Yes
Create streamable XBRL	Yes
Treat invalid cell values as	Error
Treat missing Identifier as	Error
Misc	
Help	Use Altova Online Help
Preferred Label Language	en
Taxonomies Folder	**User directory**
Table Rendering	
Enhanced Dimensional Validity	Yes
Validation	
EBA XBRL Filing Rules	Enabled
Show all executed validation rules	No

Taxonomies Folder
Specifies if the taxonomy manager should use the system wide ProgramData folder or a user specific folder to save the downloaded taxonomies. Use the user specific folder if yo...

The settings you can configure are listed below.

Instance Creation

The *Instance Creation* section allows you to define general settings related to instance creation (e.g., the way invalid cell values should be treated).

⊟ Add 'generated by' comment

Specifies if the exported XBRL instances should contain a *Generated by* comment. Set this option to `No` if your reporting authority does not allow comments in the created reports. The default value is `Yes`.

⊟ Create @id attribute on facts

According to the *EBA XBRL Filing Rules (Section 3.7)*, unused **`@id`** attributes on facts should not be included in the report unless the reporter needs them. In the EBA Add-in, **`@id`** attributes are added to all facts, because this allows navigating to cells with validation errors or warnings. By default, the option *Create @id attribute on facts* is set to `Yes`. If you select `No`, unused **`@id`** attributes on facts will not be included in the XBRL report.

⊟ Create streamable XBRL

Specifies if the exported XBRL instances should be streamable and contain the `xbrl-streamable-instance` processing instruction. Set this option to `No` if your reporting authority does not allow processing instructions in reports. The default value is `Yes`. For more information about streamable XBRL instances, see *XBRL Streaming Extensions Module 1.0*.

⊟ Treat invalid cell values as

Specifies whether invalid cell values should be treated as validation errors or warnings. The default value is `Error`.

⊟ Treat missing Identifier as

The *Identifier* property is available in the Filing Pane. By default, this parameter is empty. The *Treat missing Identifier as* option specifies whether an empty *Identifier* property should trigger a validation error or a warning. The default value is `Error`.

Misc

The *Misc* section enables you to specify the format of the add-in manual, the language used in headers, and the location of downloaded taxonomies.

⊟ Help

Specifies whether the Altova Online Help in HTML format (default option) or the locally installed PDF file should be used.

⊟ Preferred Label Language

Specifies the preferred language to be used in the headers of created worksheets. Note that the respective label resources must be defined in the taxonomy for this setting to take effect. The default value is `en`.

⊟ Taxonomies Folder

This setting enables you to specify whether downloaded taxonomies should be stored in the global-package **`ProgramData`** folder (`C:\ProgramData\Altova\pkgs\.cache`) or the user folder (**`<USER-HOME>\Documents\Altova\pkgs\.cache`**). Select the user folder if you do not have permission to write in the **`ProgramData`** folder.

Note that nothing is shared between the global-package folder and the user folder. Therefore, if you decide to switch from one folder to the other (e.g., from the global-package folder to the user folder) and you need the same taxonomies in the second folder, you must install all these taxonomies again.

Table Rendering

The *Table Rendering* section includes the *Dimension Validity* property.

⊟ Enhanced Dimensional Validity

This option activates only those cells that are dimensionally valid in at least one base set with a linkrole that contains the corresponding RCCode of the table. The default and recommended setting for the *Enhanced Dimensional Validity* option is `Yes`. If you encounter problems with data submission to your reporting authority, set this option to `No`. For more information about Dimensional Validity, see *Technical Considerations for the Use of XBRL Dimensions 1.0*.

Validation

The *Validation* section enables you to specify whether additional XBRL filing rules should be used and whether all executed validation rules should be included in the report.

⊟ XBRL Filing Rules

Specifies if the additional EBA XBRL filing rules (defined by the EBA XBRL Filing Rules) should be checked. The default value is `Enabled`.

⊟ Show all executed validation rules

Shows all the executed validation rules, including the failed ones, in the report. The default option is `No`.

6 COM API

The add-in provides a COM API that can be used from programming languages that support interacting with Excel and accessing COM objects programmatically, such as VBA or .NET languages. Specifically, the API enables you to create, import and export XBRL reports, read and write form data.

Platform requirements:

- .NET Framework 4.5 or later
- Visual Studio Tools for Office runtime, Version 4.0 or later

If you intend to distribute the API to other clients, note the following:

1. Excel and EBA XBRL Add-in must be installed on each client machine.
2. Each API client that uses your custom code or application must hold a valid EBA XBRL Add-in license.

6.1 Access API

You can access COM API of the add-in programmatically in one of the following ways:

- From Excel by using Visual Basic for Applications
- From your custom program by using the Office Interop API from any .NET language

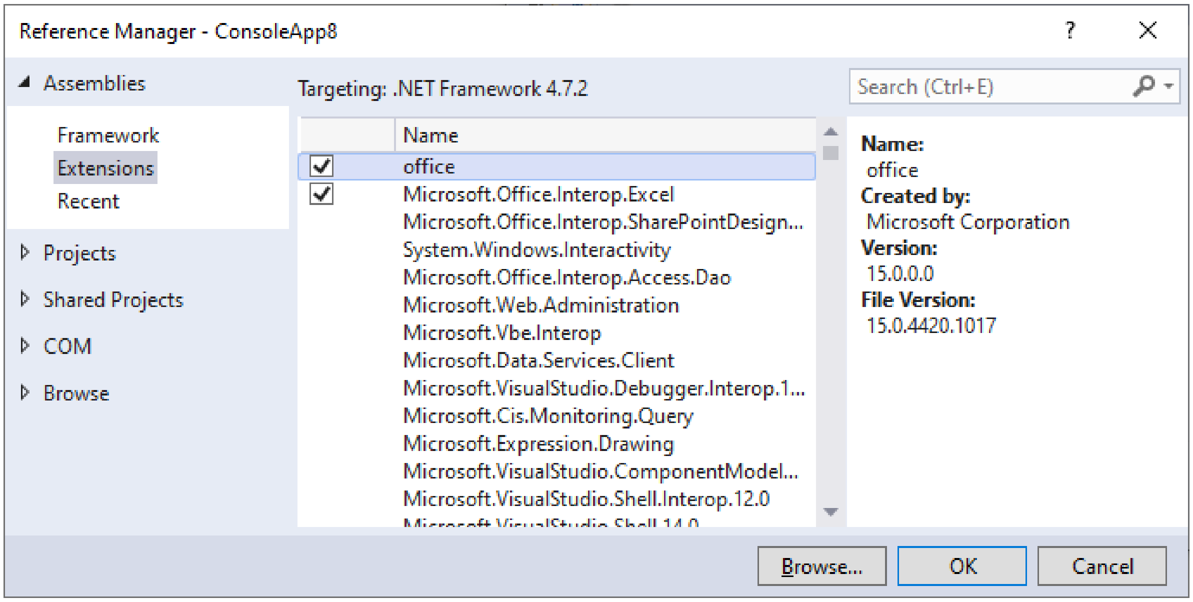
The main interface is the `IAutomationAPI` interface. The code listing below shows how to create a new instance of the automation object in VBA:

```
Dim automationObject As Object
Set automationObject = Application.COMAddIns.Item("Altova.EBAAddIn").Object
```

Access COM API from a .NET project

To access the COM API from a Visual Studio .NET project, add a reference to the assemblies **Microsoft Office Object Library (office.dll)** and **Microsoft.Office.Interop.Excel**, as shown below:

1. Right-click your project's name in **Solution Explorer** and then click **Add Reference**. The **Add Reference** dialog box appears.
2. On the **Assemblies** page, select **office** and **Microsoft.Office.Interop.Excel** from the component list and click **OK**.



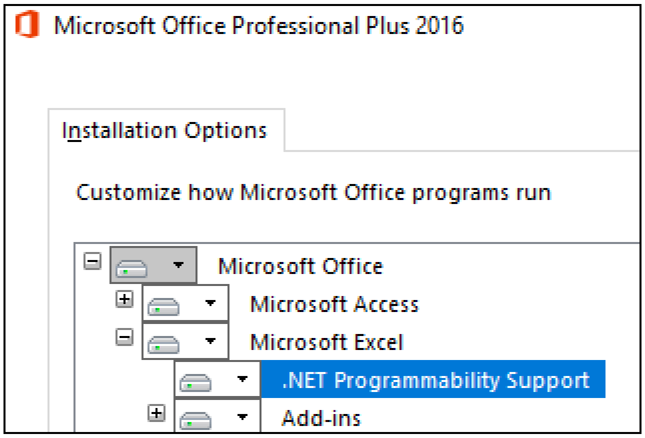
Targeting: .NET Framework 4.7.2

☑ office
☑ Microsoft.Office.Interop.Excel
Microsoft.Office.Interop.SharePointDesign...
System.Windows.Interactivity
Microsoft.Office.Interop.Access.Dao
Microsoft.Web.Administration
Microsoft.Vbe.Interop
Microsoft.Data.Services.Client
Microsoft.VisualStudio.Debugger.Interop.1...
Microsoft.Cis.Monitoring.Query
Microsoft.Expression.Drawing
Microsoft.VisualStudio.ComponentModel...
Microsoft.VisualStudio.Shell.Interop.12.0
Microsoft.VisualStudio.Shell.14.0

Name: office
Created by: Microsoft Corporation
Version: 15.0.0.0
File Version: 15.0.4420.1017

If you do not see the assemblies above, take the following steps:

1. Make sure that you have installed Microsoft Office and that you have selected the **.NET Programmability Support** feature for Excel (*see screenshot below*).



Microsoft Office Professional Plus 2016

Installation Options

Customize how Microsoft Office programs run

2. Run the Visual Studio setup and make sure that you choose the **Office/SharePoint development** workload or **Microsoft Office Developer Tools**, if applicable.

For more information about accessing Office interop assemblies from .NET projects, see the Microsoft documentation. After adding the assembly references, you can create a new add-in instance as shown below.

C#

```
// Make sure that your project references the following two assemblies:
//  * Microsoft.Office.InterOp.Excel
//  * Microsoft Office Object Library (office.dll)
var app = new Microsoft.Office.Interop.Excel.Application();
dynamic automationObject = app.COMAddIns.Item("Altova.EBAAddIn").Object;
```

6.2 C# Example

Each of the C# code listings illustrated below represents the **`Program.cs`** file in a standard .NET Framework console application. Before attempting to run the program code, make sure that you have added the required assembly references to the Visual Studio project, as described in Access API[55].

Export XBRL from Excel

The code listing below shows how to create a new Excel workbook using a specific XBRL entry point, populate a few properties and data cells, and save data to an XBRL file on the disk. The code will help you save both the Excel workbook and the XBRL file to the **`C:\XBRL_Examples`** directory.

```
using System;
using Excel = Microsoft.Office.Interop.Excel;

namespace EbaAddinClient
{
    class Program
    {
        static void Main(string[] args)
        {
            var app = new Excel.Application();

            try
            {
                // Suppress Excel alerts and create a new workbook
                Console.WriteLine("Creating a new workbook...");
                app.DisplayAlerts = false;
                var wb = (Excel._Workbook)(app.Workbooks.Add());

                // Get the Automation API object
                Console.WriteLine("Getting the COM automation object...");
                dynamic addIn = app.COMAddIns.Item("Altova.EBAAddIn");
                dynamic automationObject = addIn.Object;

                // Create a new report using taxonomy entry point
                Console.WriteLine("Creating the new report...");
                automationObject.InsertNewReport("http://www.eba.europa.eu/eu/fr/xbr
l/crr/fws/fp/gl-2019-05/2020-02-29/mod/fp_con.xsd");

                // Set the report properties
                Console.WriteLine("Setting the report properties...");
                var rp = automationObject.GetReportProperties(wb);
                rp.ReportingEntityScheme = "http://standards.iso.org/iso/17442";
                rp.ReportingEntityIdentifier = "123456";

                // Find table by its code and ensure it is included in this report
                var tab = automationObject.GetTableTree(wb);
                var tableNode = tab.FindTableByRCCode("P 00.01");
                tableNode.IncludeInFiling = true;

                // Populate cells
                Console.WriteLine("Populating cells...");
                tableNode.Forms.Item(0).DataRange.Item(1).Value = "National GAAP";
                tableNode.Forms.Item(0).DataRange.Item(2).Value = "Consolidated";

                // Export data to XBRL
                Console.WriteLine("Exporting the XBRL instance...");
                automationObject.ExportXBRL(wb, @"C:\XBRL_Examples\Example.xbrl");

                // Save and close the .xlsx workbook
                Console.WriteLine("Saving the .xlsx file...");
                wb.SaveAs(@"C:\XBRL_Examples\Example.xlsx");
                wb.Close();

                Console.WriteLine("Task completed.");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                app.DisplayAlerts = true;
                app.Quit();
            }
        }
    }
}
```

Import XBRL to Excel

The code listing below shows how to convert an XBRL file to an Excel file. To run this example successfully, an XBRL instance file must exist at **`C:\XBRL_Examples\Example.xbrl`**. Otherwise, change the path accordingly. Use the **Export** command to create an XBRL file by running the previous code listing or manually from Excel. For more information, see Export Data to XBRL 26.

```
using System;
using Excel = Microsoft.Office.Interop.Excel;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var app = new Excel.Application();
            try
            {
                // Suppress Excel alerts and create a new workbook
                Console.WriteLine("Creating a new workbook...");
                app.DisplayAlerts = false;
                var wb = (Excel._Workbook)(app.Workbooks.Add());

                // Get the Automation API object
                Console.WriteLine("Getting the COM automation object...");
                dynamic addIn = app.COMAddIns.Item("Altova.EBAAddIn");
                dynamic automationObject = addIn.Object;

                // Import EBA report eba_example.xbrl
```

```
                        Console.WriteLine("Importing XBRL...");
                        automationObject.ImportXBRL(@"C:\XBRL_Examples\Example.xbrl");

                        // Save as xlsx
                        Console.WriteLine("Saving the .xlsx file...");
                        wb.SaveAs(@"C:\XBRL_Examples\Example.xlsx");
                        wb.Close();

                        Console.WriteLine("Task complete.");
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine(e.Message);
                    }
                    finally
                    {
                        app.DisplayAlerts = true;
                        app.Quit();
                    }
                }
            }
        }
```

Map XBRL table to Excel report table

The code listing below shows how to map an XBRL table to a report table in Excel.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Excel = Microsoft.Office.Interop.Excel;

namespace EBAAddInAPITest
{
    class Program
    {
        internal struct Data
        {
            internal Data(string rc, string cc, string value)
            {
                RowCode = rc;
                ColumnCode = cc;
                Value = value;
            }

            internal string RowCode;
            internal string ColumnCode;
            internal string Value;
        }

        static Data[] testData = new Data[]
        {
            new Data("0010", "0060", "163.59000" ),
            new Data("0010", "0070", "2032.33000" ),
            new Data("0020", "0010", "1016.16000"),
            new Data("0022", "0010", "1000.00000"),
            new Data("0040", "0010", "16.16000"),
            new Data("0050", "0010", "1016.16000"),
            new Data("0020", "0030", "16.16000"),
            new Data("0020", "0050", "1016.16000"),
            new Data("0020", "0060", "81.29000"),
            new Data("0040", "0030", "16.16000"),
            new Data("0050", "0030", "1016.16000"),
            new Data("0050", "0050", "1016.16000"),
            new Data("1050", "0060", "81.29000")
        };

        static void Main(string[] args)
        {
            var app = new Excel.Application();

            try
            {
                // Suppress Excel alerts and create a new workbook
                Console.WriteLine("Creating a new workbook...");
                app.DisplayAlerts = false;
                app.Visible = false;
                var wb = (Excel._Workbook)(app.Workbooks.Add());

                // Get the Automation API object
                Console.WriteLine("Getting the COM automation object...");
                // dynamic addIn = app.COMAddIns.Item("Altova.EBAAddIn");
                dynamic addIn = app.COMAddIns.Item("EBAAddInForExcel");
                dynamic automationObject = addIn.Object;

                // Create a new report using taxonomy entry point
                Console.WriteLine("Creating the new report...");
                automationObject.InsertNewReport("http://www.eba.europa.eu/eu/fr/xbrl/crr/f
ws/if/its-002-2021/2021-05-08/mod/if_class2_con.xsd");

                // Set the report properties
                Console.WriteLine("Setting the report properties...");
                var rp = automationObject.GetReportProperties(wb);
                rp.ReportingEntityScheme = "http://standards.iso.org/iso/17442";
                rp.ReportingEntityIdentifier = "123456";

                // Find table by its code and ensure it is included in this report
                var tab = automationObject.GetTableTree(wb);
                var tableNode = tab.FindTableByRCCode("C 21.00");
                tableNode.IncludeInFiling = true;

                // Read column/row headers
                var columnIndices = new Dictionary<string, int>();
                var rowIndices = new Dictionary<string, int>();
                var form = tableNode.Forms.Item(0);
                var dataRange = form.DataRange;
                for (int i = 1; i <= dataRange.Columns.Count; ++i)
                    columnIndices.Add(dataRange.Item(0, i).Value.ToString(), i);
                for (int i = 1; i <= dataRange.Rows.Count; ++i)
                    rowIndices.Add(dataRange.Item(i, 0).Value.ToString(), i);
                // Populate cells
                Console.WriteLine("Populating cells...");
                foreach (var data in testData)
                {
                    int row, column;
                    if (!rowIndices.TryGetValue(data.RowCode, out row))
                    {
                        Console.WriteLine(String.Format("Warning: RowCode {0} not found in
form {1}!", data.RowCode, form.Text));
                        continue;
                    }
                    if (!columnIndices.TryGetValue(data.ColumnCode, out column))
                    {
                        Console.WriteLine(String.Format("Warning: ColumnCode {0} not found
in form {1}!", data.ColumnCode, form.Text));
                        continue;
                    }

                    dataRange.Item(row, column).Value = data.Value;
                }

                // Export data to XBRL
                Console.WriteLine("Exporting the XBRL instance...");
                automationObject.ExportXBRL(wb, @"C:\XBRL_Examples\Example.xbrl");

                var valReport = automationObject.GetValidationReport(wb);
                if (valReport != null)
                    Console.WriteLine(valReport.CreateTextReport());

                // Save and close the .xlsx workbook
                Console.WriteLine("Saving the .xlsx file...");
                wb.SaveAs(@"C:\XBRL_Examples\Example.xlsx");
                wb.Close();

                Console.WriteLine("Task completed.");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                app.DisplayAlerts = true;
                app.Quit();
            }
        }
    }
}
```

6.3 VBA Example

The following code listing shows how to insert an XBRL report into an Excel file and populate the first cell using VBA:

```
' VBA Example 1:
' Creates a new EBA COREP ALM report with form 'C 68.00.a' and sets the value of the
first cell
Sub Example1()
    Dim addIn As COMAddIn
    Dim automationObject As Object
    Dim Workbook As Object
    Dim tableTree As Object
    Dim tableNode As Object

    ' Get the Automation API object
    Set addIn = Application.COMAddIns.Item("Altova.EBAAddIn")
    Set automationObject = addIn.Object

    ' Insert a new EBA 2.9.1 COREP ALM report
    Set Workbook =
automationObject.InsertNewReport("http://www.eba.europa.eu/eu/fr/xbrl/crr/fws/corep/
cir-680-2014/2019-11-15/mod/corep_alm_con.xsd")
    Set tableTree = automationObject.GetTableTree(Workbook)

    ' Find table tree node for form 'C 68.00.a'
    Set tableNode = tableTree.FindTableByRCCode("C 68.00.a")

    ' Include this table in the filing (this will also create the respective Excel
worksheet)
    tableNode.IncludeInFiling = True

    ' Get the Data range of this form and set the value of the first cell to 42
    tableNode.Forms.Item(0).DataRange.Item(1).Value = "42"
End Sub
```

6.4 API Reference

This section provides information about the interfaces of the EBA Add-in COM API.

6.4.1 IAutomationAPI

The `IAutomationAPI` interface is the main automation interface of the EBA Add-in. This interface is the starting point for doing any further operations with the add-in. This interface allows you to create, import and export reports, read and write data. For information about creating an instance of this interface, see Accessing the API[55].

Methods

The `IAutomationAPI` interface allows you to use the following methods:

⊟ InsertNewReport

Use this method to insert a new report of the respective taxonomy entry point.

Signature

```
InsertNewReport(in entryPointUrl:String) -> Microsoft.Office.Interop.Excel.Workbook
```

Parameters

Name	Type	Description
entryPointUrl	`String`	The URI of the taxonomy a report should be created for. Use the `GetEntryPointTree` method to get the available entry points.

⊟ ImportXBRL

Imports an XBRL report. Returns the Excel workbook that contains the imported XBRL report.

Signature

```
ImportXBRL(in inputFile:String) -> Microsoft.Office.Interop.Excel.Workbook
```

Parameters

Name	Type	Description
inputFile	`String`	The path to the XBRL report which should be imported.

⊟ ExportXBRL

Exports the report from the respective Excel workbook to XBRL and also validates it. To get the validation results, call `GetValidationReport` after calling this method. You can get the file name by using the `GetReportBaseName` method.

Signature

```
ExportXBRL(workbook as Microsoft.Office.Interop.Excel.Workbook, outputFile as String)
-> Void
```

Parameters

Name	Type	Description
`workbook`	`Microsoft.Office.Interop.Excel.Workbook`	The Excel workbook to be exported.
`outputFile`	`String`	The path to the output XBRL file.

⊟ ExportXBRL_CSV

Exports the report from the respective Excel workbook to the XBRL-CSV report package and validates it. To get the validation results, call `GetValidationReport` after calling this method.

Signature

```
ExportXBRL_CSV(in workbook:Microsoft.Office.Interop.Excel.Workbook, in
outputFile:String) -> Void
```

Parameters

Name	Type	Description
workbook	`Microsoft.Office.Interop.Excel.Workbook`	The Excel workbook to be exported.
outputFile	`String`	The path to the output XBRL-CSV report package file.

⊟ Validate

Validates the current report. To get the validation results, call `GetValidationReport` after calling this method.

Signature

```
Validate(in workbook:Microsoft.Office.Interop.Excel.Workbook) -> Void
```

Parameters

Name	Type	Description
workbook	`Microsoft.Office.Interop.Excel.Workbook`	The Excel workbook to be validated.

⊟ GetValidationReport

Returns an `IValidationReport` object representing the validation report currently shown in the validation report pane.

Signature

```
GetValidationReport(in workbook:Microsoft.Office.Interop.Excel.Workbook) ->
IValidationReport
```

Parameters

Name	Type	Description
workbook	**Microsoft.Office.Interop.Excel.Workbook**	The Excel workbook from which to get the validation report.

GetEntryPointTree

Returns an `IEntryPointTree` object representing a tree of the available taxonomy entry points.

Signature

```
GetEntryPointTree() -> IEntryPointTree
```

GetTableTree

Returns an `ITableTree` object representing the tree of the available tables in the report opened in the specified Excel workbook.

Signature

```
GetTableTree(in workbook:Microsoft.Office.Interop.Excel.Workbook) -> ITableTree
```

Parameters

Name	Type	Description
workbook	**Microsoft.Office.Interop.Excel.Workbook**	The Excel workbook containing the XBRL report.

GetReportProperties

Returns an `IReportProperties` object providing the properties of the XBRL report.

Signature

```
GetReportProperties(in workbook:Microsoft.Office.Interop.Excel.Workbook) ->
IReportProperties
```

Parameters

Name	Type	Description
workbook	**Microsoft.Office.Interop.Excel.Workbook**	The workbook containing the XBRL report.

GetFormProperties

Returns an `IFormProperties` object providing the properties of the XBRL form in the specified Excel worksheet.

Signature

```
GetFormProperties(in worksheet:Microsoft.Office.Interop.Excel.Worksheet) ->
IFormProperties
```

Parameters

Name	Type	Description
worksheet	`Microsoft.Office.Interop.Excel.Worksheet`	An Excel worksheet containing an XBRL form.

GetCellProperties

Returns an `ICellProperties` object providing the properties of the fact in the specified Excel range.

Signature

```
GetCellProperties(in cell:Microsoft.Office.Interop.Excel.Range) -> ICellProperties
```

Parameters

Name	Type	Description
cell	`Microsoft.Office.Interop.Excel.Range`	An Excel cell containing an XBRL fact (must be within the data range of a form).

GetReportBaseName

Returns the file name that Excel provides automatically when you export an XBRL file (via the `ExportXBRL` method) from an Excel workbook.

Signature

```
GetReportBaseName(workbook as Excel.Workbook) -> String
```

Parameters

Name	Type	Description
`workbook`	`Excel.Workbook`	The name of the XBRL file exported via the `ExportXBRL` method.

6.4.2 IEntryPointTree

The `IEntryPointTree` interface provides information about the available taxonomy entry points in a structured way.

Properties

Name	Description
`Groups` 67	Read-only. Returns a collection of `IEntryPointGroup` representing the available taxonomy groups, for example, "2.9" or "Bank of England Banking Taxonomy 3.1.1".

6.4.2.1 Properties

6.4.2.1.1 Groups

Returns a collection of `IEntryPointGroup` representing the available taxonomy groups, for example, "2.9" or "Bank of England Banking Taxonomy 3.1.1".

Signature

```
Groups : Collection
```

6.4.3 IEntryPointGroup

The `IEntryPointGroup` interface provides information about a group of taxonomy entry points.

Properties

Name	Description
`Name` 68	Read-only. The Name of the group. For example, "EBA 2.9".
`Country` 68	Read-only. The country code for which this entry point group is relevant. This is set only for country-specific taxonomies such as "Bank of England Banking Taxonomy".
`Version` 68	Read-only. The version of the taxonomy. May be empty for sub-groups.

Name	Description
`IsCurrentVersion` 69	Read-only. **True** if this group contains the current version of the taxonomy, **false** for older versions.
`Groups` 69	Read-only. A collection of `IEntryPointGroup` representing the sub-groups of this entry point group.
`EntryPoints` 69	Read-only. A collection of `IEntryPoint` representing the specific taxonomy entry points of this group.

6.4.3.1 Properties

6.4.3.1.1 Name

The Name of the group. For example, "EBA 2.9".

Signature

```
Name : String
```

6.4.3.1.2 Country

The country code for which this entry point group is relevant. This is set only for country-specific taxonomies such as "Bank of England Banking Taxonomy".

Signature

```
Country : String
```

6.4.3.1.3 Version

The version of the taxonomy. May be empty for sub-groups.

Signature

```
Version : String
```

6.4.3.1.4 IsCurrentVersion

True if this group contains the current version of the taxonomy; **false** for older versions.

Signature

```
IsCurrentVersion : Boolean
```

6.4.3.1.5 Groups

A collection of `IEntryPointGroup` representing the sub-groups of this entry point group.

Signature

```
Groups : Collection
```

6.4.3.1.6 EntryPoints

A collection of `IEntryPoint` representing the specific taxonomy entry points of this group.

Signature

```
EntryPoints : Collection
```

6.4.4 IEntryPoint

The `IEntryPoint` interface provides information about a specific taxonomy entry point.

Properties

Name	Description
`Name` 70	Read-only. The name of the entry point. For example, "Additional Liquidity Monitoring - COREP, Consolidated".
`ShortName` 70	Read-only. The abbreviated form of the entry points name. For example, "COREP ALM Con".
`Version` 70	Read-only.

Name	Description
	The version of the entry point. This may be empty (use the parents group version in this case).
`URI` 71	Read-only. The URI of the taxonomy entry point, for example, "http://www.eba.europa.eu/eu/fr/xbrl/crr/fws/corep/cir-680-2014/2019-04-30/mod/corep_alm_con.xsd". Pass this to the `IAutomationAPI.InsertNewReport` method.
`Needs64Bit` 71	Read-only. **True** if the entry point requires the 64-bit version of Excel; **false** otherwise.

6.4.4.1 Properties

6.4.4.1.1 Name

The name of the entry point. For example, "Additional Liquidity Monitoring - COREP, Consolidated".

Signature

```
Name : String
```

6.4.4.1.2 ShortName

The abbreviated form of the entry points name. For example, "COREP ALM Con".

Signature

```
ShortName : String
```

6.4.4.1.3 Version

The version of the entry point. This may be empty (use the parents group version in this case).

Signature

```
Version : String
```

6.4.4.1.4 URI

The URI of the taxonomy entry point, for example, "http://www.eba.europa.eu/eu/fr/xbrl/crr/fws/corep/cir-680-2014/2019-04-30/mod/corep_alm_con.xsd". Pass this to the `IAutomationAPI.InsertNewReport` method.

Signature

```
URI : String
```

6.4.4.1.5 Needs64Bit

True if the entry point requires the 64-bit version of Excel; **false** otherwise.

Signature

```
Needs64Bit : Boolean
```

6.4.5 ITableTree

The `ITableTree` interface provides structured information about the available tables and forms in an XBRL report.

Properties

Name	Description
`Nodes` 72	Read-only. Returns a collection of `IGroupNode` and `ITableNode` objects, which represent groups of tables and tables respectively.

Methods

Name	Description
`FindTableByRCCode` 72	Returns the table node with the specified RC Code.

6.4.5.1 Properties

6.4.5.1.1 Nodes

Returns a collection of `IGroupNode` and `ITableNode` objects, which represent groups of tables and tables respectively.

Signature

```
Nodes : Collection
```

6.4.5.2 Methods

6.4.5.2.1 FindTableByRCCode

Returns the table node with the specified RC Code.

Signature

```
FindTableByRCCode(in rcCode:string) -> ITableNode
```

Parameters

Name	Type	Description
rcCode	`string`	The RC Code of the desired table node.

6.4.6 IGroupNode

The `IGroupNode` interface provides information about a group of tables.

Properties

Name	Description
`Text` 73	Read-only. The name of the group of tables as displayed in the EBA Filing Pane.
`IsGroup` 73	Read-only.

Name	Description
	This is always **true** for group nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.
`IsTable` 73	Read-only. This is always **false** for group nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.
`Nodes` 74	Read-only. Returns a collection of `IGroupNode` and `ITableNode` objects, which represent groups of tables and tables respectively.

6.4.6.1 Properties

6.4.6.1.1 Text

The name of the group of tables as displayed in the EBA Filing Pane.

Signature

```
Text : String
```

6.4.6.1.2 IsGroup

This is always **true** for group nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.

Signature

```
IsGroup : Boolean
```

6.4.6.1.3 IsTable

This is always **false** for group nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.

Signature

```
IsTable : Boolean
```

6.4.6.1.4 Nodes

Returns a collection of `IGroupNode` and `ITableNode` objects, which represent groups of tables and tables respectively.

Signature

```
Nodes : Collection
```

6.4.7 ITableNode

The `ITableNode` interface provides information about a table.

Properties

Name	Description
`Text` 75	Read-only. The name of the table as displayed in the EBA Filing Pane.
`IsGroup` 75	Read-only. This is always **false** for table nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.
`IsTable` 76	Read-only. This is always **true** for table nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.
`Forms` 76	Read-only. Returns a collection of `IForm` objects that represent concrete forms that can be displayed as Excel worksheet.
`CanAddSubForm` 76	Read-only. This is **true** if additional sub forms can be added to the table, for example, if the table has open aspects on the z-axis. In most cases, this is a form for each country or currency.
`FilingIndicator` 76	Read-only. The filing indicator code of the table.

Name	Description
`RCCode` 77	Read-only. The RC code of the table.
`IncludeInFiling` 77	**True** if the table should be part of the report; **false** otherwise. If you set this property to **true** for the first time, a new worksheet for this table will be created.
`HasOpenXAxis` 77	**True** for tables with an open X-axis; **false** otherwise.
`HasOpenYAxis` 77	**True** for tables with an open Y-axis; **false** otherwise.

Methods

Name	Description
`AddSubForm` 78	Creates a new sub-form of this table and returns the respective `IForm` object. This method returns null if no sub-form can be added.

6.4.7.1 Properties

6.4.7.1.1 Text

The name of the table as displayed in the EBA Filing Pane.

Signature

```
Text : String
```

6.4.7.1.2 IsGroup

This is always **false** for table nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.

Signature

```
IsGroup : Boolean
```

6.4.7.1.3 IsTable

This is always **true** for table nodes. Use this to distinguish between `IGroupNode` and `ITableNode` members of the `Nodes` collection.

Signature

```
IsTable : Boolean
```

6.4.7.1.4 Forms

Returns a collection of `IForm` objects, which represent concrete forms that can be displayed as an Excel worksheet.

Signature

```
Forms : Collection
```

6.4.7.1.5 CanAddSubForm

This is **true** if additional sub forms can be added to the table, for example, if the table has open aspects on the z-axis. In most cases, this is a form for each country or currency.

Signature

```
CanAddSubForm : Boolean
```

6.4.7.1.6 FilingIndicator

The filing indicator code of the table.

Signature

```
FilingIndicator : String
```

6.4.7.1.7 RCCode

The RC code of the table.

Signature

```
RCCode : String
```

6.4.7.1.8 IncludeInFiling

True if the table should be part of the report; **false** otherwise. If you set this property to **true** for the first time, a new worksheet for this table will be created.

Signature

```
IncludeInFiling : Boolean
```

6.4.7.1.9 HasOpenXAxis

True for tables with an open X-axis; **false** otherwise.

Signature

```
HasOpenXAxis : Boolean
```

6.4.7.1.10 HasOpenYAxis

True for tables with an open Y-axis; **false** otherwise.

Signature

```
HasOpenYAxis : Boolean
```

6.4.7.2 Methods

6.4.7.2.1 AddSubForm

Creates a new sub-form of this table and returns the respective `IForm` object. This method returns null if no sub-form can be added.

Signature

```
AddSubForm() -> IForm
```

6.4.8 IForm

The `IForm` interface provides information about a form.

Properties

Name	Description
`Text` 79	Read-only. The name of the form as displayed in the EBA Filing Pane.
`DataRange` 79	Read-only. The Excel range containing the data of this form.
`FormSelectorRange` 79	Read-only. The Excel range containing the form selector of this form. Namely, the cells that contain the data that distinguishes this form from the other forms of the same table. This returns null if the table does not consist of multiple forms.
`Worksheet` 80	Read-only. The Excel worksheet containing this form. This may be null if `IncludeInFiling` is false.
`IncludeInFiling` 80	**True** if this form should be part of the report; **false** otherwise. This shows/hides the respective Excel worksheet.

Methods

Name	Description
`Remove` 80	Removes this form and deletes the respective Excel worksheet.
`AddRow` 80	Adds a row to the data range of tables with open y-axis. Returns the excel range of the new row.

Name	Description
AddColumn 81	Adds a column to the data range of tables with open x-axis. Returns the excel range of the new column.

6.4.8.1 Properties

6.4.8.1.1 Text

The name of the form as displayed in the EBA Filing Pane.

Signature

```
Text : String
```

6.4.8.1.2 DataRange

The Excel range containing the data of this form.

Signature

```
DataRange : Microsoft.Office.Interop.Excel.Range
```

6.4.8.1.3 FormSelectorRange

The Excel range containing the form selector of this form. Namely, the cells that contain the data that distinguishes this form from the other forms of the same table. This returns null if the table does not consist of multiple forms.

Signature

```
FormSelectorRange : Microsoft.Office.Interop.Excel.Range
```

6.4.8.1.4 Worksheet

The Excel worksheet containing this form. This may be null if `IncludeInFiling` is false.

Signature

```
Worksheet : Microsoft.Office.Interop.Excel.Worksheet
```

6.4.8.1.5 IncludeInFiling

True if this form should be part of the report; **false** otherwise. This shows/hides the respective Excel worksheet.

Signature

```
IncludeInFiling : Boolean
```

6.4.8.2 Methods

6.4.8.2.1 Remove

Removes this form and deletes the respective Excel worksheet.

Signature

```
Remove() -> Void
```

6.4.8.2.2 AddRow

Adds a row to the data range of tables with open y-axis. Returns the excel range of the new row.

Signature

```
AddRow() -> Microsoft.Office.Interop.Excel.Range
```

6.4.8.2.3 AddColumn

Adds a column to the data range of tables with open x-axis. Returns the excel range of the new column.

Signature

```
AddColumn() -> Microsoft.Office.Interop.Excel.Range
```

6.4.9 IReportProperties

The `IReportProperties` interface provides properties of the whole XBRL report.

Properties

Name	Description
`EntryPointURI` 82	Read-only. The URI of the taxonomy entry point. For example, "Additional Liquidity Monitoring - COREP, Consolidated".
`EntryPointModuleName` 82	Read-only. The module name of the taxonomy entry point. For example, "http://www.eba.europa.eu/eu/fr/xbrl/crr/fws/corep/cir-680-2014/2019-04-30/mod/corep_alm_con.xsd".
`EntryPointModuleDPMID` 82	Read-only. The Data Point Model Database ID of the module. For example, "218".
`ReportingEntityScheme` 83	The scheme of the reporting entity. For example, "http://standards.iso.org/iso/17442".
`ReportingEntityIdentifier` 83	The identifier of the reporting entity.
`ReferenceDate` 83	The reference date of the report.
`MonetaryCellsAccuracy` 83	The accuracy of monetary facts in this report. Applies to each monetary fact for which no separate accuracy was specified (at cell or table level).
`PercentageCellsAccuracy` 83	The accuracy of percentage facts in this report. Applies to each percentage fact for which no separate accuracy was specified (at cell or table level).
`PureCellsAccuracy` 84	The accuracy of pure facts in this report. Applies to each pure fact for which no separate accuracy was specified (at cell or table level).

Name	Description
ReportingCurrency 84	The reporting currency used in the XBRL report as an ISO 4217 currency code.
ReportingLanguage 84	The language of footnotes in the XBRL report as BCP-47 language tag. For example, "en-US".

6.4.9.1 Properties

6.4.9.1.1 EntryPointURI

The URI of the taxonomy entry point. For example, "Additional Liquidity Monitoring - COREP, Consolidated".

Signature

```
EntryPointURI : String
```

6.4.9.1.2 EntryPointModuleName

The module name of the taxonomy entry point. For example, "http://www.eba.europa.eu/eu/fr/xbrl/crr/fws/corep/cir-680-2014/2019-04-30/mod/corep_alm_con.xsd".

Signature

```
EntryPointModuleName : String
```

6.4.9.1.3 EntryPointModuleDPMID

The Data Point Model Database ID of the module. For example, "218".

Signature

```
EntryPointModuleDPMID : String
```

6.4.9.1.4 ReportingEntityScheme

The scheme of the reporting entity. For example, "http://standards.iso.org/iso/17442".

Signature

```
ReportingEntityScheme : String
```

6.4.9.1.5 ReportingEntityIdentifier

The identifier of the reporting entity.

Signature

```
ReportingEntityIdentifier : String
```

6.4.9.1.6 ReferenceDate

The reference date of the report.

Signature

```
ReferenceDate : DateTime
```

6.4.9.1.7 MonetaryCellsAccuracy

The accuracy of monetary facts in this report. Applies to each monetary fact for which no separate accuracy was specified (at cell or table level).

Signature

```
MonetaryCellsAccuracy : String
```

6.4.9.1.8 PercentageCellsAccuracy

The accuracy of percentage facts in this report. Applies to each percentage fact for which no separate accuracy was specified (at cell or table level).

Signature

```
PercentageCellsAccuracy : String
```

6.4.9.1.9 PureCellsAccuracy

The accuracy of pure facts in this report. Applies to each pure fact for which no separate accuracy was specified (at cell or table level).

Signature

```
PureCellsAccuracy : String
```

6.4.9.1.10 ReportingCurrency

The reporting currency used in the XBRL report as an ISO 4217 currency code.

Signature

```
ReportingCurrency : String
```

6.4.9.1.11 ReportingLanguage

The language of footnotes in the XBRL report as BCP-47 language tag. For example, "en-US".

Signature

```
ReportingLanguage : String
```

6.4.10 IFormProperties

The `IFormProperties` interface provides properties of one form of the report.

Properties

Name	Description
`MonetaryCellsAccuracy` 85	The accuracy of monetary facts in this form. Applies to each monetary fact for which no separate accuracy was specified.
`PercentageCellsAccuracy` 86	The accuracy of percentage facts in this form. Applies to each percentage fact for which no separate accuracy was specified.
`PureCellsAccuracy` 86	The accuracy of pure facts in this form. Applies to each pure fact for which no separate accuracy was specified.
`TableRCCode` 86	Read-only. The RC Code of the table. For example, "C 66.01.a".
`FilingIndicatorCode` 86	Read-only. The filing indicator code of the table. For example, "C_66.01".
`Label` 87	Read-only. The label of the table. For example, "C 66.01.a".
`VerboseLabel` 87	Read-only. The verbose label of the table. For example, "C 66.01.a Maturity ladder. Total. Overnight and higher maturity".
`TableID` 87	Read-only. The id of the table resource. For example, "eba_tC_66.01.a".
`TableDPMID` 87	Read-only. The Data Point Model Database ID of the table. For example, "429.1431".
`ValidationRules` 88	Read-only. The collection of validation rules that apply to this table.

6.4.10.1 Properties

6.4.10.1.1 MonetaryCellsAccuracy

The accuracy of monetary facts in this form. Applies to each monetary fact for which no separate accuracy was specified.

Signature

```
MonetaryCellsAccuracy : String
```

6.4.10.1.2 PercentageCellsAccuracy

The accuracy of percentage facts in this form. Applies to each percentage fact for which no separate accuracy was specified.

Signature

```
PercentageCellsAccuracy : String
```

6.4.10.1.3 PureCellsAccuracy

The accuracy of pure facts in this form. Applies to each pure fact for which no separate accuracy was specified.

Signature

```
PureCellsAccuracy : String
```

6.4.10.1.4 TableRCCode

The RC Code of the table. For example, "C 66.01.a".

Signature

```
TableRCCode : String
```

6.4.10.1.5 FilingIndicatorCode

The filing indicator code of the table. For example, "C_66.01".

Signature

```
FilingIndicatorCode : String
```

6.4.10.1.6 Label

The label of the table. For example, "C 66.01.a".

Signature

```
Label : String
```

6.4.10.1.7 VerboseLabel

The verbose label of the table. For example, "C 66.01.a Maturity ladder. Total. Overnight and higher maturity".

Signature

```
VerboseLabel : String
```

6.4.10.1.8 TableID

The id of the table resource. For example, "eba_tC_66.01.a".

Signature

```
TableID : String
```

6.4.10.1.9 TableDPMID

The Data Point Model Database ID of the table. For example, "429.1431".

Signature

```
TableDPMID : String
```

6.4.10.1.10 ValidationRules

The collection of validation rules that apply to this table.

Signature

```
ValidationRules : Collection
```

6.4.11 ICellProperties

The `ICellProperties` interface shows properties of one fact of the report.

Properties

Name	Description
Accuracy 89	The accuracy of the numeric item as string. This can be "INF" or a number representing the number of decimal places for which this fact is accurate. This is null for non-numeric items.
Footnote 89	The footnote of the fact.
Name 89	Read-only. The concept name of the fact. For example, "eba_met:mi256".
Type 89	Read-only. The type of the fact. For example, "xbrli:monetaryItemType".
Label 90	Read-only. The label of the fact. For example, "Cash value".
DPMID 90	Read-only. The Data Point Model Database ID of this fact. For example, "6062".
Dimensions 90	Read-only. The collection of `IDimension` objects representing the dimensions for which this fact is reported.

6.4.11.1 Properties

6.4.11.1.1 Accuracy

The accuracy of the numeric item as string. This may be "INF" or a number representing the number of decimal places for which this fact is accurate.

> This is null for non-numeric items.

Signature

```
Accuracy : String
```

6.4.11.1.2 Footnote

The footnote of the fact.

Signature

```
Footnote : String
```

6.4.11.1.3 Name

The concept name of the fact. For example, "eba_met:mi256".

Signature

```
Name : String
```

6.4.11.1.4 Type

The type of the fact. For example, "xbrli:monetaryItemType".

Signature

```
Type : String
```

6.4.11.1.5 Label

The label of the fact. For example, "Cash value".

Signature

```
Label : String
```

6.4.11.1.6 DPMID

The Data Point Model Database ID of this fact. For example, "6062".

Signature

```
DPMID : String
```

6.4.11.1.7 Dimensions

The collection of `IDimension` objects representing the dimensions for which this fact is reported.

Signature

```
Dimensions : Collection
```

6.4.12 IDimension

The `IDimension` interface provides basic information of a `Dimension` and its value.

Properties

Name	Description
`Name` 91	Read-only. The name of the dimension.
`Value` 91	Read-only. The value of the dimension as `String`.

6.4.12.1 Properties

6.4.12.1.1 Name

The name of the dimension.

Signature

```
Name : String
```

6.4.12.1.2 Value

The value of the dimension as `String`.

Signature

```
Value : String
```

6.4.13 IValidationReport

The `IValidationReport` interface provides access to the validation report.

Properties

Name	Description
`Messages` 92	Read-only. Returns a collection of `IValidationReportMessage` representing the main lines of the validation report.

Methods

Name	Description
`CreateTextReport` 92	Returns a textual representation of the validation report.
`CreateHTMLReport` 92	Returns an HTML representation of the validation report.

6.4.13.1 Properties

6.4.13.1.1 Messages

Returns a collection of `IValidationReportMessage` representing the main lines of the validation report.

Signature

```
Messages : Collection
```

6.4.13.2 Methods

6.4.13.2.1 CreateTextReport

Returns a textual representation of the validation report.

Signature

```
CreateTextReport() -> String
```

6.4.13.2.2 CreateHTMLReport

Returns an HTML representation of the validation report.

Signature

```
CreateHTMLReport() -> String
```

6.4.14 IValidationReportMessage

The `IValidationReportMessage` interface represents a main line in the validation report.

Properties

Name	Description
`Severity` 93	Read-only.

Name	Description
	Returns the severity of this report message as `String`. Possible values are: "success", "info", "warning" and "error".
`Text` 93	Read-only. Returns the value of this report message.
`Details` 93	Read-only. Returns the details of this report message (if any) as `String`.

6.4.14.1 Properties

6.4.14.1.1 Severity

Returns the severity of this report message as `String`. Possible values are: "success", "info", "warning" and "error".

Signature

```
Severity : String
```

6.4.14.1.2 Text

Returns the value of this report message.

Signature

```
Text : String
```

6.4.14.1.3 Details

Returns the details of this report message (if any) as `String`.

Signature

```
Details : String
```

7 License Information

This section contains information about:

- the distribution of this software product
- software activation and license metering
- the license agreement governing the use of this product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

To view the terms of any Altova license, go to the Altova Legal Information page at the Altova website.

7.1 Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge for 30 days before making a purchasing decision. *(Note: Altova MobileTogether Designer is licensed free of charge.)*
- Once you decide to buy the software, you can place your order online at the Altova website and get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes an onscreen help system that can be accessed from within the application interface. The latest version of the user manual is available at www.altova.com in (i) HTML format for online browsing, and (ii) PDF format for download (and to print if you prefer to have the documentation on paper).

30-day evaluation period

After downloading this product, you can evaluate it for a period of up to 30 days free of charge. About 20 days into the evaluation period, the software will start to remind you that it has not yet been licensed. The reminder message will be displayed once each time you start the application. If you would like to continue using the program after the 30-day evaluation period, you must purchase a product license, which is delivered in the form of a license file containing a key code. Unlock the product by uploading the license file in the Software Activation dialog of your product.

You can purchase product licenses at https://shop.altova.com/.

Helping Others within Your Organization to Evaluate the Software

If you wish to distribute the evaluation version within your company network, or if you plan to use it on a PC that is not connected to the Internet, you may distribute only the installer file, provided that this file is not modified in any way. Any person who accesses the software installer that you have provided must request their own 30-day evaluation license key code and after expiration of their evaluation period, must also purchase a license in order to be able to continue using the product.

7.2 Software Activation and License Metering

As part of Altova's Software Activation, the software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the software and to improve customer service. Activation is based on the exchange of license related data such as operating system, IP address, date/time, software version, and computer name, along with other information between your computer and an Altova license server.

Your Altova product has a built-in license metering module that further helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

Single license

When the application starts up, as part of the license metering process, the software sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application.

Multi-user license

If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to help you to better determine that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses.

We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (*see the IANA Service Name Registry for details*) and our license-metering module is tested and proven technology.

If you are using a firewall, you may notice communications on port 2799 between the computers that are running Altova products. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

Note about certificates

Your Altova application contacts the Altova licensing server (`link.altova.com`) via HTTPS. For this communication, Altova uses a registered SSL certificate. If this certificate is replaced (for example, by your IT department or an external agency), then your Altova application will warn you about the connection being insecure. You could use the replacement certificate to start your Altova application, but you would be doing this at your own risk. If you see a *Non-secure connection* warning message, check the origin of the certificate and consult your IT team (who would be able to decide whether the interception and replacement of the Altova certificate should continue or not).

If your organization needs to use its own certificate (for example, to monitor communication to and from client machines), then we recommend that you install Altova's free license management software, Altova LicenseServer, on your network. Under this setup, client machines can continue to use your organization's certificates, while Altova LicenseServer can be allowed to use the Altova certificate for communication with Altova.

7.3 Altova End-User License Agreement

- The Altova End-User License Agreement is available here: https://www.altova.com/legal/eula
- Altova's Privacy Policy is available here: https://www.altova.com/privacy

Index