Unified Framework for Driving Transformations

Labnaf Language Transformer

User Guide

See also the "**Labnaf Language Transformer - Reference Guide**"

WARNING

NEVER use the language transformer on your production repository before performing all necessary tests.

ALWAYS test your language transformer commands using a repository backup.

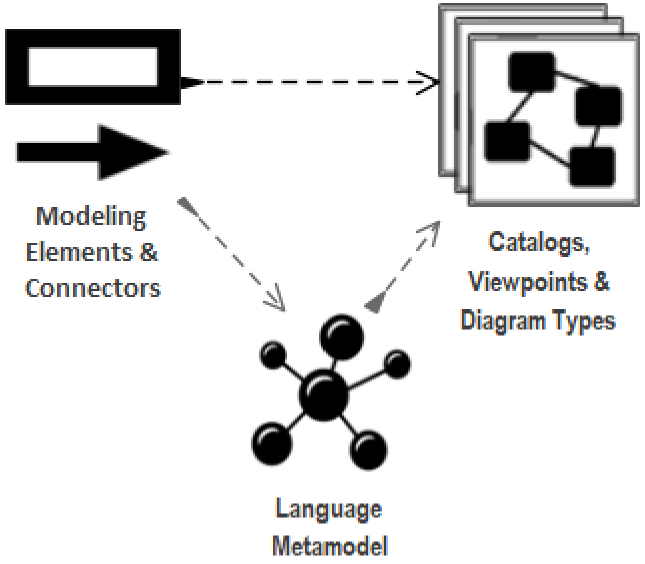
ALWAYS carefully check the resulting transformations and possible side effects. For example items could be deleted because you misspelled a type.

ALWAYS remember that type and stereotype names are case sensitive.

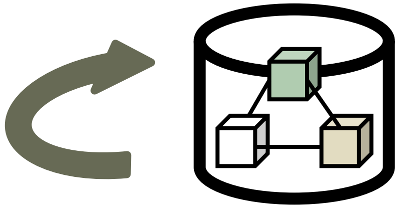
The Labnaf Language Transformer has been tested using Sparx Systems' Enterprise Architect versions 13.5 to 16.1

Labnaf Customization Steps

1. Customize the language following your organization requirements

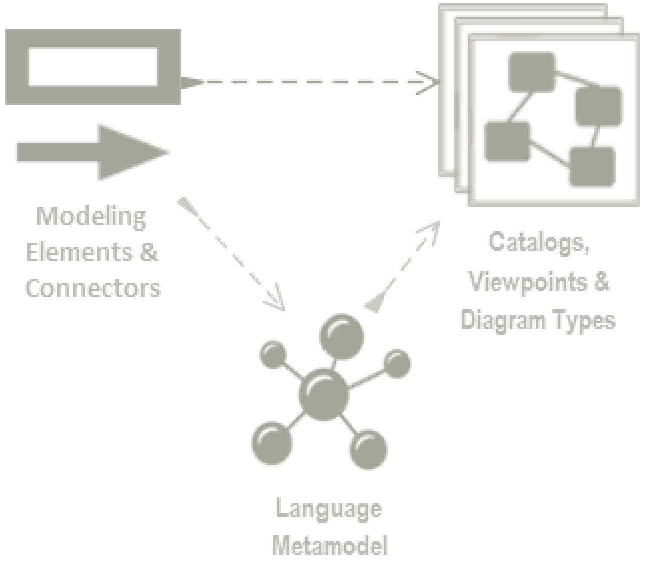

Modeling Elements & Connectors
Catalogs, Viewpoints & Diagram Types
Language Metamodel

2. Adapt existing repository content

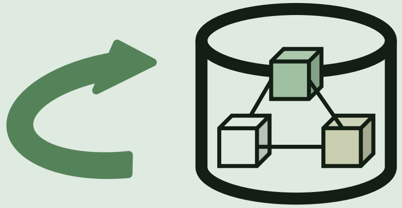


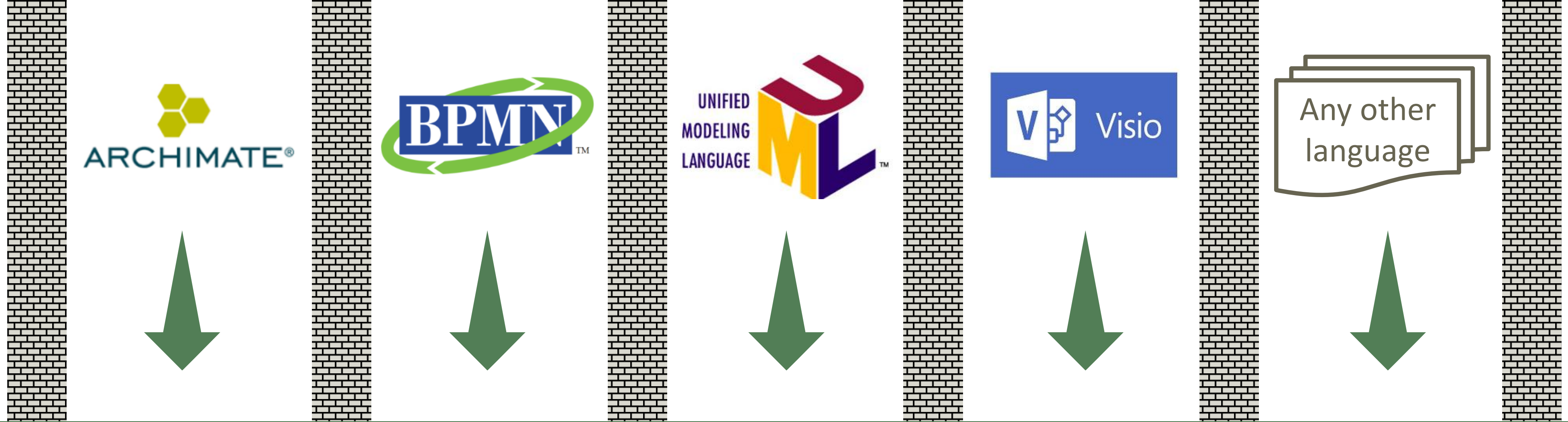
Labnaf Language Transformer

1. Customize the language

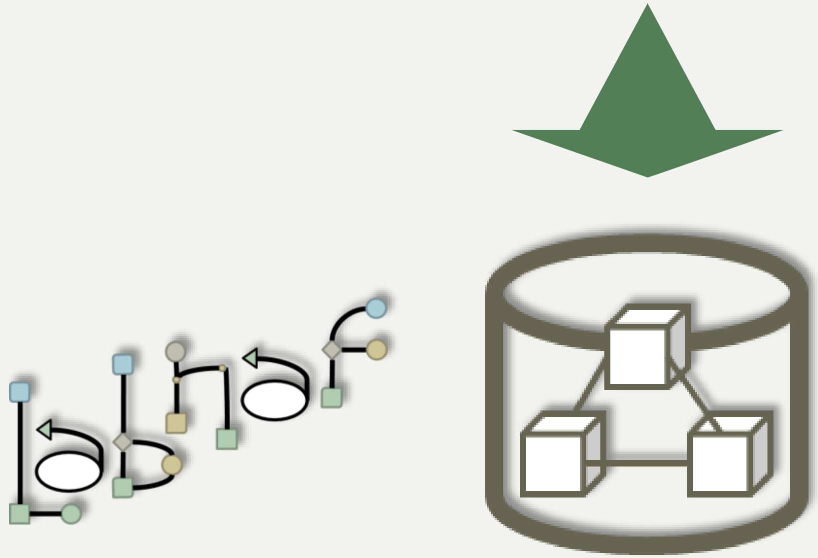

Modeling Elements & Connectors
Catalogs, Viewpoints & Diagram Types
Language Metamodel

2. **Adapt existing repository content**

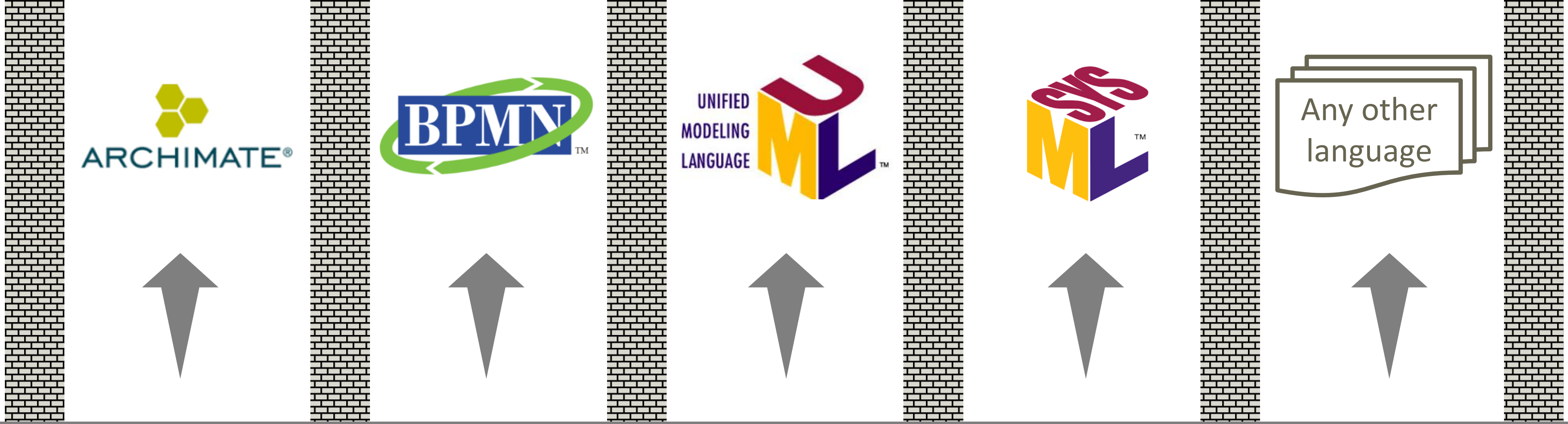




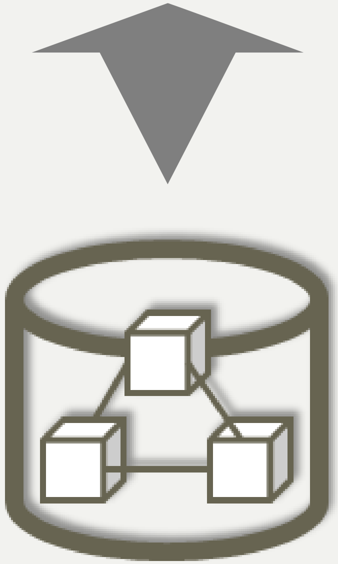
Transform any language…



We all speak the same language



… in any direction

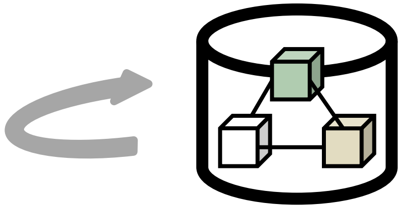


One **Single**
Strategy & Architecture
Modeling **Language**

You are never locked in Labnaf !

The Language Transformer adapts the language in existing repository content

- **ChangeElementType**
- **ChangeConnectorType**
- **ChangeDiagramType**
- **ChangeDiagramTypesDefinedInCsv**
- **TvRename**
- **TvDelete**



See the **Language Transformer Reference Guide** for command specific information.

Prerequisites

Install the Language Transformer using the

Installer/LabnafLanguageTransformerSetup.msi

Unzip the sample language transformations files stored in

Doc/SampleLanguageTransformations.zip

Development Lifecycle

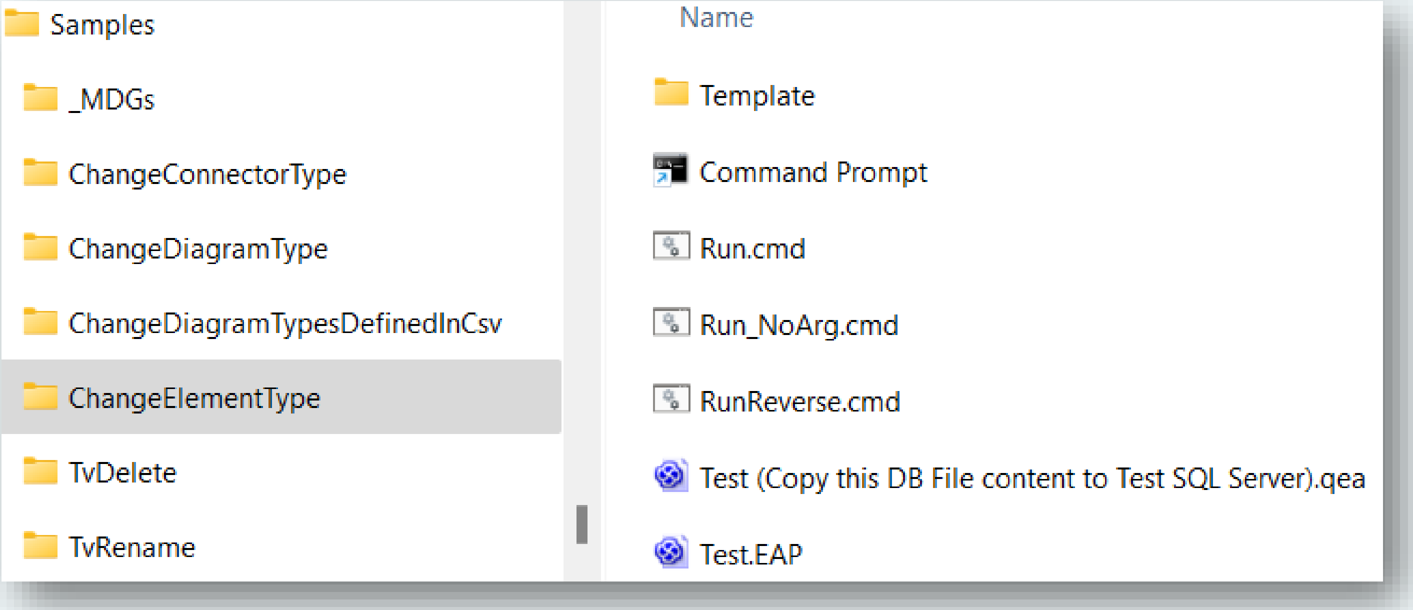
- To transform a language in a repository, you will need a combination of transformer commands.
- We advise to address each command separately and in that order:
  - **ChangeElementType**
  - **ChangeConnectorType**
  - **ChangeDiagramType or ChangeDiagramTypesDefinedInCsv**
  - **TvRename**
  - **TvDelete**

The development lifecycle is the same for each command.

Keep backups of your different versions.

Transforming Element, Connector and Diagram Types

Review the content of the Sample language transformations folder that you unzipped



Samples
- _MDGs
- ChangeConnectorType
- ChangeDiagramType
- ChangeDiagramTypesDefinedInCsv
- ChangeElementType
- TvDelete
- TvRename

Name
- Template
- Command Prompt
- Run.cmd
- Run_NoArg.cmd
- RunReverse.cmd
- Test (Copy this DB File content to Test SQL Server).qea
- Test.EAP

For each Language Transformer command, there is a sample folder with an example.
Folder names are language transformer command names.

The folders have a common structure. The file names are about the same.
The content of the “Run.cmd” and “Test (Copy this DB File content to Test SQL Server).qea” files are specific to each folder/command.

Develop and Test Your **Type Transformation Script**

Edit Test.eap using Notepad (not Sparx EA)

- Change the connection string to SQL server DB for testing language transformations
- Copy the content of “Test (Copy this DB File content to Test SQL Server).qea” to your SQL Server Test database. See **how**.

Edit and run the test script

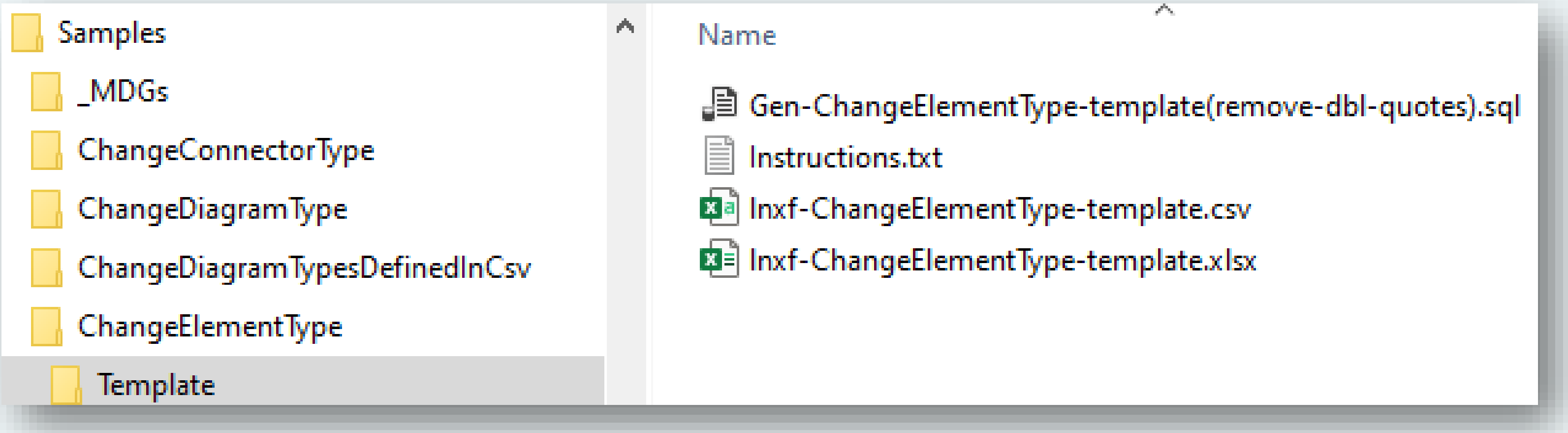
- Edit **Run.cmd.** Put your own transformation commands
- Execute **Run.cmd**

See the language transformation results

- Open **Test.eap**
- **In Sparx EA**

Keep backups of your different versions.

Use the Excel Templates to define your mappings

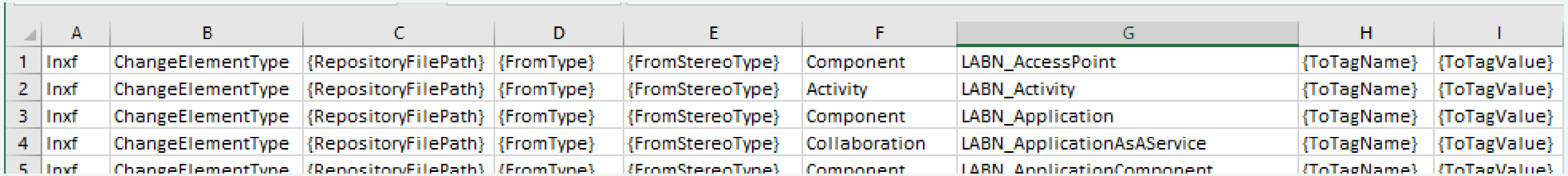


- Samples
  - _MDGs
  - ChangeConnectorType
  - ChangeDiagramType
  - ChangeDiagramTypesDefinedInCsv
  - ChangeElementType
    - Template

Name

- Gen-ChangeElementType-template(remove-dbl-quotes).sql
- Instructions.txt
- Inxf-ChangeElementType-template.csv
- Inxf-ChangeElementType-template.xlsx

- Map source to target types and stereotypes in Excel



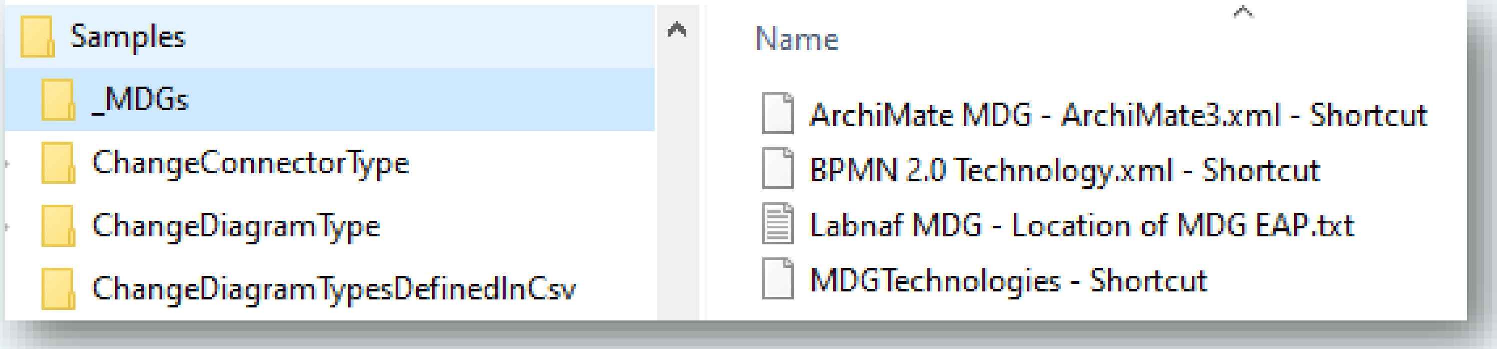
	A	B	C	D	E	F	G	H	I
1	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_AccessPoint	{ToTagName}	{ToTagValue}
2	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Activity	LABN_Activity	{ToTagName}	{ToTagValue}
3	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_Application	{ToTagName}	{ToTagValue}
4	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Collaboration	LABN_ApplicationAsAService	{ToTagName}	{ToTagValue}
5	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_ApplicationComponent	{ToTagName}	{ToTagValue}

- Create your transformation commands based on the Excel

Development Lifecycle

To find out what types and stereotypes are

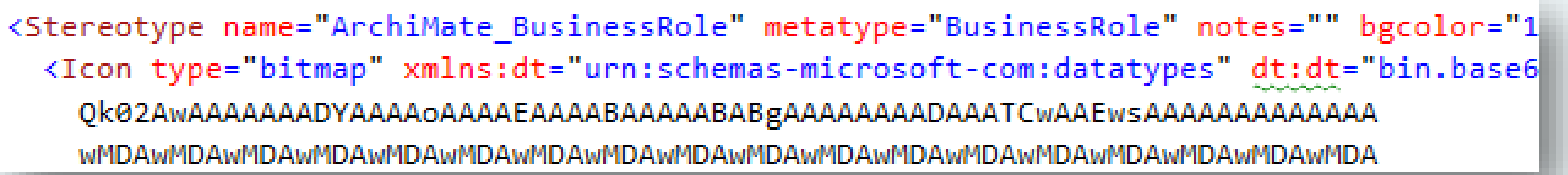
- Look inside the MDG XML files



Samples
_MDGs
ChangeConnectorType
ChangeDiagramType
ChangeDiagramTypesDefinedInCsv

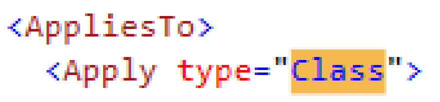
Name
ArchiMate MDG - ArchiMate3.xml - Shortcut
BPMN 2.0 Technology.xml - Shortcut
Labnaf MDG - Location of MDG EAP.txt
MDGTechnologies - Shortcut

- Search for **Stereotype name=“*{stereotype I am looking for}*”**
  Or **metatype=“*{metatype I am looking for}*”**



```
<Stereotype name="ArchiMate_BusinessRole" metatype="BusinessRole" notes="" bgcolor="1
  <Icon type="bitmap" xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="bin.base6
    Qk02AwAAAAAAADYAAAAoAAAAEAAAABAAAAABABgAAAAAAAADAAATCwAAEwsAAAAAAAAAAAAA
    wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDA
```

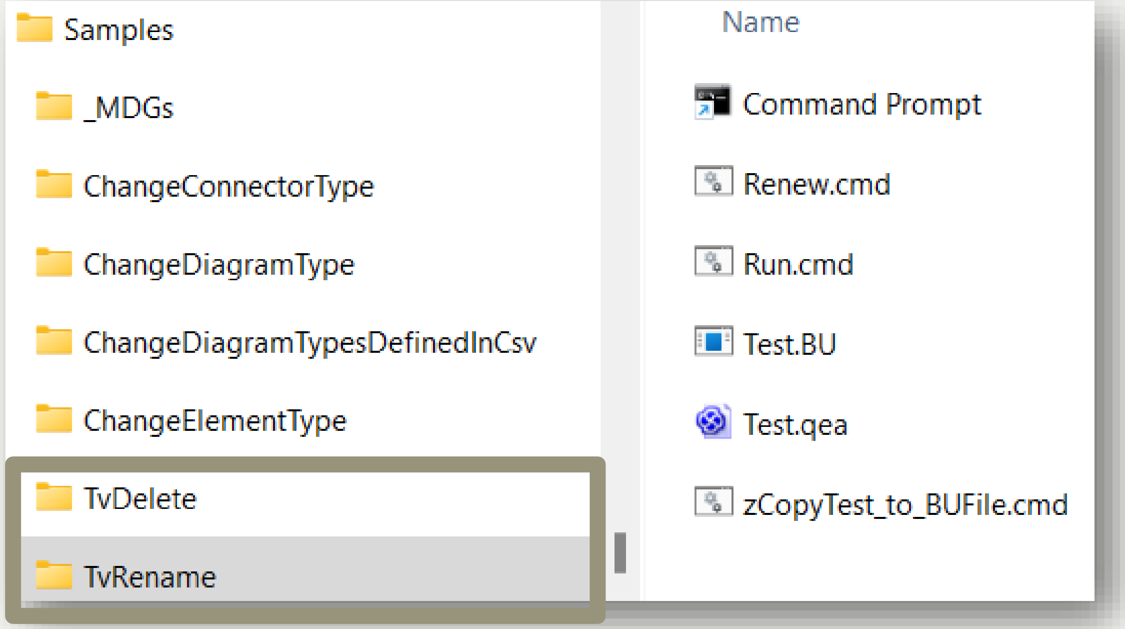
- See the extended type



```
<AppliesTo>
  <Apply type="Class">
```

Renaming or Deleting Tagged Values

Review the content of the Sample language transformations folder that you unzipped

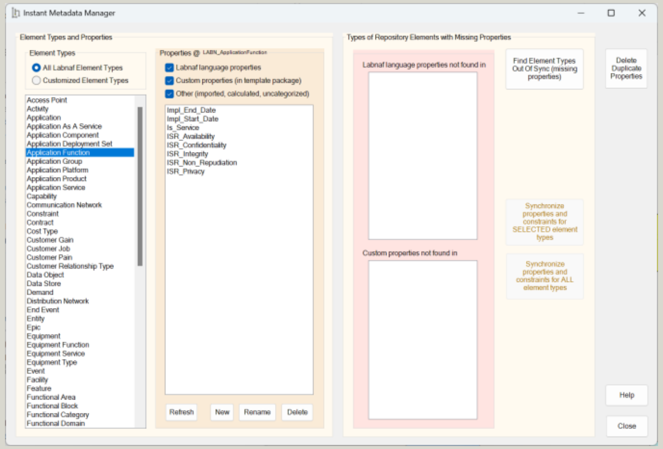


Samples
- _MDGs
- ChangeConnectorType
- ChangeDiagramType
- ChangeDiagramTypesDefinedInCsv
- ChangeElementType
- TvDelete
- TvRename

Name
Command Prompt
Renew.cmd
Run.cmd
Test.BU
Test.qea
zCopyTest_to_BUFile.cmd

Interactive Alternative

- Use the Labnaf Instant Data Manager



Instant Metadata Manager

Element Types and Properties

Element Types
- ● All Labnaf Element Types
- ○ Customized Element Types

Access Point
Activity
Application
Application As A Service
Application Component
Application Deployment Set
Application Function
Application Group
Application Platform
Application Product
Application Service
Capability
Communication Network
Constraint
Contract
Cost Type
Customer Gain
Customer Job
Customer Pain
Customer Relationship Type
Data Object
Data Store
Demand
Distribution Network
End Event
Entity
Epic
Equipment
Equipment Function
Equipment Service
Equipment Type
Event
Facility
Feature
Functional Area
Functional Block
Functional Category
Functional Domain

Properties @ LABN_ApplicationFunction
- ☑ Labnaf language properties
- ☑ Custom properties (in template package)
- ☑ Other (imported, calculated, uncategorized)

Impl_End_Date
Impl_Start_Date
Is_Service
ISR_Availability
ISR_Confidentiality
ISR_Integrity
ISR_Non_Repudiation
ISR_Privacy

Refresh
New
Rename
Delete
Types of Repository Elements with Missing Properties
Labnaf language properties not found in
Custom properties not found in
Find Element Types Out Of Sync (missing properties)
Synchronize properties and constraints for SELECTED element types
Synchronize properties and constraints for ALL element types
Delete Duplicate Properties
Help
Close

For each Language Transformer command, there is a sample folder with an example.
Folder names are language transformer command names.

The folders have a common structure. The file names are about the same.
The content of the "Run.cmd" and "Test.qea" files are specific to each folder/command.

Develop and Test Your **Tagged Value Management Script**

Adapt the test data as needed

- Execute **Renew.cmd**
  - to copy the backup database into a fresh **Test.qea**

- Open and edit **Test.qea.**
  - Replace the content by you own test data
  - Close **Test.qea**

- Execute "**zCopyTest_to_BUFile.cmd**"
  - to create a new backup database from **Test.qea.** So you will then be able to Renew the test database with your own data before running each test.

Keep backups of your different versions.

Development Lifecycle

Test on real data

- Backup your PROD repository into an file-based repository (QEA).
- Replace “Test.qea” by that backup repository.
- Execute the transformation script “Run.cmd” and see the result

Run your transformation script on the PROD repository

- Ensure nobody is working on the repository
- Make a backup of you PROD repository
- Replace “Test.eap” by a shortcut to your PROD repository.
- Execute the transformation script “Run.cmd” and see the results

Keep backups of your different versions.