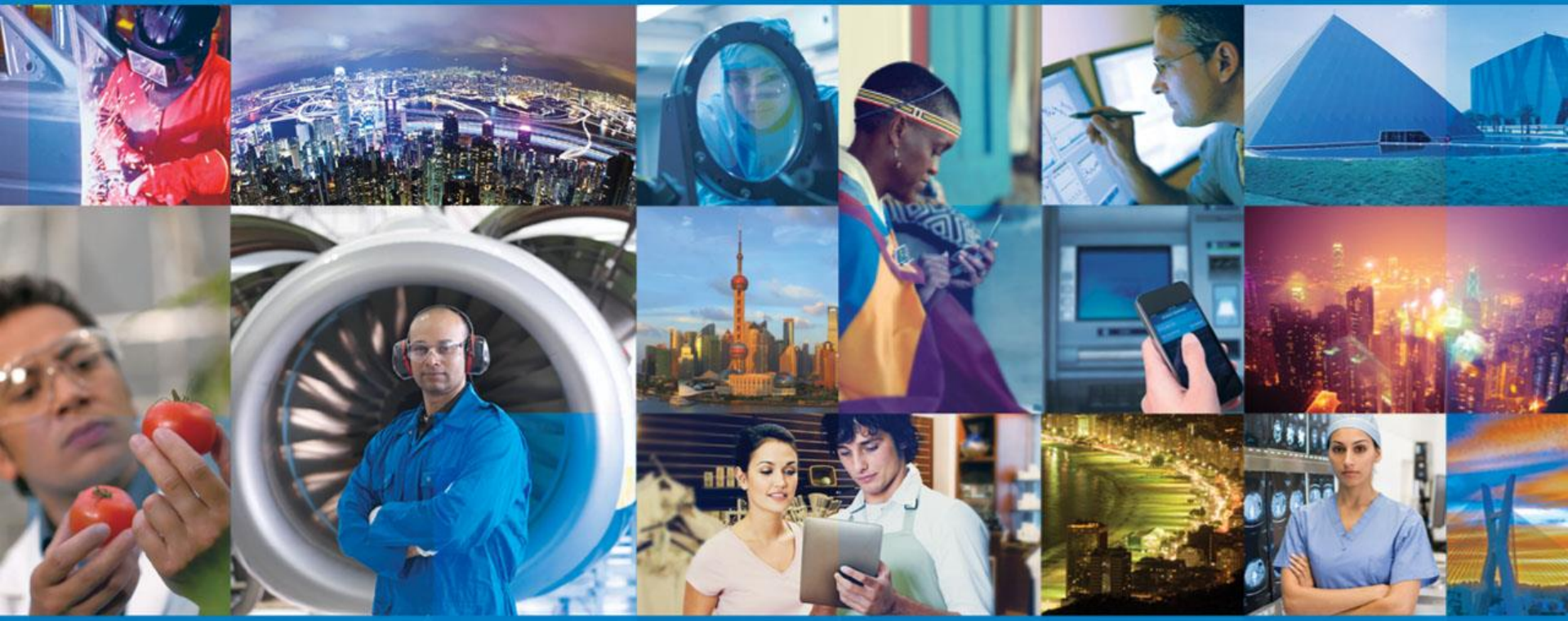
FormsGroup Presentation



Person’s Name: Pushparaj Kamaraj
Name of Meeting: FormsGroup
Date and Year: August 2014

Infosys® | Finacle
Building Tomorrow’s Enterprise

Module Objectives

At the end of this module, you will be able to:

- Explain the architectural overview and goals in the web layer
- Explain the structure of FormsGroup (FG)
- Utilise the FGIs supported by FG Generator Tool
- Demonstrate the use of UI controls supported by FG
- Explain customisation and extension in FG

Module Overview

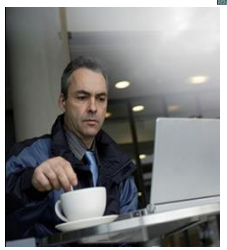
Welcome to the module on **FormsGroup (FG)**. The module defines FG, covers details about its architectural overview and goals in the web layer, and details out the various components of the FG structure. This is followed by the FG Control Instructions, and examples about customisations in FG.

Pre-requisites:

Before you take this module, you should be familiar with:

- Type systems
- UI Layer
- Basic Java

FormsGroup



FormsGroup: Definition

- Formsgroup (FG) is a component which handles a set/group of related user actions initiated in a UI.
- FG is a logical set of web actions on a related set of data.
- In the context of earlier Finacle e-Banking paradigm, this can be defined as a set of Actions and corresponding Commands.

FormsGroup: Architecture Goals

The Architecture Goals of FG in the Web Layer are:

Separation of Model and View	Formsgroup helps to achieve Model-View separation so that View can be changed independently.
Inversion of Control	In the context of ever changing requirements, FG applies Inversion of Control pattern to cope with changed requirements without any product change.
Configuration Driven	Formsgroup is completely configuration driven, thus avoiding the risks in free form code.
Flexible Extension Points	There are multiple extension points exposed in FG layer that facilitates easy customisation.
Easy Maintainability	Formsgroup helps in component maintainability by controlling free form codes.
Tooling	This tool is provided to compile the FG specification. This highly increases productivity of developers.
Multi platform support	Formsgroup specifications can be run in different platforms like J2EE, Portal, .NET and so on.

Separation of Concern

Listed below are the points of concern for the Usecase Developer's and the Framework.

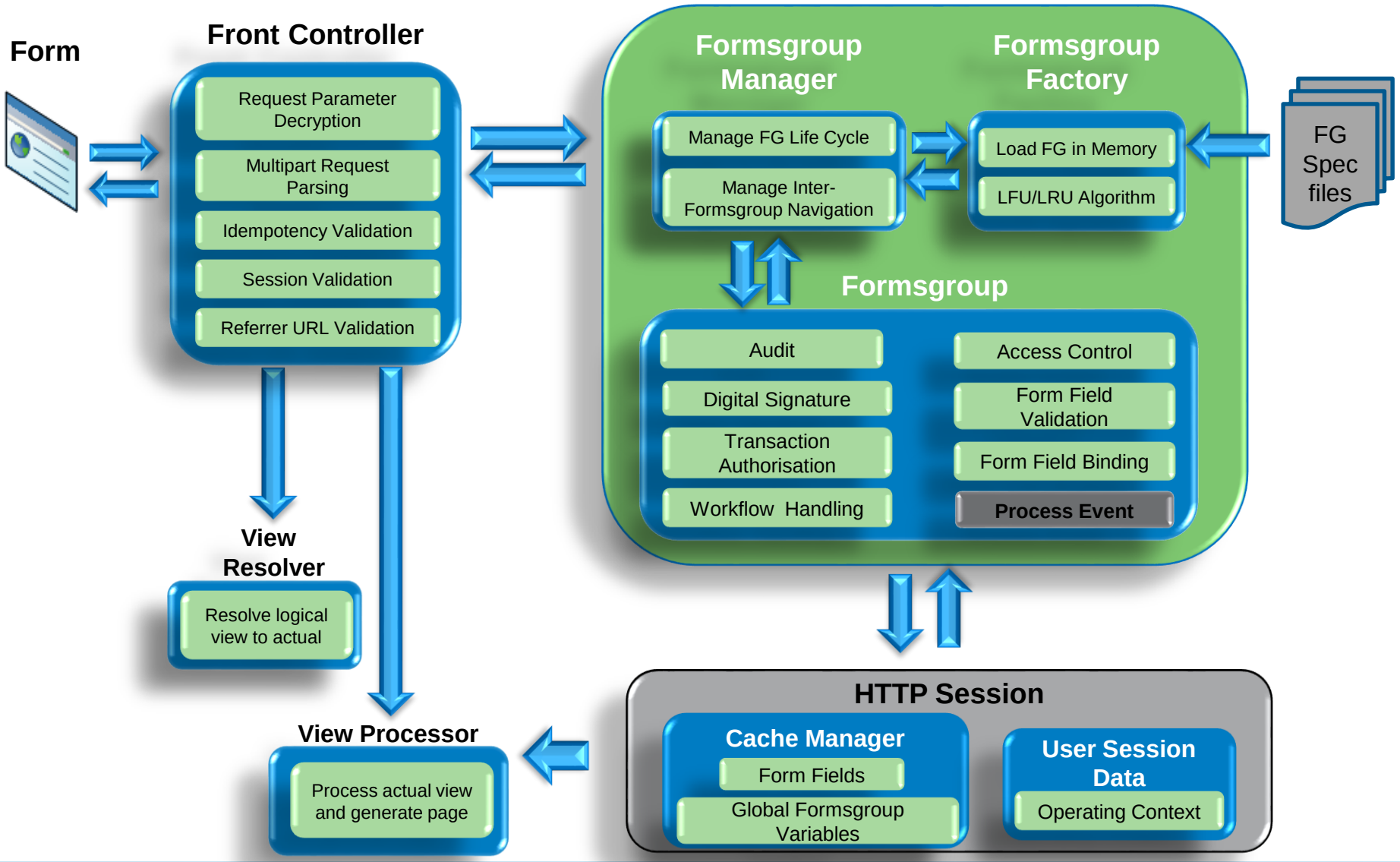
Usecase Developer's Concerns

- Configure
- Event Processing Steps
- Configure Service Orchestration
- Configure Data Mapping
- Configure UI Controls

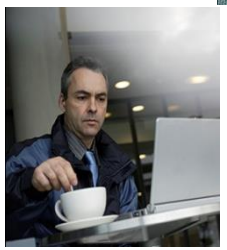
Framework's Concerns

- Flow Management
- Form Management
- Form field validation and binding
- Access Control
- eBanking Workflow
- Transaction Password Validation
- Digital Signature Validation
- Audit
- UI Controls Processing
- Multi Branding Support

Formsgroup: Architecture Overview

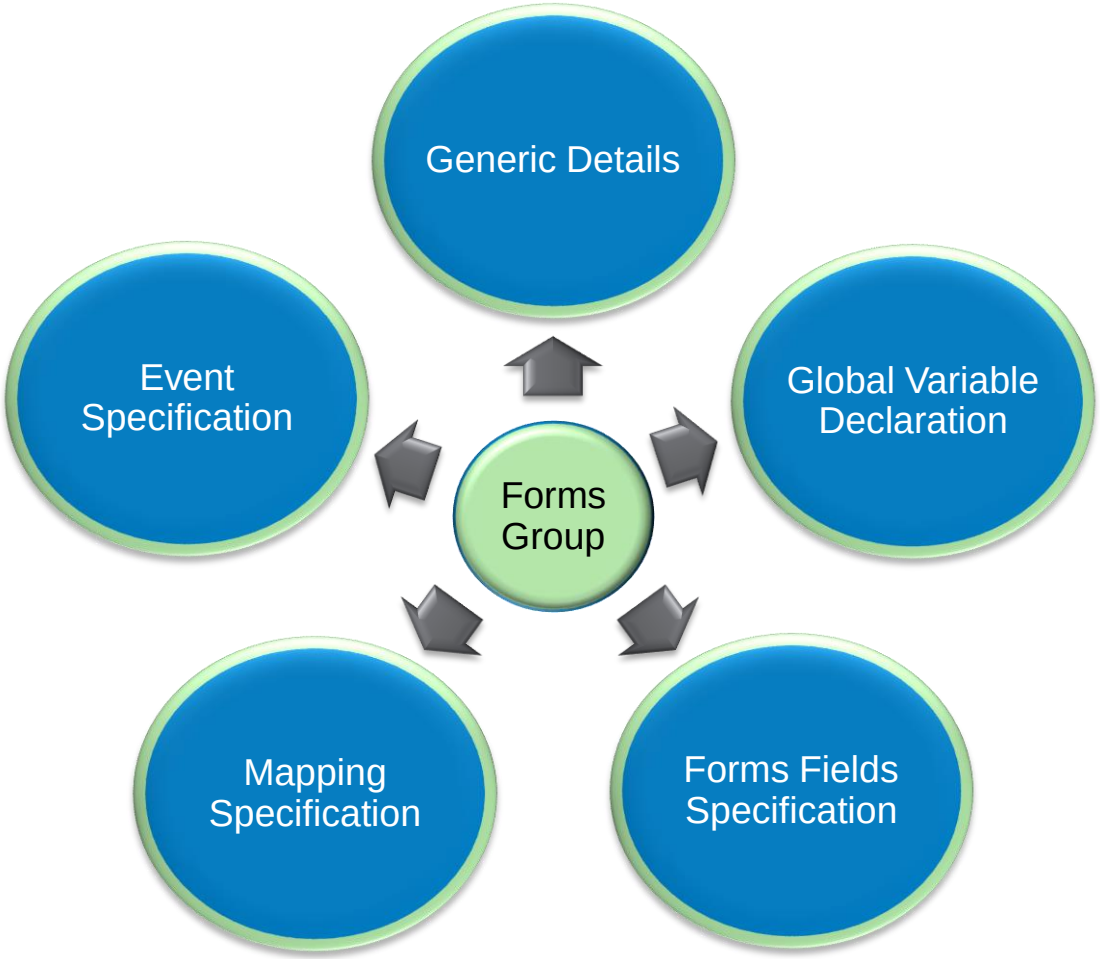


Formsgroup: Definition and Structure



FormsGroup: Structure

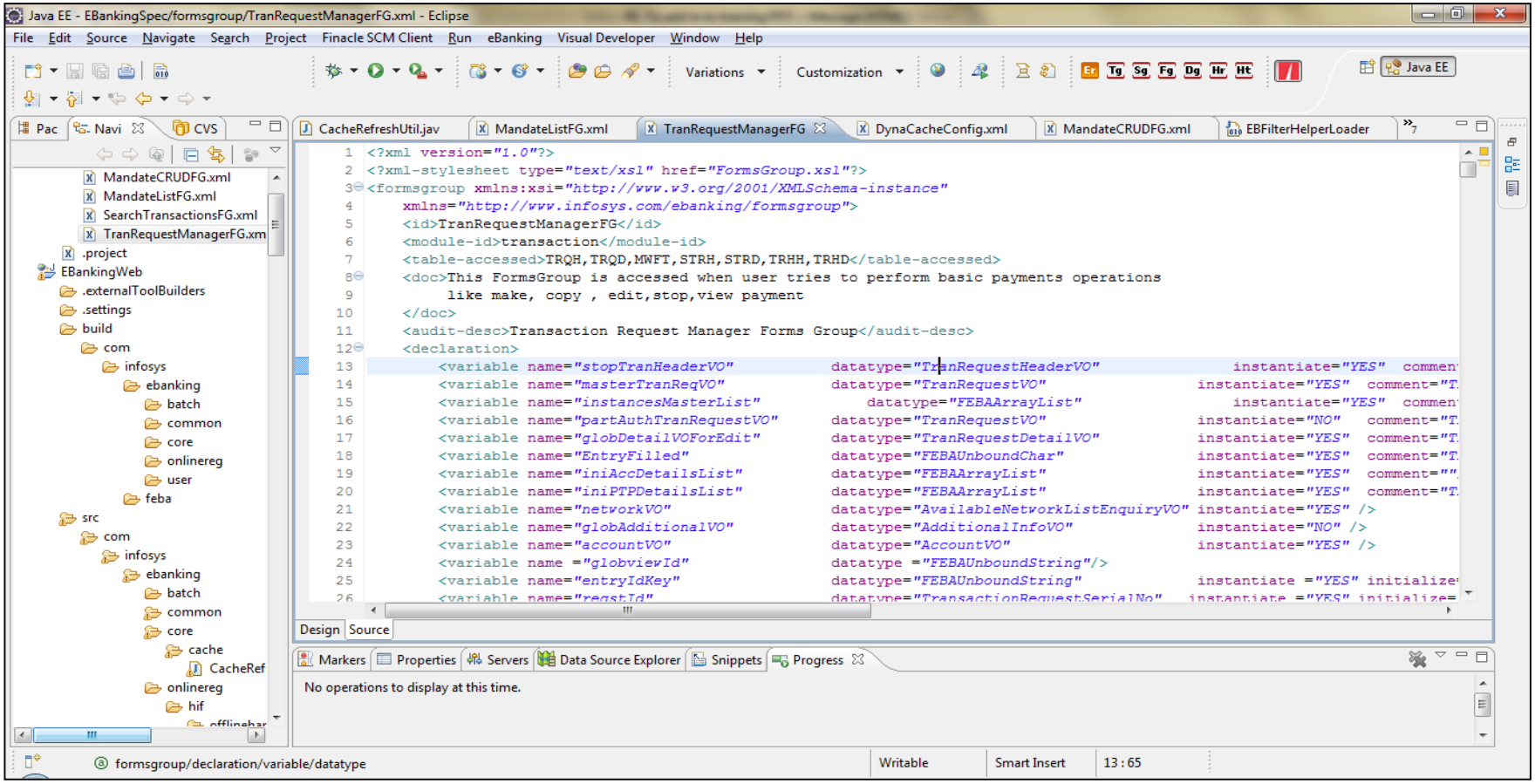
FormsGroup can be structured into the given five components:



Each of these components will be further discussed in detail in the following slides.

Generic Details Section

This section provides the overview about the FG, that is tables access, modules referred and so on. It acts as documentation for the FG.



Global Variable Declaration Section

The Global Variable Declaration Section is not mandatory. It could be included in case of a requirement. It is given within the <variable/> tag.

Sample:

```
<declaration>
<variable name="enquiryVO"  datatype="StateCodeCacheVO“ instantiate="YES" />
<variable name="selectdCntry“  datatype="CountryCode"        instantiate="YES" />
<variable name="stateSelected” datatype="StateCode“          instantiate="YES" />
</declaration>
```

Global Variable Declaration Section (Contd.)

The Global Variable Declaration tag has the following attributes:

- **Name:** Mandatory Attribute
- **Instantiate:** Mandatory Attribute
- **Datatype:** Non-Mandatory Attribute
- **Initialise:** Non-Mandatory Attribute
- **Comment:** Non-Mandatory Attribute

FormsField Specification Section

This is used to specify a list of all the form fields which constitute the FormsGroup.

- These fields are automatically refreshed by the FormsGroup framework from the UI.
- Form fields can be of two types:
  - Basic
  - Array

Form fields has following attributes:

Type

It is an optional attribute. If not specified, the tool will default it as a “basic” data type.

Name

Field is used to identify the form field and it should be unique in the FormsGroup.

<doc>

</doc> tag is used for documentation purpose. It is a mandatory field.

Datatype

Specifies the type of the form field.

FormsField Specification Section (Contd.)

Sample:

```
<form-data>
        <form-field type=”basic”>
            <doc>
        This is the field name with which user selected index
        will be associated
            </doc>
            <name>SELECTED_INDEX</name>
            <datatype>FEBAUnboundInt</datatype>
        </form-field>
        <form-field type="array">
            <doc>
            This is the field name against which country will be
            associated.
            </doc>
            <name>CNTRY_ARRAY</name>
            <datatype>CountryCode</datatype>
        </form-field>
    </form-data>
```

Mapping Specification Section

The Mapping section is used to perform data mapping between various data structures.

- There are two types of mapping: VO Mapping and List Mapping.

VO Mapping

Example:

```
<mapping  type="VO"  id="HeaderMapping" datatype="TranRequestVO">
<doc>This is used in Manual Release required</doc>
<field-mappings>
<field-mapping form-field="REQUEST_ID"     vo-field="requestHeader.requestId"
    comment="" />
<field-mapping form-field="HLD_FLG"   vo-field="requestHeader.HoldFlg"
    comment="" />
</field-mappings>
```

Mapping Specification Section (Contd.)

List Mapping

Example:

```
<mapping type="LIST"
id="StoppedInstancesMapping" datatype="StoppedInstancesEnquiryVO"
recordType="TranRequestInstancesVO">
<doc></doc>
<field-mappings>
<field-mapping form-field="INSTANCE_DATE_ARRAY[i]"    vo-field="resultList[i].instanceDate"
comment="" />
<field-mapping form-field="INSTANCE_STATUS_ARRAY[i]“    vo-
field="resultList[i].statusER.cmCode" comment="" />
</field-mappings>
</mapping>
```

Event Specification Section

Event specification is the area where event handling logic is written. Every use case specific event needs to be handled at this section.

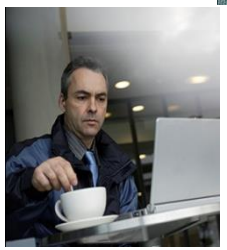
- All FormsGroup need to have at least one event called “LOAD”. This is the first event that will be called when the FormsGroup is loaded.
- All the events will be coded within <events> </events> tag.

Example:

```
<events>
<event name="LOAD">
                                        <doc>This event calls the Retrieve_List event</doc>
                                        <event-handling-block>
                                                <step>FGI::Assign(selectdCntry,CNTRY);</step>
                                                <step>FGI::Assign(stateSelected,STATE_CODE);</step>
                                                <step>FGI::Clear(STATE_CODE);</step>
                                                <step>FGI::CallEvent("RETRIEVE_LIST");</step>
                                        </event-handling-block>
                        </event>

</events>
```

FG Control Instructions



FG Control Instructions

- **FGI** is the acronym for **FormsGroup Instruction**. It is used as part of event specification in FormsGroup.
- Each FGI is processed and interpreted during code generation and corresponding codes are generated.
- The tool does not accept any other FGI other than the supported ones. If used as part of FG spec, an exception will be thrown.

FG Control Instructions (Contd.)

FGLs allow language constructs such as conditional processing or looping processing. The following examples illustrate the usage of FGLs.

Supported FGLs

- FGI:IF
- FGI:ELSE
- FGI:ELSEIF
- FGI:ENDIF
- FGI:ENDCOND inserted by the compiler
- FGI:WHILE
- FGI:ENDWHILE
- FGI:ENDEVENT

Supported Conditional Operators

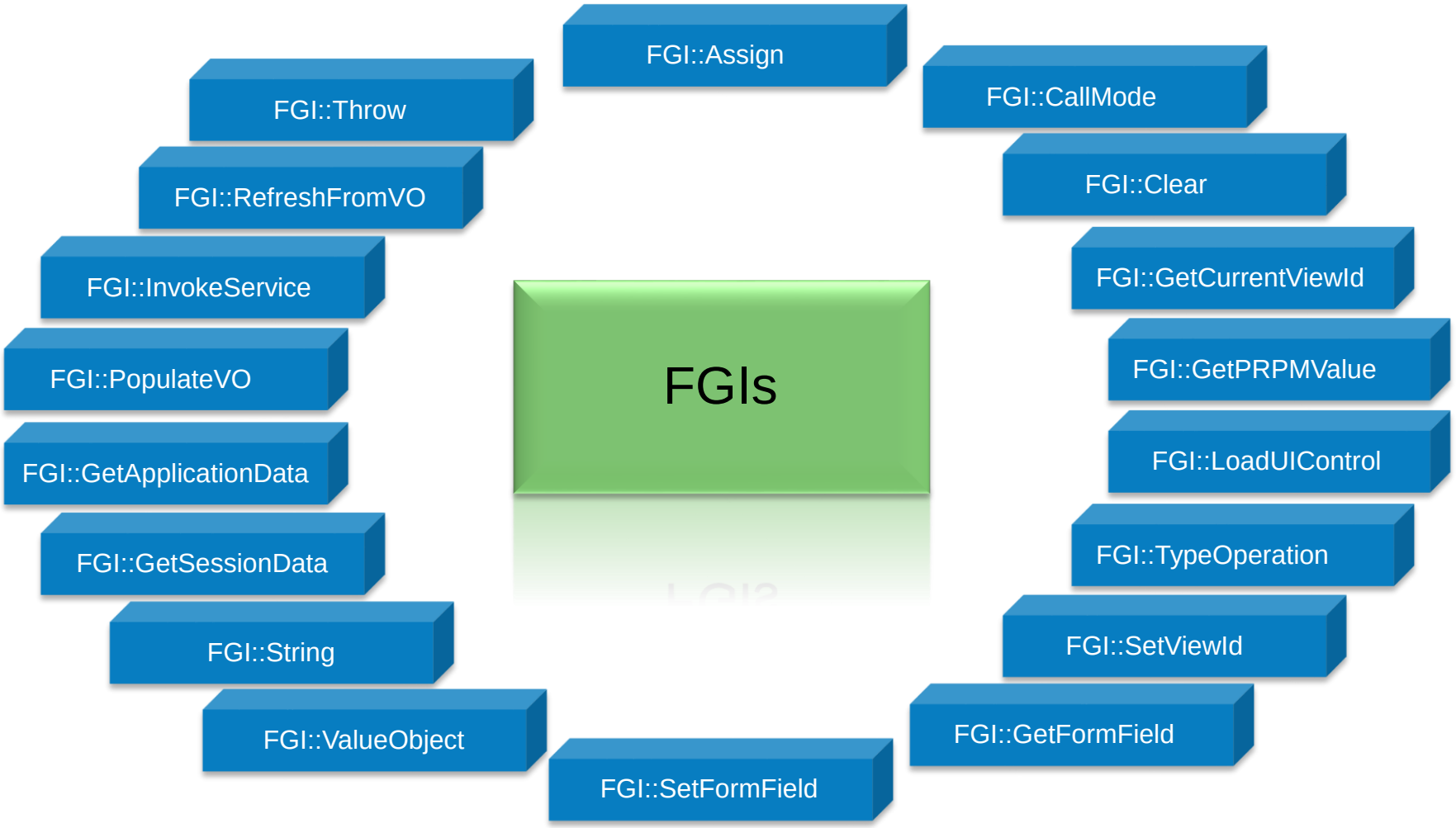
- ==
- !=
- >
- >=
- <
- <=

Supported Logical Operators

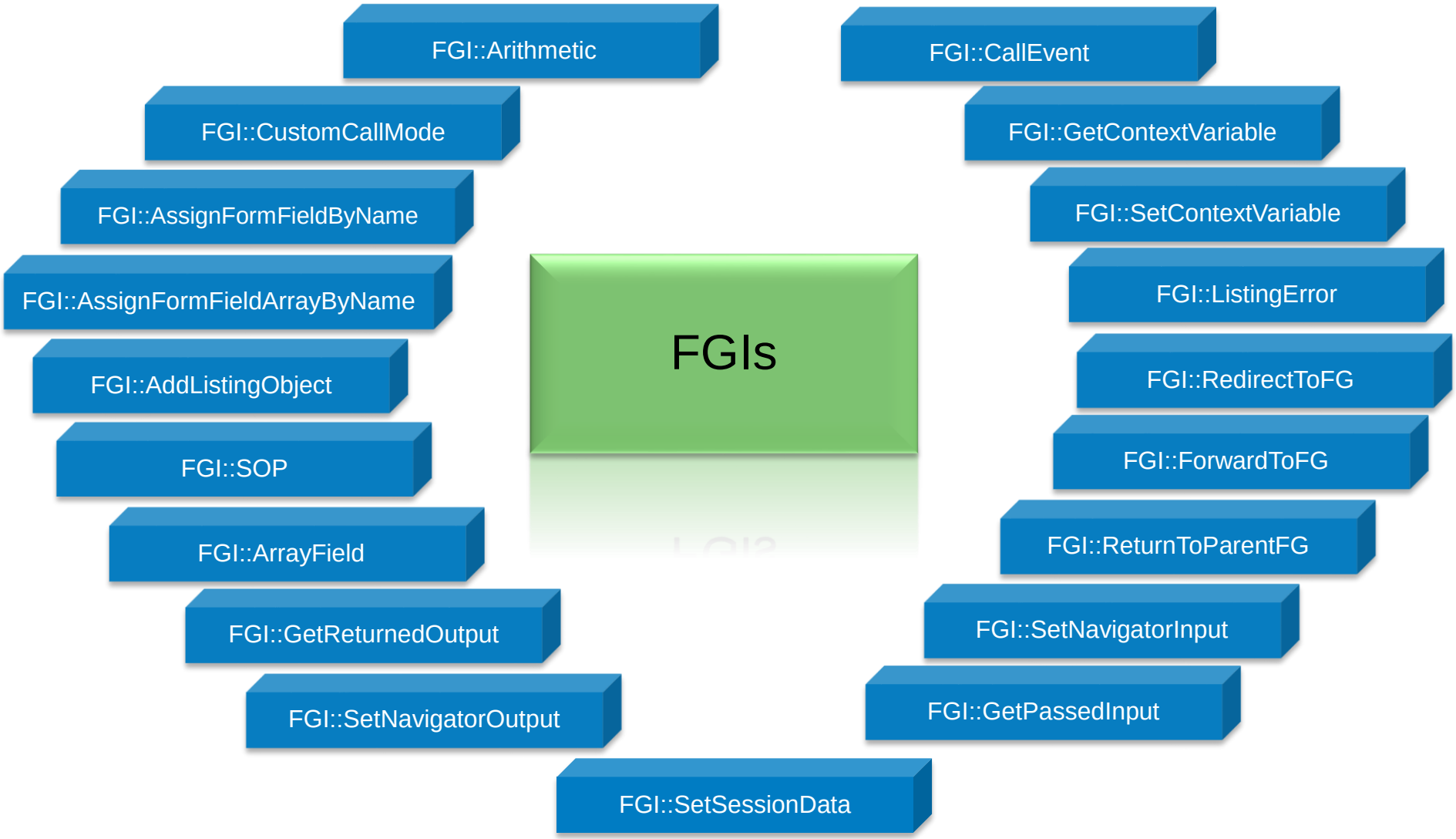
- OR
- AND

FGIs Supported

Following are the FGIs supported by the FormsGroup Generator tool:



FGIs Supported (Contd.)



FGI::Arithmetic

FGI::Arithmetic

- This instruction is used for doing simple arithmetic operations. It supports addition, multiplication and subtraction of 2 operands.

- **FGI::Arithmetic(**operation, total, operand1, operand2**)**

Example:

- *<step>FGI::Arithmetic("ADD",total, I, J);</step>*
- *<step>FGI::Arithmetic("SUB",total, I, J);</step>*

FGI::CallEvent

FGI::CallEvent

- This instruction is used for calling an event from another event within the same FormsGroup.

FGI::CallEvent (eventName**)**

- ***Example:***
- *<step>FGI::CallEvent("LOAD_LIST");</step>*

FGI::GetContextVariable

FGI::GetContextVariable

- This is the FGI to get a context variable from the context object given the Name. It accepts 2 parameters, which are mandatory.
- Parameter 1 - FEBA Type object to which the context variable value needs to be set.
  Parameter 2 = Name of the Context variable. It is case sensitive. If no values are present, it will try to set NULL.

FGI::GetContextVariable (variable,"contextVariable")

Example:

- *<step>FGI::GetContextVariable(USER_TYPE,"USER_TYPE");</step>*

FGI::SetContextVariable

FGI::SetContextVariable

- This is the FGI to set the Context variable against a given NAME. It accepts 2 parameters, which are mandatory.
- Parameter 1 - FEBA Type object to which the context variable value needs to be set.
  Parameter 2 = Name of the Session variable. It is case sensitive. If no values are present, it will try to set NULL.

FGI::SetContextVariable(localVariable,"sessionVariable"**)**

- ***Example:***
- *<step>FGI::SetContextVariable(BANK_ID,"BANK_ID");</step>*

FGI:: ForwardToFG

FGI:: ForwardToFG

- This is the FGI to forward to another FG. There are requirements where one FG needs to invoke another FG. Such requirement can be fulfilled by this FGI. The inputs to next FG can be given using the FGI:SetNavigatorInput.
- FGI::ForwardToFG accepts 3 parameters. They are all mandatory.
- Name of the FG it is calling.
- Call Mode. If no call mode is being used in child FG, you can pass a default value.
- Call Back Event - This is the event to which the control will return to after the control is passed back from the Child or called FG.

FGI::ForwardToFG(nameOfTheFG which needs to be called,callMode,returnEvent**)**

- ***Example:***
- *<step>FGI::ForwardToFormsGroup("CorporateListFormsGroup",1,*
- *"RETURN_FROM_CORPORATE_LOOK_UP");</step>*

FGI::ReturnToParentFG

FGI::ReturnToParentFG

- This is the FGI to return to the calling FG. It does not accept any parameter. However if you want to pass back some values to the calling FG, it can be done using the SetNavigatorOutput.

FGI::ReturnToParentFG()

- ***Example:***
- *<step> FGI::ReturnToParentFG();</step>*

FGI::SetNavigatorInput

FGI::SetNavigatorInput

- This instruction is used for passing input from calling FG or parent FG to child FG. It can pass Form Field values, local variable or global variable as inputs to the child FG.
- It takes in 2 parameters.
- First parameter is the indicator for the type of input. It can be “FIELD” for Form Field or “LVAR” for local variable or “GVAR” for global variable.
- Second parameter is the reference to the variable or value being passed.

FGI::SetNavigatorInput (name using which the field will be accessed, actual value to be passed)

- ***Example:***
- *FGI::SetNavigatorInput("workflowVO",workflowVO);*

FGI::GetPassedInput

FGI::GetPassedInput

- This instruction is used for getting passed input from calling FG or parent FG to child FG. It can pass Form Field values, local variable or global variable as inputs to the child FG.
- It takes in 2 parameters.
- First parameter is the name of the input variable which has been passed from the calling FormsGroup.

FGI::GetPassedInput(variable to get the value, field from where the value needs to be accessed**)**

- ***Example:*** *FGI::GetPassedInput(corpId,"WorkflowFG.CORP_ID");*

FGI::SetNavigatorOutput

FGI::SetNavigatorOutput

- This instruction is used for passing output from child FG to calling FG or parent FG. It can pass Form Field values, local variable or global variable as inputs to the child FG.
- It takes in 2 parameters.
- First parameter is the indicator for the type of input. It can be “FIELD” for Form Field or “LVAR” for local variable or “GVAR” for global variable.
- Second parameter is the reference to the variable or value being passed.

FGI::SetNavigatorOutput(name using which the field will be accessed, actualvalue to be passed**)**

- ***Example:***<step>FGI::SetNavigatorOutput("corporateID",CORP_ID);</step>

FGI::GetReturnedOutput

FGI::GetReturnedOutput

- This instruction is used for getting the output being passed back from the child FG. It will query against the name associated with the value.
- It takes two parameters:
  - Variable in which the value will be accessed.
  - Variable which contains the output value.

FGI::GetReturnedOutput(variable to get the value, variable in which the value is set)

- ***Example:***<step>FGI::GetReturnedOutput(selectedCorpId ,"selectedCorpId");</step>

FGI:: ArrayField

- **FGI:: ArrayField**
- This FGI is used to perform operations on arraylist in the UI Listing Control. Operations such as (GET_LENGTH, GET_ELEMENT_AT, ADD_ELEMENT) can be done. These are explained below:
- Operation : Description
- GET_LENGTH: Gets the Length of the Array
- GET_ELEMENT_AT: Get the value of 'nth' Element of the array
- ADD_ELEMENT: Adds the value of an element in the array
- ***Example:***
  - ***GET_LENGTH***
- *FGI::ArrayField("GET_LENGTH", arraySize, SELECTED_INDEX_ARRAY");*
  - *-it gets the Length of the Array "SELECTED_INDEX_ARRAY" to a integer variable "arraySize"*
  - ***GET_ELEMENT_AT*** *FGI::ArrayField("GET_ELEMENT_AT",field,SELECTED_INDEX_ARRAY,elementVariable);*
- *- it gets the element from the index "elementVariable" from SELECTED_INDEX_ARRAY and sets in "field"*
  - ***ADD_ELEMENT***
- *FGI::ArrayField("ADD_ELEMENT",SELECTED_INDEX_ARRAY,elementValue);*

FGI::SOP

FGI::SOP

- This is used for debugging purpose. It accepts a string variable.

Example:

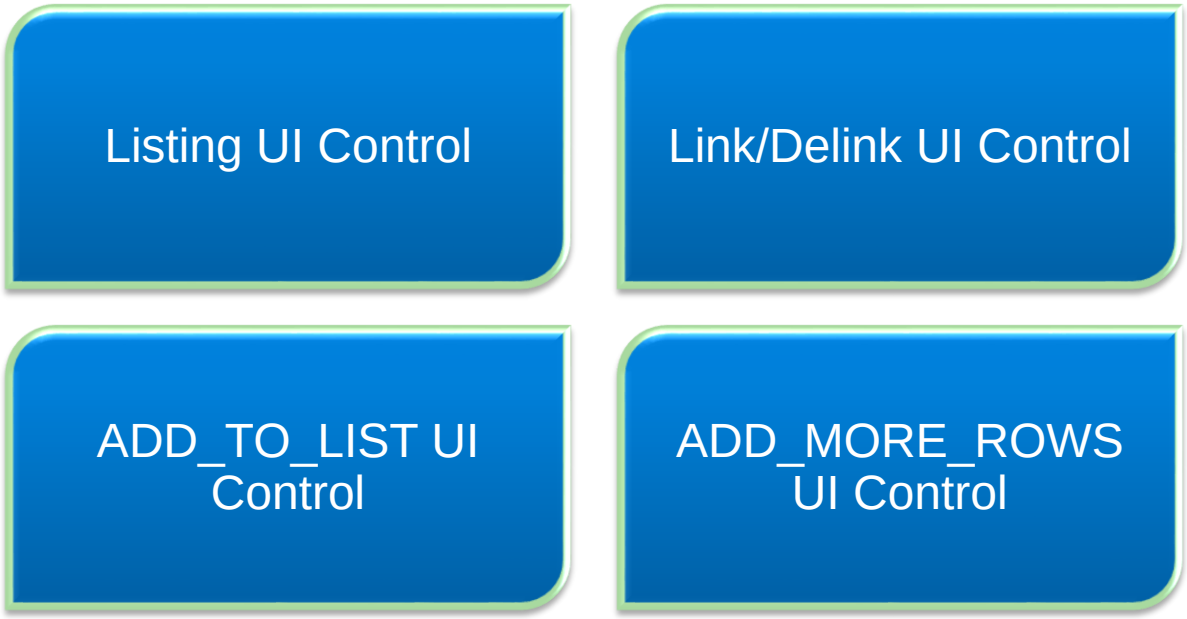
- *<step>FGI::SOP("Retval : " , RetVal);</step>*

FG UI Controls

FG UI Controls

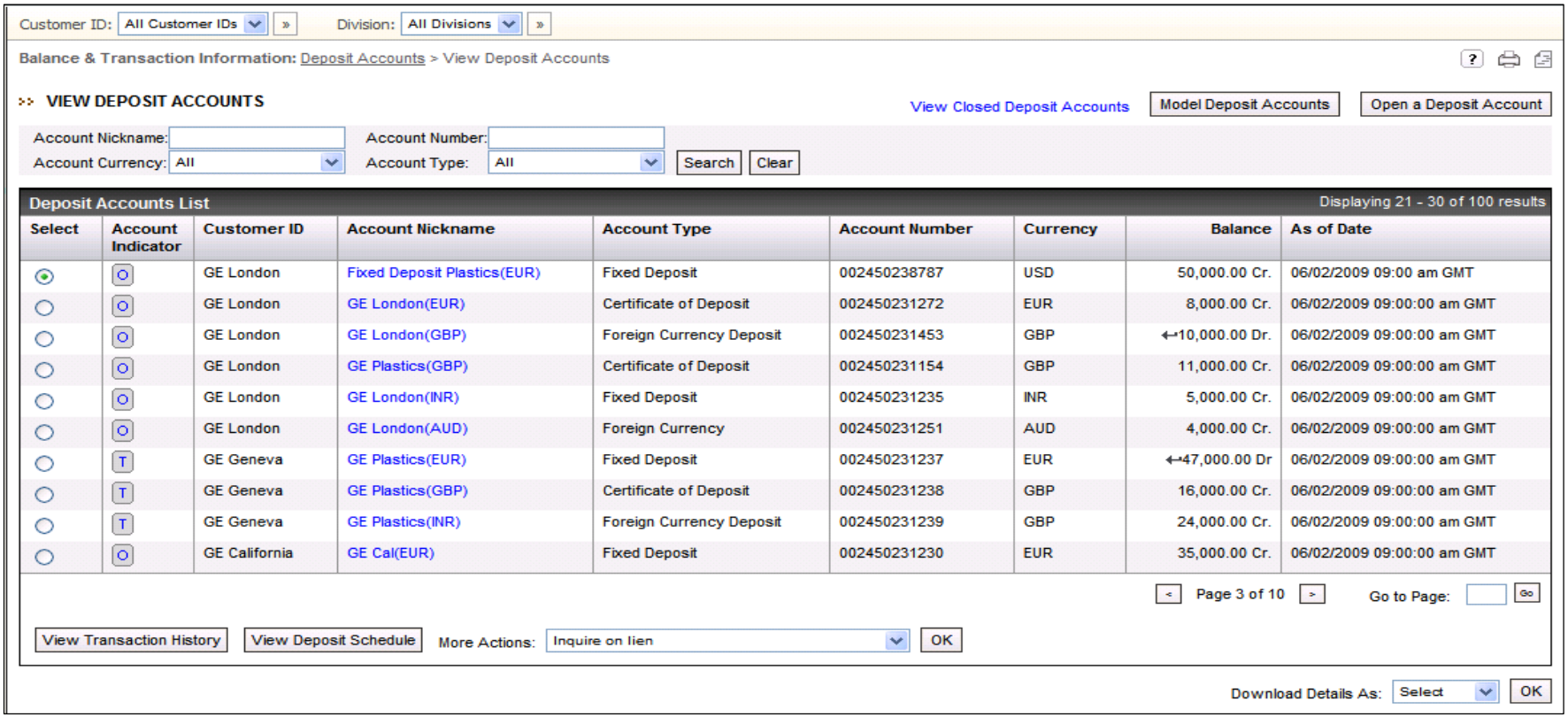
While analysing the screens across Finacle eBanking, it is observed that data representation has specific patterns such as list display and link delink screen.

Currently FG supports the following UI controls:



Listing UI Control Screenshot

Listing UI control encapsulates the records of a table such as arraylist which, enables the display of data in the UI control in a tabular format. It also enables us to toggle between pages with the help of pagination.



Listing UI Control

- **For Parent Listing:**
- <control id="MailListing" type="Listing">

<doc>Parent listing to be used by all listing</doc>

<properties> <property name="inq-obj-name" value="enquiryVO" />

<property name="crit-map-name" value="mailCriteriaMapping" />

<property name="list-map-name" value="mailListMapping" />

<property name="error-map-name" value="errorListMapping" />

<property name="service-name" value="MailListService" />

<property name="service-method" value="fetch" /> </properties> </control>

- **Sample Child Listing:**
- <control id="SearchListing" type="Listing" extends="MailListing">

<doc>Listing to be used by search events</doc>

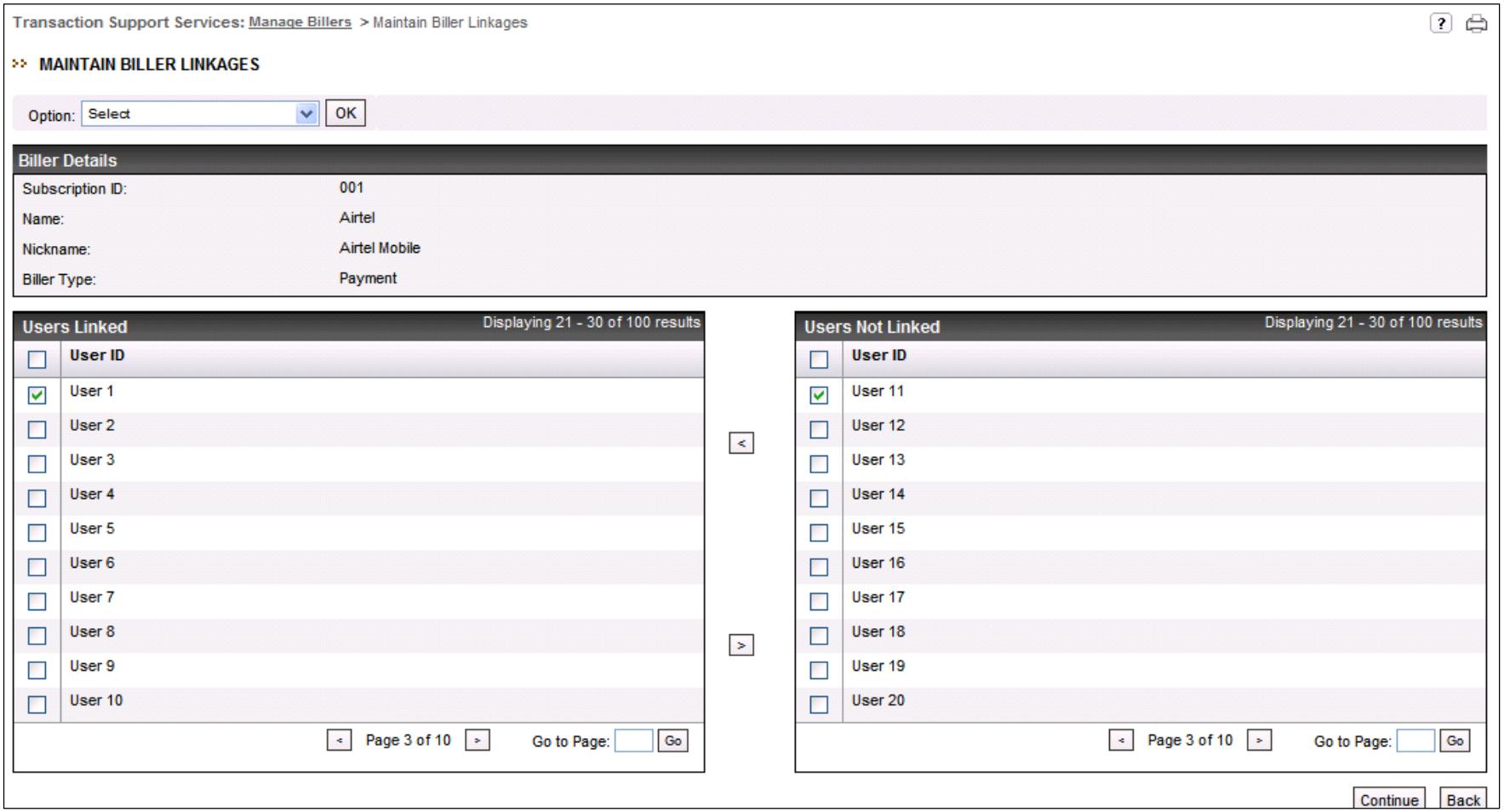
<properties> <property name="list-ui" value="MailViewSearchList" />

<property name="crit-map-name" value="searchMailCriteriaMapping" />

</properties> </control>

Link/Delink UI Control Screenshot

The Link/Delink UI control encapsulates similar events like linking some records from left table to right table or vice versa.



Link/Delink UI Control

- After linking and delinking, once the page is submitted, it's possible to get linked or delinked records(i.e. the delta part)from the UI control memory to send it to service.
- **FGI::UIControl** This FGI helps invoke common UI-based operations such as linking and delinking. These operations are listed in table below:
- **Operations allowed:**
- GET_LEFT_TO_RIGHT_LINKAGES It fetches the list of elements moved from left to right
- GET_RIGHT_TO_LEFT_LINKAGES It fetches the list of elements moved from right to left
- **Example:**
- GET_LEFT_TO_RIGHT_LINKAGES - FGI::UIControl ("LinkDelink","GET_LEFT_TO_RIGHT_LINKAGES",linkedElementList,"MailLinkDelink"); - it creates a list of linkedElementList using UIControl type and UIControl id
- GET_RIGHT_TO_LEFT_LINKAGES -FGI::UIControl ("LinkDelink","GET_RIGHT_TO_LEFT_LINKAGES",deLinkedElementList,"MailLinkDelink"); - it creates a list of delinkedElementList using UIControl type and UIControl id

Link/Delink UI Control

- **Points to note:**

GET_LEFT_TO_RIGHT_LINKAGES-

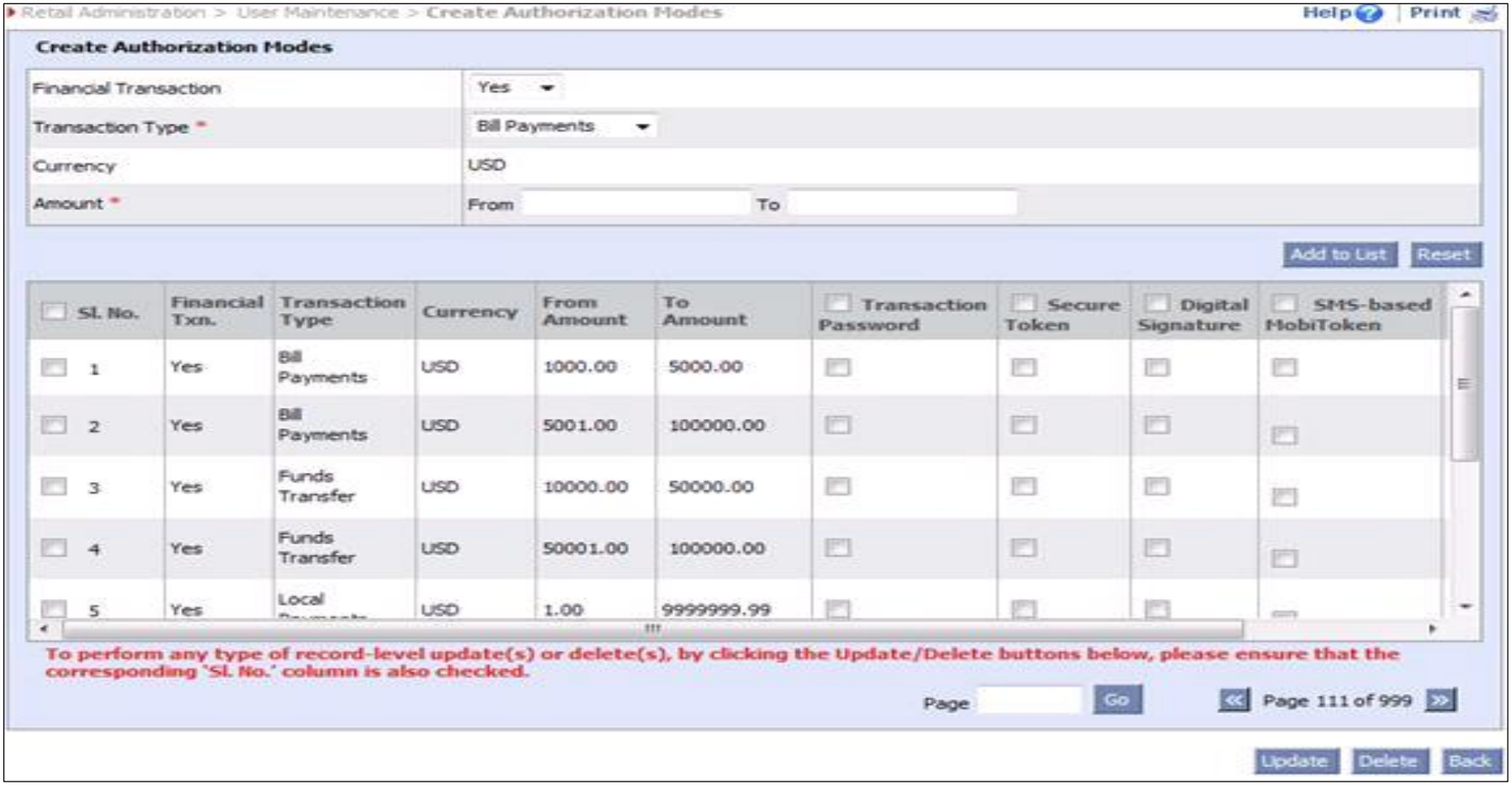
- First parameter must be LinkDelink
- Last parameter should be a UIControl id
- Second parameter should be a operation name
- Output parameter should be list

GET_RIGHT_TO_LEFT_LINKAGES –

- First parameter must be LinkDelink
- Last parameter should be a UIControl id
- Second parameter should be a operation name
- Output parameter should be list

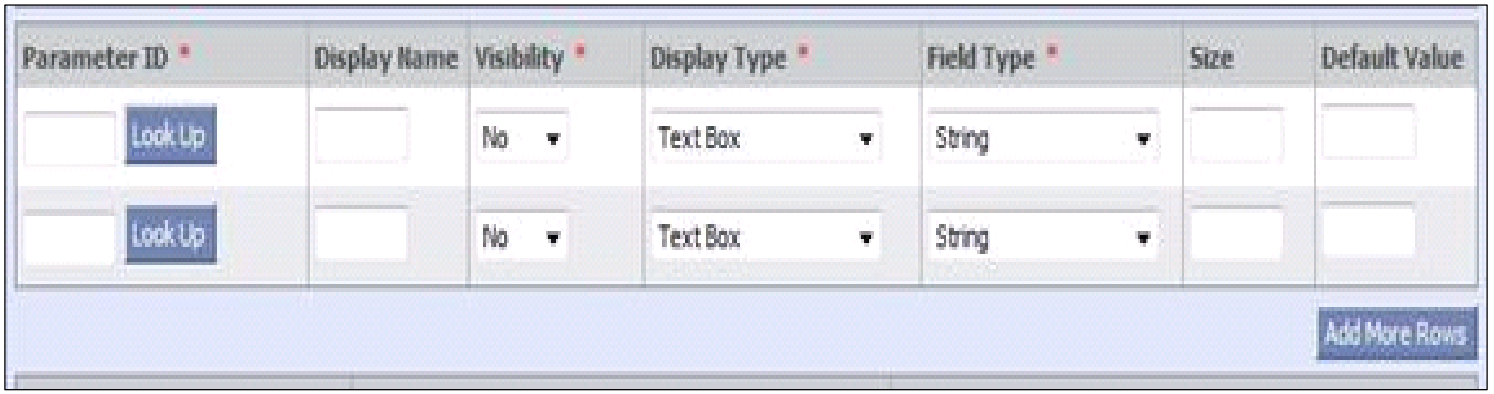
ADD_TO_LIST UI Control Screenshot

This concept is used to add the row of records to the table, which is added/updated to the existing arraylist.



ADD_MORE_ROWS UI Control Screenshot

Add_more_rows adds an entry into arraylist, which represents a row in a table.



Customisation in Formsgroup

Extension in FormsGroup

A product FG can:

- Extend Form fields and Global variables
- Overwrite an existing event (LOAD event can not be overwritten)
- Write a Pre-event and Post-event of an event of the product FG
- Extend an existing mapping

Note: **devrun_formsgroupgen.bat** is the tool used to generate the FG.

Extension: Example 1

```
<extern>
                    <reference name="TXN_USER_CATEGORY" type="Field" />
                    <reference name="TXN_CATEGORY_IMAGE" type="Field" />
                    < reference name="enquiryVO" type="Global" />
</extern>

 <mapping type="VO" id="StateCodeCriteriaMapping_CUSTOM"
   datatype="StateCodeCacheVO">

<doc>Mapping for fields entered in the criteria screen</doc>

<field-mappings>

          <field-mapping form-field="STATE_DESC" vo-
            field="criteria.extensionVO.stateDescriptionCustom" />

          </field-mappings>

</mapping>
```

Extension: Example 2

```
<events>

        <event name="PRE_SEND_COMPOSED">

                <doc>This is implementing pre Send Compose hook.</doc>
                <event-handling-block>
                        <step>FGI::SOP(“In pre event”)</step>
                </event-handling-block>
        </event>

        <event name="POST_SEND_COMPOSED">
                <doc>This is implementing post Send Compose hook.</doc>
                <event-handling-block>
                        <step>FGI::SOP(“In pre event”)</step>
                </event-handling-block>
        </event>

</events>
```

Custom FGI

Following are the steps to write Custom FGI:

1. Create “CustomFGILoader” jave file which extends IFGILoader

```
public class CustomFGILoader implements IFGILoader {
        public void load(Map<String, IFGIStepInterpreter> fgiInterpreterMap) {
        fgiInterpreterMap.put(CustomFGIConstants.SOP , new SOPInterpreter());
        }
```

2. Configure the “FGI_LOADER_CUSTOMIZATION” property in Appconfig.xml

```
<Param name = "FGI_LOADER_CUSTOMIZATION"   value=
"com.futurebank.feba.framework.formsgroup.interpretor.fgi.CustomFGILoader" />
```

3. Create the CustomFGI aJava file which extends AbstractFGIStepInterpreter. Code the logic inside “interpret” method

Custom FGI (Contd.)

4. Make an entry in “CustomFGIConstants.java “ available in path “***com.futurebank.feba.custom.common.formsgroup.fgi***“

```
public class CustomFGIConstants {
    public static final String SOP                                    = "SOP";
    private CustomFGIConstants(){}
}
```

5. Now the FGI is ready to use inside FG post build and compilation

```
<event-handling-block>
<step>FGI::SOP("Hello. U are using Custom FGI.");</step>
</event-handling-block>
```

Module Summary

In this module you learnt to:

- Explain the FormsGroup architecture.
  - Define Formsgroup.
  - Explain the seven Architectural goals
  - Define the separation of concern for the Usecase Developer and Framework.
  - Present an overview of the FG architecture
- Explain the structure of FG.
  - Explain each component of the FG structure, including: Generic Details Section, Even Specification Section, Mapping Specification, Forms Field Specification, Global Variable Declaration.

Module Summary (Contd.)

- Utilise the Control Instructions and the FGIs supported by the FG Generator Tool
- Demonstrate the use of UI controls supported by FG, including:
  - Listing
  - Link/Delink
  - ADD_TO_LIST
  - ADD_MORE_ROWS
- Explain Customisation in FG.
  - Illustrate customisation in FG using suitable examples.
  - Perform the steps to write Custom FGI.

Thank You

Infosys® | Finacle
Building Tomorrow's Enterprise