A Language Built in its Native Framework

The Labnaf Strategy & Architecture Framework

Labnaf Language Transformer

Reference Guide

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

WARNING

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

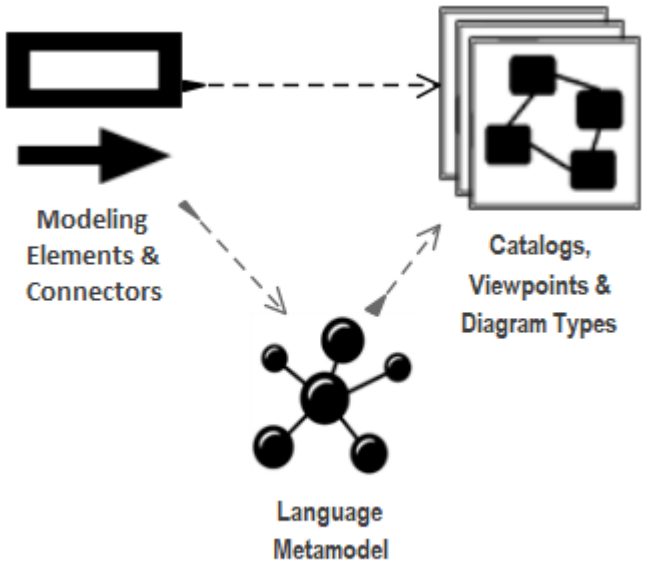
ALWAYS test your language transformer commands using a repository backup.

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

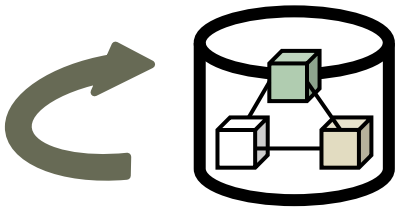
ALWAYS remember that type and stereotype names are case sensitive.

Labnaf Customization Steps

1. Customize the language following your organization requirements

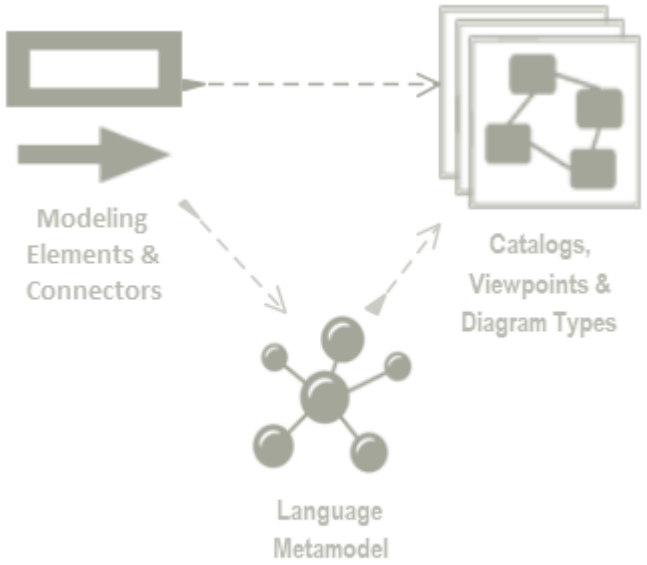


2. Adapt existing repository content

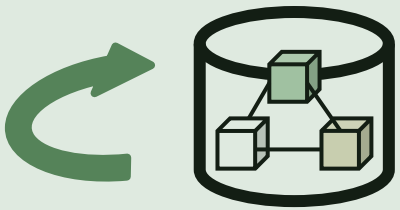


Labnaf Language Transformer

1. Customize the language

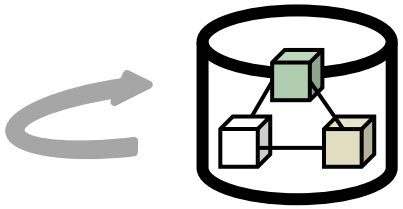


2. Adapt existing repository content



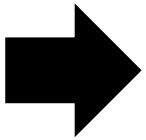
The Language Transformer adapts the language in existing repository content

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



lnxf usage

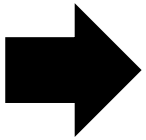
lnxf



```
Usage : lnxf Command [arguments]

Available Commands:

   ChangeElementType

   ChangeConnectorType

   ChangeDiagramType

   ChangeDiagramTypesDefinedInCsv

   SynchronizeStereotypesExtended

   TemplateMetamodelFromActiveMetamodel

   TvRename

   TvDelete
```

lnxf {command name}

Example: if you type « lnxf TvRename » you get the following info:



```
Command: TvRename

Description: Rename tag.

Usage : lnxf TvRename [arguments]

Arguments:

   RepoPathName: Model repository path name.

   FromTagName: Name of the tag to be renamed.

   ToTagName: New tag name.

   [ElementType]: Restrict the scope to this element type.

   [ElementStereotype]: Restrict the scope to this element stereotype.
```

ChangeElementType

To change the type and/or the stereotype of existing elements in a repository:

```
Command: ChangeElementType

Description: Change element types and stereotypes in a model repository.

Usage : lnxf ChangeElementType [arguments]

Arguments:

   SourceRepoPathName: Path name of the source model repository where the elements must be changed.

   FromType: The element type to be changed OR '*' if the type is not a selection criteria.

   FromStereotype: The element stereotype to be changed OR '-' for changing elements without a stereotype.

   ToType: The new element type to be set for each element.

   ToStereotype: The new element stereotype to be set for each element.

   [ToTagName]: The name of the tag to be added to each element.

   [ToTagValue]: The tag value for that named tag.
```

After the type and or stereotype has been changed, you need to

A. Reload the project (or close and reopen the repository)

B. Update the icons in the project browser as follows:

1. **Create an empty diagram of any type**
2. **Drop the changed elements on the diagram**
   - either from the project browser or
   - or from the result of a query. For example if you transformed into applications: select ea_guid AS CLASSGUID, Object_Type AS CLASSTYPE, Name from **t_object** where Stereotype = 'LABN_Application'

ChangeConnectorType

To change the type and/or the stereotype of existing connectors in a repository:

```
Command: ChangeConnectorType

Description: Change connector types and stereotypes in a model repository following criteria.

Usage : lnxf ChangeConnectorType [arguments]

Arguments:

   /all|/selective: Change all connectors of a certain type and stereotype or only a selected subset based on source and destination types and stereotypes.

   /samedirection|/reversedirection: Keep or reverse the connector direction.
If the connector direction is reversed, please perform a repository integrity check to finalise updates.

   SourceRepoPathName: Path name of the source model repository where the connectors must be changed.

   FromType: The connector type to be changed OR '*' if the type is not a selection criteria.

   FromStereotype: The connector stereotype to be changed OR '-' for changing connectors without a stereotype.

   ToType: The new connector type to be set for each selected connector.

   ToStereotype: The new connector stereotype to be set for each selected connector.

   [SrcElmType]: Change the connecor type only when the connector's source element is of this type.

   [SrcElmStereotype]: Change the connecor type only when the connector's source element is of this stereotype.

   [DestElmType]: Change the connecor type only when the connector's destination element is of this type.

   [DestElmStereotype]: Change the connecor type only when the connector's destination element is of this stereotype.
```

ChangeDiagramType

To change the type and/or the stereotype of existing diagrams in a repository:

```
Command: ChangeDiagramType

Description: Change diagram types in a model repository.

Usage : lnxf ChangeDiagramType [arguments]

Arguments:

   SourceRepoPathName: Path name of the source model repository where the diagrams must be changed.

   FromType: The diagram type to be selected OR '*' if the type is not a selection criteria.

   FromStereotype: The diagram stereotype to be selected.

   ToType: The new diagram type to be set for each selected diagram.

   ToStereotype: The new diagram stereotype to be set for each selected diagram.
```

ChangeDiagramTypesDefinedInCSV

To change multiple types and/or stereotypes of existing diagrams in a repository:

```
Command: ChangeDiagramTypesDefinedInCsv

Description: Change diagram types in a model repository based on a CSV configuration file.
Wild cards '*' and '-' used in command ChangeDiagramType are applicable.

Usage : lnxf ChangeDiagramTypesDefinedInCsv [arguments]

Arguments:

   SourceRepoPathName: Path name of the source model repository where the diagrams must be changed.

   CsvConfigPathName: Path name of the CSV configuration file.
     The first row in the CSV file is fixed and contains the field headers:
     FromType,FromStereotype,ToType,ToStereotype

   [CsvFieldDelimiter]: The field delimiter in the CSV configuration file (default is ',')
```

Sample input CSV

	A	B	C	D
1	**FromType**	**FromStereotype**	**ToType**	**ToStereotype**
2	Logical	LABN::Corporate Strategy Map	Logical	Labnaf - Vision::Corporate Strategy Map
3	Logical	LABN::Goals	Logical	Labnaf - Vision::Goals
4	Logical	LABN::Principles	Logical	Labnaf - Vision::Principles
5	Logical	LABN::Standards (as rules)	Logical	Labnaf - Vision::Standards (as rules)
6	Logical	LABN::Demands	Logical	Labnaf - Vision::Demands
7	Logical	LABN::High Level Requirements Roadmap	Logical	Labnaf - Vision::High Level Requirements Roadmap

SynchronizeStereotypesExtended

Performs extended element stereotype synchronization following **template elements store in a template package** and which are used as reference.

The following element feature items will be synchronized with MDG definition

- Default child diagram type
- Tag grouping
- Default tag group state (open/closed)

Prerequisite

Your repository contains a template package with template elements.
If you don't have such package, you can either add the package and new elements by hand or use the command "**TemplateMetamodelFromActiveMetamodel"**.

To perform the stereotype synchronization that Sparx System's EA does not do:

```
Command: SynchronizeStereotypesExtended

Description: Extended element stereotype synchronization following template elements store in a template package and which are used as reference.
What is being synchronized is: Default diagram type, tag grouping and default tag group state (open/closed)

Usage : lnxf SynchronizeStereotypesExtended [arguments]

Arguments:

   SourceRepoPathName: Path name of the source model repository where the diagrams must be changed.

   GuidOftemplateElementsPackage: The GUID of a package containing well formed elements to be used as templates.
```

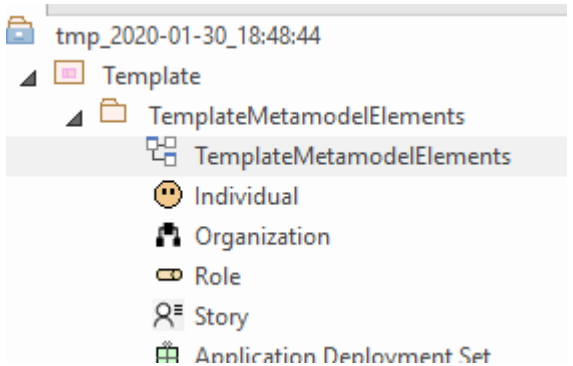
TemplateMetamodelFromActiveMetamodel

Creates a template metamodel package for building your own metamodel from scratch. What you get is a new package with new elements of the same type and stereotype as in the current metamodel but without any connection.
Prerequisite: The original reference metamodel must exist in the repository.

To easily create the new metamodel from scratch:

```
Command: TemplateMetamodelFromActiveMetamodel

Description: Create a template metamodel package for building you own metamodel from scratch.


Usage : lnxf TemplateMetamodelFromActiveMetamodel [arguments]

Arguments:

   RepoPathName: Model repository path name.
```

When the process is completed, a package contains the list of metamodel elements. **They will appear like UML elements.**



Next steps:

1) Create a new diagram of any type
2) Drag and drop all the elements onto that diagram =>

TvRename

To rename a tagged value for existing elements in a repository:

```
Command: TvRename

Description: Rename tag.


Usage : lnxf TvRename [arguments]

Arguments:

   RepoPathName: Model repository path name.

   FromTagName: Name of the tag to be renamed.

   ToTagName: New tag name.

   [ElementType]: Restrict the scope to this element type.

   [ElementStereotype]: Restrict the scope to this element stereotype.
```

TvDelete

To delete a tagged value for existing elements in a repository:

```
Command: TvDelete

Description: Delete tag.


Usage : lnxf TvDelete [arguments]

Arguments:

   RepoPathName: Model repository path name.

   TagName: Name of the tag to be deleted.

   [ElementType]: Restrict the scope to this element type.

   [ElementStereotype]: Restrict the scope to this element stereotype.
```