IRD to Composer Migration

Composer 8.1.2

12/30/2021

Table of Contents

Welcome to the IRD To Composer Migration Guide

This guide contains reference information you will need when migrating IRD strategies into Composer 8.1.2. Also see:

- IRD Functionality Included in Composer.
- Orchestration Extensions for information on actions, objects, functions and events that enable developers to interact with the Genesys platform without having to directly leverage existing SDKs. This allows developers to create feature rich and open applications.
- The *Orchestration Developer's Guide*, Migration from IRD.

Important Note

The information described in this guide is intended to be a migration starter and not a parity+ approach. Composer/ORS handles routing differently than IRD/URS. For this reason, you may need to rethink your migration approach to take advantage of Orchestration Server and Composer's SCXML-based routing strategies. For more information, please see Important Note on Migration.

Strategies That Can Be Imported

You can import IRD routing strategies (8.0 and later) for voice interactions into your Composer environment, where they become routing workflow diagrams. You can only import strategies and/or subroutines that have been previously exported from IRD as XML. After the import process completes, you can edit and validate these new routing workflow diagram files using Composer's diagram editor.

Types of Data That Can Be Imported

In Composer 8.1.2, the following types of data can imported from the IRD 8.x strategy to the Composer workflow diagram:

- Most IRD object blocks that have an equivalent in Composer (based on a mapping between IRD 8.x blocks and Composer blocks, and excluding those blocks that are not being carried forward such as the Custom Server)
- Links between blocks
- Application variables

- Block positioning on the canvas (may be approximated)
- Embedded strategy comments
- Other metadata about the strategy

Placeholder Blocks

After the import process completes, if an IRD object cannot be migrated, the new Composer workflow diagram file contains a "placeholder" block, which is visually distinguishable from other blocks. A placeholder block indicates that further editing of the block properties is required before the new Composer workflow diagram can be completed. Any specific problems in placeholder blocks are highlighted after you validate the new workflow diagram file in Composer.

Manual Tweaking

Most of the migration of IRD routing strategies to Composer workflow diagrams is automated, but a few tasks require manual configuration.

Other Notes on Import Process

- You don't have to import strategies one at a time; you can import strategies and/or subroutines in bulk.
- The new workflow diagram is stored in `Workflows` folder of the Project you select.
- IRD SubRoutines are converted to Composer Sub-Workflow diagrams. New Sub-Workflow diagrams are stored in `Workflows` folder of the Project you select.
- A routing Project created in Composer already contains an Interaction Process Diagram (IPD); therefore, the migration process will not create a new IPD.
- Conversion of IRD Business Processes into Composer Interaction Process Diagrams is currently not supported.

Important Note on Migration

The information in this note applies to Composer 8.1.2.

Composer 8.1 was the first release to introduce IRD to Composer Migration functionality. As you use the migration wizard and migration report, you will find that a good one-to-one mapping exists between IRD objects and Composer blocks for routing voice interactions. A good mapping also exists between frequently used IRD voice routing functions and functions/properties made available within Composer's Expression Builder. However, it is important to understand that a workflow generated in Composer as a result of migration is an Orchestration approximation of the IRD strategy.

In almost all cases, the migration report will list items that you must manually complete before the workflow can be considered fit to run on the Orchestration platform. After migration, those IRD constructs that have an equivalent mapping in Orchestration can be expected to work reasonably well with minimal manual changes. However, certain IRD constructs will require more attention as they may be implemented very differently for the Orchestration platform. Still others, such as script variables, are simply not supported by Orchestration and therefore the migration report will instruct you to find a manual alternative and lists links to helpful topics where available. This wiki also contains recommended alternate approaches for IRD constructs that are not supported in Orchestration.

In general, the migration process is expected to eliminate the majority of the manual work required to redraw an IRD diagram into a Composer workflow. Careful review, editing, and post-migration testing will be key to making the workflow usable.

Due to differences in implementation, Genesys strongly encourages customers to validate the business logic exposed within the migrated items to ensure they are correct prior to deploying them to a production environment.

How to Use the Migration Information

Composer 8.1.2 allows you to migrate routing strategies created with Interaction Routing Designer 8.0+ into Composer Projects as SCXML-based workflow diagrams, which can run on the Orchestration Platform. This guide details the process for doing so.

Important: Composer 8.1.2 migration only supports IRD 8.0+. If the exported IRD XML strategy files were created in versions prior to IRD 8.0, the migrated content may not be supported. In the case where the IRD XML strategy is not version 8.0+, please import the strategy to an Interaction Routing Designer 8.0+ version, make a small change, save, and export as XML. This will insure that the exported IRD XML format is compliant for the IRD to Composer 8.1.2 Migration.

Read a summary of the process below.

Summary of Migration Process

- Your strategies must be Interaction Routing Designer (IRD) 8.0+ strategies before starting this process. For information on migrating IRD strategies, refer to Chapter 1 in the *Universal Routing 8.1 Reference Manual*.
- Start by using IRD to export the strategy as an XML file. Use IRD's **Export to File** option. Set **Files of type** to `(*.xml)` (open). You can also export export multiple strategies at once.
- Continue in Composer by using **File** > **Import** > **Composer** > **Import IRD Strategy** to bring up a wizard. Browse for the XML file(s) to import, select the Composer Project, and click **Finish** to start the migration process.
- View the imported strategy as a workflow diagram in Composer's `*.workflow` diagram editor. The migration process attempts to re-create the IRD strategy so it can run on the Orchestration platform.
- A dialog box appears asking if the user would like to see the migration report. Consult the migration report, which indicates the status of objects migrated and indicates if any manual steps are required. The report also appears in Composer's *Reports* project folder. When viewing the report in the future, right-click the report and select **Open With** > **Web Browser**. Or open the Internal Web Browser view (**Window** > **Show View** > **Other** > **General**).
- After making any necessary changes to the workflow diagram, click the **Validate** button on the main toolbar to check that objects exist in the Configuration Database to which you have connected. In the case of errors, the Problems view becomes visible and error markers are put on the blocks that contain errors. Double clicking on an error in the Problems view will take you to the corresponding blocks that contain the errors. Review each of the errors and do the fixes, then validate again.
- After successful validation, click the **Generate Code** button on the main toolbar to create a properly-formatted SCXML file from the workflow diagram. Static pages (pure SCXML code) are generated in the *src-gen* folder of the Composer Project.
- Use the Log block under Flow Control to record information about an application.

Command Line Migration

A command line interface is also supported for migrating IRD strategies to Composer. Below is a sample command line that can be used to launch Composer and migrate strategies in bulk:

```
Composer.exe -console -consoleLog -debug --launcher.suppressErrors -migrate
-composerProject "JavaComposerProject" -irdXmlFile "D:\\Documents and
Settings\\skathire.AP01\\Desktop\\New Folder\\URS_Multifunc5_drop3.xml" -Xms40m
-Xmx256m -XX:MaxPermSize=256m
```

Command Line Argument	Required	Description
`-migrate`	Yes	IRD Migration flag to indicate that command line migration is involve other command line arguments, li below, are used for information or IRD exported XML file(s) should be for the migration.
`-composerProject`	Yes	An existing Composer project whe migrateddiagramsshould be adde Composer should be set to autom use the workspace where this proj exists.
`-irdXmlFile`	Yes (Mutually exclusive with "-irdXmlFolder".)	IRD XML File Location. Should be specified migrating only a single file.
`-irdXmlFolder`	Yes (Mutually exclusive with "-irdXmlFile".)	IRD XML Folder Location for bulk migration be specified only while migrating multiple present in the specified folder. The migrati proceed to migrate the XML file extensions directly under the specified folder. Please r sure any XML files not relevant to the expo XML are not located directly under this fold

Important: Use a new workspace for migrating IRD strategies using the command line method. If you use an existing workspace, graphical aspects of the diagram may not migrate over correctly, for example, you may see black color filled Entry and / or Exit blocks and the connectors will be of a different style, etc. If you need to use an existing workspace, use a new workspace for migration purposes and then copy over the Composer diagrams to the older workspace.

Note: Dialogs that are shown while migrating using the wizard are not displayed in the IRD command line migration execution. Continue with the About IRD To Composer Migration.

About IRD To Composer Migration

IRD Objects That Can/Cannot Be Migrated

You can migrate routing strategies created with Interaction Routing Designer 8.0+ into Composer Projects as SCXML-based workflow diagrams, which can run on the Orchestration Platform. If your strategies were compiled using an earlier version of IRD, you will need to open them in IRD 8.0, make a change even if it is small change, and save them. Composer 8.1 supports migrating IRD 8.0+ routing strategies only.

Note: IRD 8.x must be used to make a change to the strategy, save the strategy and then export it to XML. If a change is not made in IRD 8.x, the strategy is not saved in the latest format and therefore the exported XML file may result in unpredictable behavior of the migration process.

In the initial 8.1 release of Composer, the following categories of IRD objects can be migrated: Voice Treatments, Data and Services, Segmentation, Routing (inbound voice only), and Miscellaneous (inbound voice only).

Note: The following categories of IRD objects cannot yet be migrated: Multimedia, Outbound, and SMS.

Note: Migration is not available for IRD's Business Rules object. In the 8.1.0 release, Composer adds a Business Rules block, which allows you to interface with the Genesys Business Rules Engine.

Tasks Performed

The IRD to Composer migration:

- Converts IRD strategy objects that have an equivalent block in Composer.
- Creates a placeholder block for those IRD objects that cannot be migrated.
- Provides migration support for converting IRD Subroutines to Composer SubWorkflows.
- Supports migrating IRD strategy variables, comments, links between blocks, and block positioning (approximate).
- Provides migration support for all IRD functions that have an equivalent in Composer.

Note: Conversion of IRD Business Processes into Composer Interaction Process Diagrams is currently not supported.

Manual Updates

The migration process attempts to map IRD elements to their Orchestration equivalent. Given that

IRD/Universal Routing Server strategies and SCXML-based Orchestration workflows are two different implementations, a 1:1 mapping is not always possible. In this case, the migration process flags any manual updates required before the output diagram is able to generate SCXML code that will execute as intended on the Orchestration platform.

Placeholder Blocks

If an IRD object cannot be migrated, the new Composer workflow diagram file contains a "SCXML State" placeholder block, which is visually distinguisable from other blocks and supports arbitrary SCXML content. A placeholder block indicates that further editing of the block properties is required before the new Composer workflow diagram can be completed. Any specific problems in placeholder blocks are highlighted after you validate the new workflow diagram file in Composer.

Migration Report

The migration report documents:

- IRD objects converted to Composer blocks.
- IRD functions converted into the Composer equivalent.
- IRD Items that cannot be migrated or objects that need further user attention. The report suggests a course of action.

Application Variables

IRD to Composer Migration will only support local scoped variables. Migration of IRD application variables of scope local will migrate to Composer's Entry block Variables property.

Important

Starting with IRD 8.0.1, additional variable scopes were introduced that do not exist within the Orchestration platform. Currently, and for the foreseeable future, Orchestration only supports LOCAL scoped variables as these map directly to variables that can be created within SCXML.

Known Issues in Initial 8.1 Release

The initial 8.1 release of IRD to Composer Migration contains the following Known Issues:

- IRD strategy descriptions do not migrate into Composer. IRD Notes not linked to an IRD object are not migrated over. Linked notes objects are migrated into the Composer block notes property.
- Composer block names do not have any correlation with IRD block (object) names since IRD blocks do not have names. Composer blocks are processed in the order in which they were serialized by IRD into the exported .XML file. This could be different from the order in which they appear in the IRD strategy and therefore block names may not appear sequential.

Exporting in Bulk

To Export multiple strategies at once from IRD into XML files:

- In IRD's shortcut bar, select **Export/Import**.
- Click on **Solution Export**.
- In the list of strategies inside the Scripts folder, right-click on any strategy you want to export.
- Select **Add the object to export list** from the context menu. In the table, a checkbox will appear in the cell under the "Add" column. (Note: Subsequently, right-clicking and selecting the same option will remove the object from the export list.) Alternatively, you can just double-click on the "Add" cell itself and it will toggle between being selected (indicated by an "x") and unselected.
- Right-click on the strategy and select **Select format** from the context menu. In the Select format cell of the table, select `open(*.xml)` from the list. Alternatively, in the table row containing the strategy, double-click on the **Select format** cell and select `open(*.xml)` from the list.
- Repeat the above steps for all strategies and subroutines you want to export.
- Right-click on any of the strategies you have included for export, and select **Export** from the context menu.
- Select a folder for export and click **Select**.
- The XML files for all the strategies will be created in the destination you have selected.

Alternatively, see the IRD help. See Exporting and Importing > Exporting and Importing Strategies and Objects.

Multimedia Objects That Cannot Be Migrated

Composer 8.1 migrates Inbound Voice objects and common objects from IRD. It does not support migration for eServices or multimedia blocks. The following IRD 8.0+ Multimedia objects cannot be migrated:

- Route Interaction (Routing toolbar)
- Queue Interaction (Routing toolbar)
- Workbin (Routing toolbar)
- Stop Interaction
- Acknowledgement
- Autoresponse
- Chat Transcript
- Send E-Mail
- Redirect E-Mail
- Forward E-mail
- Reply E-Mail from External Resource
- Screen
- MultiScreen
- Classify
- Attach Categories
- Create Interaction
- CreateEmailOut
- Create Notification
- CreateSMS
- Identify Contact
- Update Contact
- Render Message Content
- Find Interaction
- Update Interaction
- Update UCS Record
- Submit New Interaction
- Distribute Custom Event

Outbound Objects That Cannot Be Migrated

In Composer 8.1, the following IRD 8.0+ Outbound objects cannot be migrated:

- Add Record
- Do Not Call
- Processed
- Update Record
- Reschedule

SMS Objects That Cannot Be Migrated

In Composer 8.1, the following IRD 8.0+ SMS objects cannot be migrated:

- Create SMS Out
- Send SMS Out

IRD and Function Migration

In IRD strategies, functions may be used:

- Within an explicit IRD object such as the Function object or
- As a part of the setting of various IRD object parameters.

IRD Objects Supporting Function Expressions

The table below shows how migration handles IRD Objects that support using Functions in Expressions.

Function Migration Types

IRD Object	Composer Block Used with Valid Expression	Failed/Not Supported (In the ECMAScript block, the failed expression will be commented out.)
If Object	Branching Block	SCXML State Block
Assign Object	Assign Block	ECMAScript Block
MultiAssign	Assign Block	ECMAScript Block
Function Object	ECMAScript Block	ECMAScript Block
MultiFunction Object	ECMAScript Block	ECMAScript Block
Selection Object	Target Block	Target Block

Function Migration Types

Migration attempts to parse function expressions, break them down into individual function calls, and then form an equivalent expression. However, this is a complex operation. Some IRD functions are not supported in Orchestration while other functions are implemented as either Events or Services in Orchestration Server, and their Orchestration equivalent is now a Composer block. Given the different approaches used in Orchestration, migration classifies functions into the following types and handles each type differently.

Function Migration Types

Type	Description	Comment
Direct Mapping	A simple mapping exists. Migration will replace with an equivalent function call.	
Tag Mapping Direct	Function now requires an SCXML tag. For example, GetPriority()	

Type	Description	Comment
	now needs a <queue:query> and then pick up the property from the Done event.	
Tag Mapping Indirect	Function needs to be converted to a property of a block, which was migrated as a counterpart of another block. Effectively, it will merge into another block. For example, Priority() now will become a block property but will not create a new block.	
Composer block	Function requires a new block to be implemented in Composer.	
Composer Implementation	Function for which Composer implements an ECMAScript function.	
Not Supported	Not supported in IRD to Composer migration. You will need to migrate these functions manually. Migration will add a placeholder Generic State block with a SCXML comment inside it. Comment will include IRD block properties. You can add arbitrary SCXML code in this block to achieve the desired functionality.	

Migration Based on IRD Categories

The *Universal Routing 8.1 Reference Manual* organizes functions into categories. The IRD Function object dialog box also uses these categories. The table below summarizes migration handling for each category of IRD functions. For migration detail, click a function category name.

IRD Function Category	General Migration Comment
CallInfo	Most functions map directly to Composer properties, with a few exceptions
Configuration Options	Primarily mapped to Genesys Functional Modules, with a few exceptions
Data Manipulation	Migration through either Database Wizard or External Service block
Date/Time	Primarily replaced with standard ECMAScript Date() manipulation, with a few specialized functions
Force	Implemented as a service
List Manipulation	Replaced with ECMAScript utility script
Miscellaneous	This category is so wide that functions in this category fall within all migration classification

IRD Function Category	General Migration Comment
	types
Reporting	Supported via inbuilt functions
Statistical	Primarily all supported functions, with one exception implemented as a service
String Manipulation	Standard ECMASCript functions
Target Manipulation	A mixture of functions and services is used

CallInfo Function Migration

The table below describes how migration is handled for IRD Functions in the CallInfo category as defined in the *Universal Routing 8.1.x Reference Manual*.

- The `_genesys` Data Subcategory is described in the Orchestration Server Wiki

CallInfo Function	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
ACDQ	_genesys.ixn.interactions[x].voice.acdq	Auto		
ANI	_genesys.ixn.interactions[x].voice.ani	Auto		
Attach	_genesys.ixn.interactions[x].udata	Auto		May use _genesys.ixn.setuData() to attach data directly.
BearerCapability	_genesys.FMname.interactions[x].xdata (BEARER_CAP)	Auto		
BusinessData	_genesys.ixn.interactions[x].udata	Auto		
BusinessDataINT	_genesys.ixn.interactions[x].udata	Auto		
CallID	_genesys.ixn.interactions[x].voice.callid	Auto		
CallType	_genesys.ixn.interactions[x].FMObjectname.type	Auto		
CallUUID	_genesys.ixn.interactions[x].g_uid	Auto		
CED	_genesys.ixn.interactions[x].voice.ced	Auto		
ConnID	_genesys.ixn.interactions[x].voice.connid	Auto		
DeleteAttachedData	Use ECMAScript delete function on _genesys.ixn.interactions[x].udata.xxxx	Auto		May use _genesys.ixn.deleteuData() or the explicit property or Use ECMAScript delete function on _genesys.FMname.interactions[x]. property.
Dest	_data._dest	Auto		Note: Composer 8.1.2 will change this mapping to _genesys.ixn.interactions[InteractionID].pa + '-1'].device In earlier releases, the final expression can be changed to this value manually.
DNIS	_genesys.ixn.interactions[x].voice.dnis property	Auto		

CallInfo Function	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
	or _genesys.ixn.interactions[x].contactedaddr			
ExtensionData	_genesys.ixn.interactions[x].xdata	Semi-Auto		_genesys.FMname.interactions[x]. property can be used to obtain the extension data as well as the datamodel if the datamodel has been mapped correctly with the corresponding id's for the event.
ExtensionAttach	_genesys.ixn.interactions[x].xdata	No		Not Supported. No setXData() function defined. We currently have no means, other than implicitly via the data model, to attach xdata to an interaction and no explicit methods of setting it (i.e., no setXData() function defined).
ExtensionUpdate	_genesys.ixn.interactions[x].xdata	No		Not Supported. No setXData() function defined. We currently have no means, other than implicitly via the data model, to attach xdata to an interaction and no explicit methods of setting it (i.e., no setXData() function defined).
FirstHomeLocation				Not Supported
GetCurrentSwitch	_genesys.ixn.interactions[x].location.media_server	Auto		
GetCurrentTServer	_genesys.ixn.interactions[x].location.control_server	Auto		
CustomerSegment	_genesys.ixn.interactions[x].udata property ("CustomerSegment")	Auto		
GetMediaType	_genesys.ixn.getMediaIntValue (_genesys.ixn.interactions[x].MediaFM.media)	Auto		

CallInfo Function	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
GetRoutingPoint	_genesys.ixn.interactions[InteractionID].parties[InteractionID + '-1'].device	Auto		InteractionID is a system variable in Composer to keep track of the 'current' interaction
GetServiceObjective	_genesys.ixn.interactions[x].udata property ("ServiceObjective")	Auto		
GetServiceType	_genesys.ixn.interactions[x].udata property ("ServiceType")	Auto		
InformationDigits	_genesys.ixn.interactions[x].xdata (INFO_DIGITS) property	Auto		
InteractionData	_genesys.ixn.interactions[x].udata	Auto		
InteractionDataINT	_genesys.ixn.interactions[x].udata	Auto		
LATA	_genesys.ixn.interactions[x].xdata (LATA) property	Auto		
NPA	_genesys.ixn.nPA()	Auto		This function expects the ANI to be provided and so is not directly equivalent to that within IRD. As such, the IRD function would need to be replaced with something like this: _genesys.ixn.nPA(_genesys.ixn.int
NPANXX	_genesys.ixn.nPANXX()			
Orig	_genesys.ixn.interactions[InteractionID].voice.ani	Auto		
OtherTrunk				Not Supported. Not Documented. Used in IRD 8.0.100.12.
PACCode	_genesys.ixn.interactions[x].voice.ced	Auto		
PACType	_genesys.ixn.interactions[x].xdata (PAC_TYPE) property	Auto		
RequestType		Auto		
SetHomeLocation				Not Supported.
StateCode	_genesys.ixn.interactions[x].xdata (STATE_CODE)	Auto		

CallInfo Function	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
ThisTrunk				Not Supported. Not Documented. Used in IRD 8.0.100.12.
UData	_genesys.ixn.interactions[x].udata	Auto		
UDataINT	_genesys.ixn.interactions[x].udata	Auto		
Update	_genesys.ixn.interactions[x].udata	Auto		May use _genesys.ixn.setuData()
UpdateBusinessData				Not Supported. Deprecated. May use _genesys.ixn.setuData()
UpdateInteractionData	_genesys.ixn.interactions[x].udata	Auto		May use _genesys.ixn.setuData()

Configuration Options Function Migration

The table below describes how migration is handled for IRD Functions in the Configuration Options category as defined in the *Universal Routing 8.1.x Reference Manual*.

- The `_genesys` Data Subcategory is described in the Orchestration Server Wiki

Function	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
ExcludeAgents	_genesys.queue.excludeAgents()	Auto		
GetConfigOption	_genesys.session.getConfigOption()	Auto		
GetMediaTypeName	getIxnMediaType()	Auto		
Router	irdRouter()	Auto		
SetCallOption	_genesys.session.setOptions()	Auto		
SetDNIS	_genesys.queue.setDNIS()	Auto		
SetDNISOverride	irdSetDNISOverride()	Auto		
SetTranslationOverride	_genesys.queue.translationOverride()	Auto		
UseActivityType	-	-	-	Not Supported.
UseCustomAgentType	_genesys.queue.useCustomType()	Auto		
UseCustomDNType	_genesys.queue.useCustomType()	Auto		
UseCustomPlaceType	_genesys.queue.useCustomType()	Auto		
UseDNType	_genesys.queue.useDNType()	Auto		
UseMediaType	_genesys.queue.useMediaType()	Auto		

Data Manipulation Function Migration

The table below describes how migration is handled for IRD Functions in the Data Manipulation category as defined in the *Universal Routing 8.1.x Reference Manual*.

- The `_genesys` Data Subcategory is described in the Orchestration Server Wiki

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
FindServiceObjective	_genesys.queue.findServiceObjective()	Semi-Auto		Leverage the the Database Wizard or the External Service block to replace this.
ListGetDataCfg	_genesys.session.getListItemValue()	Semi-Auto		Leverage the the Database Wizard or the External Service block to replace this.
ListLookup				Not Supported. Use the Database Wizard block or the External Service block available within Composer to replace this call. Lists can also be shared via common code that could be included or referenced from the SCXML strategy in the case of static lists.
ListLookupCfg	_genesys.session.listLookupValue()	Semi-Auto		Leverage the Database Wizard or external service to replace this.

Date/Time Function Migration

The table below describes how migration is handled for IRD Functions in the Date/Time category as defined in the *Universal Routing 8.1.x Reference Manual*.

- The `_genesys` Data Subcategory is described in the Orchestration Server Wiki

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
Date	irdDate()	Semi-Auto		This is expected to return a string in the form of MM/ DD/YYYY. There are no direct matches to this defined within for the standard ECMAScript Date() Object. Date formatting functions can be implemented in ECMAScript.
DateInZone	irdDateInZone()	Auto		
Day	irdDay()	Auto		
DayInZone	_genesys.session.dayInZone()	Auto		
GetUTC	irdGetUTC()	Auto		
IsSpecialDay	irdIsSpecialDay	Auto		
IsSpecialDayEx	irdIsSpecialDayEx	Auto		
Time	irdTime()	Semi-Auto		This is expected to return a string in the form of HH:MM where HH is in 24 hour clock format. There are no direct matches to this defined within for the standard ECMAScript Date() Object. A Date formatting utility can be implemented using ECMAScript.
TimeDifference	irdTimeDifference()	Semi-Auto		Usage of this function should be

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
				validated to ensure that the operation is correct.
TimeInZone	_genesys.session.timeInZone()	Auto		
TimeStamp	irdTimeStamp()	Auto		
UTCAdd	irdUTCAdd()	Auto		
UTCFromString	irdUTCFromString()	Auto		
UTCToString	irdUTCToString()	Auto		

Force Function Migration

The table below describes how migration is handled for IRD Functions in the Force category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Composer Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
Force				Not Supported. See other supported Target Manipulations. No alternative defined at present, TRoute is the only other alternative Force function currently supported.
TRoute	Composer Force Route block	Manual		Specify properties of the ForceRoute block.

List Manipulation Function Migration

The table below describes how migration is handled for IRD Functions in the List Manipulation category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Functional Module Mapping	Automatic Update?	Update Required?	Comments/ Manual Steps Required
GetIntegerKey	irdGetIntegerKey()	Auto		
GetMaxSubList	irdGetMaxSubList()	Auto		
GetMinSubList	irdGetMinSubList()	Auto		
GetStringKey	irdGetStringKey()	Auto		
KVListGetKey	-	Manual		ECMAScript can be used to implement this function.
KVListGetListValue	-	Manual		ECMAScript can be used to implement this function.
KVListGetSize	-	Manual		ECMAScript can be used to implement this function.
KVListGetStringValue	-	Manual		ECMAScript can be used to implement this function.
ListGetInteger	irdListGetInteger()	Auto		
ListGetKey	irdListGetKey()	Auto		
ListGetSize	irdListGetSize()	Auto		
ListGetString	irdListGetString()	Auto		
SetIntegerKey	-	Manual		This may be replace with ECMAScript String manipulation on the list.
SetStringKey	-	Manual		This may be replace with ECMAScript String manipulation on the list.

Miscellaneous Function Migration

The table below describes how migration is handled for IRD Functions in the Miscellaneous category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
ActiveServerName	_genesys.session.activeServerName	Auto		
Alarm	Log Block	Auto		label = Alarm Number; expr = Alarm Message; level = 5 for alarm. • For more information, see the Orchestation Server SCXML Technical Reference.
AnswerCall	SubRoutine Block	Auto		The block invokes a bundled subroutine that uses <ixn:accept>
CheckAgentState	_genesys.queue.checkAgentState()	Auto		
ClearTargets	SubRoutine Block	Auto		The block invokes a bundled subroutine that uses <queue:cancel>
ClearUpdateTrigger	Not Supported	-		
CountSkillInGroup	_genesys.queue.countSkillInGroup()	Auto		
CreateSkillGroup	_genesys.queue.createSkillGroup()	Auto		
Delay	Pause Block	Auto		
ExpandGroup	_genesys.queue.expandGroup()	Auto		
ExpandWFActivity	_genesys.queue.expandActivity()	Auto		
ExtrouterError	_genesys.queue.extRouterError()	Auto		
ExtrouterStatus	_genesys.queue.extRouterStatus()	Auto		
GetLastErrorInfo	App_Last_Error_Event system variable in Composer	Auto		`getLastException()` function (Composer 8.1.2) can be used instead. See Composer Help wiki for more

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
				details. In certain cases like `error.queue.submit` event with `error` property "`0013 Remote Error`", the "`description`" property of the event will be populated with URS/IRD compatible error information. Consult Orchestration Server documentation for more details.
GetPriority	SubRoutine Block	Auto		The block invokes a bundled subroutine that uses <queue:query>
GetSkillInGroup	_genesys.queue.getSkillInGroup()	Auto		
JumpToTenant	Not Supported	-		This function cannot be migrated as the Orchestration Platform does not support it.
JumpToStrategy	Not Supported	-		This function cannot be migrated as the Orchestration Platform does not support it.
MultiSkill	-	Manual		Use CreateSkillGroup() with the absence of an agent group, or regular ECMAScript String expressions to build the required string.
OnCallAbandoned	-	Manual		The interaction.deleted event can be used to create logic for processing this condition.
OnRouteError	_genesys.queue.onRouteError()	Auto		
Print	SCXML State block that uses <log>	Semmi-auto		Single parameter migration is supported.

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
				In case of multiple arguments or an expression, the SCXML block is created and the IRD expression is written to it as a comment. A <log> tag should be added to the body of the block. For example, <loglevel="*5*"label="*print*"expr="'*Hello*' + ' ' + '*world*' + '*!*'"/>
Rand	irdRand()	Auto		
ReleaseCall	App_Terminate_Ixn_On_Exit system variable in Entry block	Auto		This variable is set to "1" which cause the Exit block to use <ixn:terminate> to stop the interaction.
ResetBusyTreatments	-	-		Not supported by Orchestration.
SelectTargets	-	Manual		Use ORS function: _genesys.queue.selectTargets()
SelectTargetsByThreshold	-	Manual		Use ORS function: _genesys.queue.selectTargetsByT
SendEvent	Not Supported	-		
SendRequest		Manual		Various actions may be accomplished using the existing <ixn:xx> and <resource:xxx> actions defined within the SCXML Language Spec.
SeverStatus	-	Manual		Use ORS function: _genesys.session.serverStatus()
SetDelayedAttach	-	-		Not Supported
SetInteractionAge	_genesys.queue.setInteractionAge()	Auto		
SetLastError	ECMAScript block	Auto		The block sets the App_Last_Error_Event system variable to the specific error.
SetUpdateTrigger	-	-		Not Suppported.
SuspendForEvent	SubRoutine Block	Auto		The block invokes a bundled SCXML subroutine.

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
TargetComponentSelected	App_Last_Target_Component_Selected system variable	Auto		Composer keeps track of the selected component via the App_Last_Target_Component_Sele variable.
TargetObjectSelected	App_Last_Target_Object_Selected system variable	Auto		Composer keeps track of the selected via the App_Last_Target_Object_Selected variable.
TargetSelected	App_Last_Target_Selected system variable	Auto		Composer keeps track of the selected target via the App_Last_Target_Selected variable.
Timeout	-	Manual		A SCXML State block with a <send> tag can be used.
UpdateScript	-	Manual		Update appropriate values in _genesys.ixn.interactions[x].udata property (key name - MyScript) - The value of this key-value pair will be the configuration layer DB ID of the Script object to be used. This ID will be gotten by Composer and set in the session's logic to set the pair in the interaction.udata property.
UseAgentState	_genesys.queue.useAgentState()	Auto		
UseAgentStatistics	_genesys.queue.useAgentStatistics()	Auto		
VQSelected	App_Last_VirtualQ_Selected system variable	Auto		Composer keeps track of the selected virtual queue via the App_Last_VirtualQ_Selected variable.

Reporting Function Migration

The table below describes how migration is handled for IRD Functions in the Reporting category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Composer Mapping	Automatic Migration?	Update Required	Comments/ Manual Steps Required
Peg	SCXML State Block	Manual		Not supported by Orchestration Server as a function or SCXML tag.
PegValue	_genesys.statistic.sData()	Auto		

Statistical Function Migration

The table below describes how migration is handled for IRD Functions in the Statistical category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
CallsDistributed	Not supported _genesys.stat.sData() with CallsDistributed statistic Not supported			
CallsEntered	Not supported irdCallEntered()			
CallsWaiting	Not supported _genesys.stat.sData() with CallsWaiting statistic			
ClearThresholds	_genesys.queue.clearThresholds()	Auto		
DistributedPercentage	Not supported _genesys.stat.sData() with DistributedPercentage statistic			
DistributedWaitingTime	Not supported _genesys.stat.sData() with DistributedWaitingTime statistic			
GetAvgStatData	_genesys.stat.getAvgData()	Auto		
GetMaxStatData	_genesys.stat.getMaxData()	Auto		
GetMinStatData	_genesys.stat.getMinData()	Auto		
InVQWaitTime	Not supported _genesys.stat.sData() with InVQWaitTime statistic			
NotDistributedPercentage	Not supported _genesys.stat.sData() with NotDistributedPercentage statistic			
NotDistributedWaitingTime	Not supported _genesys.stat.sData() with NotDistributedWaitingTime			

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
	statistic			
PositionInQueue	Not supported _genesys.stat.sData() with PositionInQueue statistic			
ResetStatAdjustment	_genesys.queue.resetAdjustment()	Auto		
SData	_genesys.stat.sData()	Auto		
SetStatAdjustment	_genesys.queue.resetAdjustment()	Auto		
SetStatUpdate	SCXML State Block	Manual		Use <delay send> in the SCXML State Block
SetThresholdEx	Not Supported as a function	Manual		Define a global variable/property which contains the global threshold used for target selection. This can then be used on any Target or Route Interaction block. This is not an exact workaround but is a close approximation and can be ignored in favor of the blocks that expose the threshold capability.
UseCapacity	_genesys.queue.useCapacity()	Auto		

String Manipulation Function Migration

The table below describes how migration is handled for IRD Functions in the String Manipulation category as defined in the *Universal Routing 8.1.x Reference Manual*.

Function Name	Functional Module Mapping	Automatic Migration?	Update Required?	Comments/ Manual Steps Required
Cat	irdCat()	Semi-Auto		Some expressions may require manual migration. Use "+" to concatenate strings.
StrAsciiBreak	irdStrAsciiBreak()	Auto		
StrAsciiTok	irdStrAsciiTok()	Auto		
StrChar	irdStrChar()	Auto		
StrGetChar	irdStrGetChar()	Auto		
StrNextTokInd	-	Manual		Not Supported.
StrLen	irdStrLen()	Auto		
StrReplace	irdStrReplace()	Auto		
StrStr	irdStrStr()	Auto		
StrSub	irdStrSub()	Auto		
StrTargets	-	Manual		Concatenate strings to build targets.
StrToLower	irdStrToLower()	Auto		
StrToUpper	irdStrToUpper()	Auto		

Target Manipulation Function Migration

The table below describes how migration is handled for IRD Functions in the Target Manipulation category as defined in the *Universal Routing 8.1.x Reference Manual*.

- The `_genesys` Data Subcategory is described in the Orchestration Server Wiki.

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
BlockDN	_genesys.queue.reserveTarget()	Auto		
CCTExtractTargets	_genesys.queue.cCTExtractTargets()	Auto		
DeliverCall	-	Manual		Not Supported. This function was initially developed to shuttle interactions between IVR and the original routing point. During Migration, the user should consider using either one of the routing blocks within composer to emulate the required and desired behavior. It may occur that the Force block is the most appropriate. When using this function a new block must be created and linked appropriately by the end user for the desired behavior.
DeliverToIVR	Not supported	-		
GetRemoteAccessCode	Not supported	-		
IncrementPriority	_genesys.queue.incrementPriority()	Auto		
IncrementPriorityEx	Not supported	-		
KeepQueue	Not supported	-		Use the URS Function block in Composer to call the URS HTTP interface methods (for example,

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
				KeepQueue). For more information on parameters and syntax, refer to the Universal Routing 8.1 Reference Manual.
NMTExtractTargets	_genesys.queue.nMTExtractTargets()	Auto		
Priority	Assign to Priority attribute of Target block	Manual		Assign to appropriate Target block
PriorityLimits	-	Manual		Use ORS function: _genesys.queue.priorityLimits()
PriorityTuning	_genesys.queue.priorityTuning()	Auto		
RouteCall	SCXML State Block	Manual		Use a Routing block: Target or Route Interaction block.
Routed	_genesys.queue.routed()	Auto		
SelectDN	SCXML State Block	Manual		Use a Routing block: Target or Route Interaction block.
SetTargetThreshold	Assign Block	Manual		Threshold is now exposed within the Target block and Route Interaction block
SetVQPriority	Assign Block	Manual		The SetVQPriority function is typically used prior to target selection, therefore this can be replaced with a variable to indicate the desired priority and then passed into the target selection as a part of the Target block.
SuspendForDN	Not supported as a function.	Manual		SuspendForDN[timeout] cannot be migrated automatically. A Target block with the correct timeout may be used.

Function Name	Functional Module Mapping	Automatic Migration	Update Required	Comments/ Manual Steps Required
SuspendForTreatmentEnd	Not supported as a function	Manu		This will map onto a transition event for the end of treatment. It is therefore expected that any logic required to wait for treatment end be linked off of such transitions. Such events are for example dialog.stop.done. This may have to be combined with a delay event to simulate timeouts if the desired behavior cannot be correctly modeled with the exposed Composer treatment blocks.
TargetSelectionTuning	_genesys.queue.targetSelectionTuning()	Auto		
Translate	_genesys.queue.translate()	Auto		

Note on the SelectDN Function

The SelectDN URS function's parameters differ between SCXML and IRD usage. From Composer, the SelectDN function can be invoked via either the URS Function block or the SCXML State block. Also, URS handles the request differently when the request is received from an external source, such as from ORS. Due to this difference in behaviour, ORS and Composer strategy responses might differ from that of iRD/URS.

In such cases where the SelectDN function is called from externally, the function must have an implicit integer parameter in its first place, used to refer to a specific entrance call into a queue.

In such cases where the function is used externally, users must specify an integer in the first place. For example,
`[`**`1`**`,"VirtualQueue1","10","RStatAgentsReadyvoice","StatSelectMax","agent3601@.A"]`.

When called externally as shown above, the SelectDN will place the call into a queue but does not return a ready target. It exits immediately with the result, `timeout`. Though the call is placed in a queue, the function does not wait for the result of the target selection process. As the target selection process could be a lengthy operation and might include sub-processes such as, agent reservation, and attaching data, it might not be possible to accommodate all this within the processing time used by the SelectDN function.

Important

ORS does not have a native equivalent to the SelectDN URS function. iRD/URS behaviour is different from that of Composer/ORS behaviour. Strategy design might have to be different to achieve similar outcomes.The Composer strategy can use two Target blocks - the first block to queue to the VQ and request a dummy target, and the second block to request for the actual target.

Composer Blocks Mapped to IRD Objects

The table below maps IRD objects to Composer blocks. For detailed migration information on an IRD object/Composer block, click a row under the **IRD Category** heading.

IRD Category	IRD Object	Composer Block	Automatic?
Voice Treatments	Collect Digits	User Input	Auto
	Play Announcement & Collect Digits	User Input	Auto
	Verify Digits	UserInput	Auto
	Text to Speech & Collect Input	UserInput	Auto
	Play Announcement	PlayMessage	Auto
	Text to Speech	PlayMessage	Auto
	Play Application	Play Application	Auto
	Record User Announcement	Create User Announcement	Auto
	Delete User Announcement	Delete User Announcement	Auto
	Busy	PlaySound	Auto
	Fast Busy	PlaySound	Auto
	Music	PlaySound	Auto
	Ringback	PlaySound	Auto
	Silence	PlaySound	Auto
	RAN	PlaySound	Auto
	IVR	IVR	Auto
	Cancel Call	Cancel Call	Auto
	Set Default Route	Set Default Route	Auto
	Pause	Pause	Auto
Data & Services	External Service	External Service	Semi-Auto
	WebService	WebService	Manual
	Database Wizard	DBData block	Manual
Segmentation	Generic	Branching	Semi-auto
	Date	Branching	Semi-auto
	Time	Branching	Semi-auto
	Day of Week	Branching	Semi-auto
	DNIS	Branching	Semi-auto

IRD Category	IRD Object	Composer Block	Automatic?
	ANI	Branching	Semi-auto
	Business		No - not voice
	Classify		Future
	Screen		Future
Miscellaneous	Entry	Entry	Auto
	Exit	Exit	Auto
	IF	Branching	Semi-auto
	Assign	Assign	Semi-auto
	Function	Assign	Semi-auto
	Macro	ECMAScript	Manual
	Error Segmentation	Exception Ports	Manual
	Call SubRoutine	Sub Routine	Semi-Auto
	Muti-Assign	Assign	Auto
	Multi-Attach	User Data	Semi-Auto
	Multi-Function	Assign	Auto
Routing	Selection	Target	Auto
	Service Level Rule		
	Load Balancing Rule	Routing Rule	Auto
	Percentage Rule	Routing Rule	Auto
	Statistics Rule	Routing Rule	Auto
	Switch to Strategy Rule	Orchestration Server 8.01 does not support switch to strategy routing rules.	
	Default Route	Default Route	Auto
	Force Route Rule	Force Route Block	Manual
Multimedia			Future
Outbound			Future
SMS			Future

Data and Services Object Migration

External Service Object

The External Service Object in IRD is used to exchange data with third-party (non-Genesys) processes/applications that use the Genesys Interaction SDK or any other server/application that complies with the Interaction Server communication protocol.

Composer migrates this object to ***External Service block*** which is very similar to the IRD object and enables calling ESP APIs. It supports all properties exposed by the IRD object except for a behavior difference regarding user data input to the ESP API. The IRD object has a checkbox to disable sending userdata in the ESP call whereas Composer, by default, does not send userdata. Instead, userdata keys to be included in the ESP call need to be specified in the Composer block.

What needs to be done manually?

1. Specify UserData to be passed in the block as the entire userdata will no longer be passed in the ESP request.

IRD Source Property	Composer Block Property	Migration Transformation	Comments
Application type	None	No need for migration	IRD uses it as a UI filter to narrow down the list of applications. Composer displays application in a tree organized by application type.
Application name	Application	Property value is migrated without change	Both properties have the same semantics and intent. They point to an application defined in configuration server
Service	Service Name	Property value is migrated without change	Both properties have the same semantics and intent.
Method	Method Name	Property value is migrated without change	Both properties have the same semantics and intent.
Timeout Property	Service Timeout Property	Property value is migrated without change	Both properties have the same semantics and intent.
Parameters	Method Parameters	Property value is migrated without change	Both properties have the same semantics and intent.
Don't send user data	User Data	Not migrated	To optimize the ESP request, Composer

IRD Source Property	Composer Block Property	Migration Transformation	Comments
			requires relevant user data keys to be specified.
Result Tab	Result Property	Not migrated unless IRD stored output to a variable	Composer uses other blocks to attach data and mapping results to variables.

Web Service Object

In IRD this object is used to invoke SOAP WebServices and get results that are then used in other parts of the strategy.

Composer migrates this object to the ***Web Service block***. This block is very similar to the equivalent IRD object. It uses a WSDL file (specified as part of the project or a URL) to determine details of the SOAP WebService like available services, bindings, end points etc and exposes properties to pass in parameter values and retrieve results back into the application. In addition, this block also offers offline usage where the SOAP call is not made at runtime and instead user provided values are used for output parameters. It also provides access to the Web Services Explorer that can be used to test SOAP WebServices at design time without the need for a test call or interaction.

What needs to be done manually?

IRD does not store a WSDL service URL which is used by Composer to populate all the block properties. Therefore no properties are set automatically. Specify the WSDL URL in the Composer block and select other properties that are populated based on the WSDL URL.

IRD Source Property	Composer Target Property	Migration Transformation	Comments
WSDL Location	Service URL	None. The WSDL URL has to be specified manually in the Composer block. The IRD object does not retain the WSDL URL therefore the original URL will have to be entered again in the Composer block.	Both properties have the same semantics and intent.
Service name	Available Services	None. Both source and target properties are strings.	
Method name	Operations	None. Both source and target properties are strings.	
Method namespace	Target Name Space Uri (Hidden property)	None. Both source and target properties are strings.	In Composer Web Service block, Namespace gets

IRD Source Property	Composer Target Property	Migration Transformation	Comments
			automatically set from the parsed WSDL file.
SOAPaction	Soap Action (Hidden property)	None. Both source and target properties are strings.	In Composer's Web Service block, SOAPAction gets automatically set from the parsed WSDL file.
Request Parameters	Input Parameters	None. Both source and target properties are a list of either Variable or String.	
HTTP Authentication (Anonymous / Basic)	Authentication Type	None. Both source and target properties are strings.	Digest Authentication is not supported in Composer
Name	Login Name	None. Both source and target properties are strings / / Variable names.	Authentication User name
Password	Password	None. Both source and target properties are strings / Variable names.	Authentication password
Assign output values to variables by mapping SOAP response values	Map Output Values to Variables.	String Value "AssignByKey" in IRD will be considered as True (Boolean) in Composer	
Output Values	Output Result	None. Both source and target properties are a list of either Variable or String.	Output Params mapping can be done only when the "AssignByKey" option is chosen on the IRD side. Composer doesn't support "AttachByKey" option.

Note: As IRD doesn't have any option to choose HTTP methods, the Use Protocol property of the Composer Web Service block will always be set to "SOAP".

Database Wizard

In IRD, this object is used to query a database and the queried information can then be attached to the call or assigned to a variable.

Composer migrates this object to an instance of the ***DBData Block***. The DBData block does not utilize DBServer unlike the equivalent object in IRD. Instead, it uses a set of server side pages (Java/JSP or ASP.NET/C#) that execute on the application server as part of the Composer generated application. This block uses a database connection defined in the Composer project that can be configured to use connection pooling transparently. It includes a visual query builder and a stored procedure call helper to visually design a query or to invoke a stored procedure and test it from within the block. If situation where the query is too complex to be created visually or is already available, it supports specifying a file containing a query to be used instead of a query framed using the query builder.

What is created automatically?

These siginificant differences in paradigm mean that the connection information is not migrated over. Instead migrated creates a DBData block, creates a text file containing the query from the IRD object and sets the DBData block to use this file. Stored Procedures calls are not migrated over automatically and should be specified using Composer's Stored Procedure Helper UI.

What needs to be done manually?

1. Check the SQL query written to the .sql file that the DBData block points to.
2. Create a Dababase connection using the Database Connection Manager. Set the DBData block to use this connection.

To see a list of supported databases, please consult online help in Composer.

IRD Source Property	Composer Block Property	Migration Transformation	Comments
SQL	SQL File Property set to a file containing the SQL statement	The SQL will be extracted and written to a .sql file. The DB Data block will point to this .sql file.	
Access Point	Database connection.	Connection information is not migrated.	DBServer is no longer used. See post-migration manual steps.

Miscellaneous Objects Migration

The 8.1 release of Composer supports the IRD Miscellaneous objects as indicated below.

Entry Miscellanous Object

Composer migrates the IRD Entry object to an instance of the Entry block. Local variables defined in the IRD strategy are added to the Entry block.

IRD Source Element	Composer Target Element	Transformation	Comments
Entry Object	Entry Block	Auto	

Exit Miscellaneous Object

Composer migrates the Exit object to an instance of the Exit block.

IRD Source Element	Composer Target Element	Transformation	Comments
Exit Object	Exit Block	Auto	

If Miscellaneous Object

Composer migrates the If object to an Expression.

IRD Source Element	Composer Target Element	Transformation	Comments
If Object	The type of Composer block depends on the condition used in the If block. If the condition or expression can be migrated to Composer, the If object is migrated to a Branching block whose Conditions property will hold the migrated expression. If the condition cannot be migrated to Orchestration, a Generic	Semi-Auto	Migration will parse any functions/ expressions and form an approximate Orchestration equivalent. Some instances will need manual changes.

IRD Source Element	Composer Target Element	Transformation	Comments
	State block is created which will have to be manually filled in.		

Assign Miscellaneous Object

Composer migrates the Assign object to an Assign block.

IRD Source Element	Composer Target Element	Transformation	Comments
Assign Object	Expression. If the expression is valid, the Assign object is migrated to an Assign block else a Generic State block.	Semi-Auto	Migration will parse any functions/ expressions and form an approximate Orchestration equivalent. Some instances will need manual changes. These are typically functions that are now exposed as SCXML tags or tag attributes in Orchestration. Such cases cannot be migrated to ECMAScript functions alone and require a combination of functions and other blocks.

Function Miscellaneous Object

Composer migrates the Function object to ECMAScript.

IRD Source Element	Composer Target Element	Transformation	Comments
Function Object	ECMAScript	Semi-Auto	Migration will parse any functions/ expressions and form an approximate Orchestration equivalent. Some instances will need manual changes.

Macro Miscellaneous Object

Composer migrates the Macro object to ECMAScript.

IRD Source Element	Composer Target Element	Transformation	Comments
Macro Object	ECMAScript	Manual	This object is converted to an ECMAScript block but the user is expected to fill in the equivalent ECMAScript code.

Error Segmentation Miscellaneous Object

Composer migrates the Error Segmentation object as block exception ports.

IRD Source Element	Composer Target Element	Transformation	Comments
Error Segmentation Object	Composer block exception ports	Manual	Error Segmentation is not supported as the Orchestration platform does not support the same error codes as IRD did. Therefore, this object needs to be migrated manually.

Call Subroutine Object

Composer migrates the Call Routine object to a Subroutine block.

IRD Source Element	Composer Target Element	Transformation	Comments
Call Subroutine Object	Subroutine Block	Semi-Auto	Parameter names need to be manually filled in since they are not picked up by the migration wizard

Multi-Assign Object

Composer migrates the Multi-Assign object to an Assign object.

IRD Source Element	Composer Target Element	Transformation	Comments
Multi-Assign Object, Expression property	Assign block Assign Data Property.	Auto	

Multi-Attach Object

Composer migrates the Multi-Attach object to an UserData block.

IRD Source Element	Composer Target Element	Transformation	Comments
Business Attributes, Attach, Update, Requested Skills	Business Atributes: Composer will migrate to User Data block, Assign Data property. Attach, Update, Requested Skills: May be translated to an ECMAScript block or an Assign block depending on whether there is an assignment or not. If there is an assignment, migration will create an equivalent Assign block else it will create an ECMAScript block.	Auto	

Multi-Function Object

Composer migrates the Multi-Function object to ECMAScript block.

IRD Source Element	Composer Target Element	Transformation	Comments
Multi-Function Object	ECMAScript	Semi-Auto	Migration will parse any functions/ expressions and form an approximate Orchestration equivalent. Some instances will need manual changes.

Routing Objects Migration

Selection Object

Composer migrates IRD's Selection object to an instance of the Target block. Each busy treatment defined in the IRD object is converted to an independent Treatment block and connected to the Busy Treatments outport of the Target block.

IRD Source Element	Composer Target Element	Transformation	Comments
Additional Threshold Property	Threshold attribute of Target	IRD value will be copied to each specified Target	Orchestration has target-specific thresholds rather than a common threshold for all targets.
Use Treatments Property	Use Treatments Property	None. Both source and target properties are Boolean.	
Busy Treatments Property	Independent Busy Treatment block	Each busy treatment will become a block.	
Busy Busy Treatment	Play Sound Block	See Busy object section.	
Cancel Call Busy Treatment	Cancel Call Block	See Voice Treatments section	
Collect Digits Busy Treatment	UserInput block	See Voice Treatments section	
Delete User Announcement Busy Treatment	Delete User Announcement block	See Voice Treatments section	
Exit Busy Treatment	-		
Fast Busy Busy Treatment	Play Sound block	See Voice Treatments section	
IVR Busy Treatment	IVR Block	See Voice Treatments section	
Music Busy Treatment	Play Sound block	See Voice Treatments section	
Pause Busy Treatment	Play Sound block	See Voice Treatments section	
Play Announcement	Play Message	See Voice Treatments section	

IRD Source Element	Composer Target Element	Transformation	Comments
Busy Treatment			
Play Announcement and Collect Digits Busy Treatment	UserInput block	See Voice Treatments section	
Play Application Busy Treatment	Play Application block	See Voice Treatments section	
RAN Busy Treatment	Play Sound block	See Voice Treatments section	
Record User Announcement Busy Treatment	Create User Announcement block	See Voice Treatments section	
Ringback Busy Treatment	Play Sound block	See Voice Treatments section	
Set Default Destination Busy Treatment	Set Default Route block	Refer to Default Route section below.	
Silence Busy Treatment	Play Sound block	See Voice Treatments section|	
Text to Speech Busy Treatment	Play Message	See Voice Treatments section	
Text to Speech and Collect Digits Busy Treatment	UserInput block	See Voice Treatments section	
Verify Digits Busy Treatment	UserInput block	See Voice Treatments section	
Statistics Min/Max property	Statistics Order property		
(Statistic) Name	Statistic property		
Clear Target property	Clear Targets property		
Timeout property	Timeout property		
Target property	Targets property		Property is multi-valued with 3 sub-properties in IRD and 4 in Composer. The additional entry in Composer is Threshold. See Additional Threshold Property above.
Type sub-property	Type sub-property		
Name sub-	Name sub-		

IRD Source Element	Composer Target Element	Transformation	Comments	
property	property			
StatServer sub-property	StatServer sub-property			
Use Virtual Queue property	NA		Implicitly controlled by the "Virtual Queue Alias" property. If not null, the value of this property is assumed to be true else false.	
Virtual Queue Alias property	Virtual Queue property			
Virtual Queue Switch property				
Virtual Queue Number property				

Default Route Object

Composer migrates IRD's Default Route object to an instance of the Default Route block, which routes to the default destination.

IRD Source Element	Composer Target Element	Transformation	Comments
IRD object does not have properties.		Neither the Source element nor the Target element have properties.	

Force Object

IRD's Force routing object forces the interaction to the target specified in the selected Force routing, without any other operations.

IRD Source Element	Composer Target Element	Transformation	Comments
		Orchestration does not support force routing. Suggest converting to Force Route <ixn:redirect>	

Load Balancing (Routing Rule) Object

IRD's Load Balancing object distributes interactions evenly to targets according to estimated wait time.

IRD Source Element	Composer Target Element	Transformation	Comments
Load Balancing Rule Name	Rule Name.	None. Both the values are String.	The Rule Name property is directly mapped. The Rule Type property is set to "Load Balancing".

Percentage (Routing Rule) Object

The Percentage object uses a specified percentage for distributing interactions among several targets.

IRD Source Element	Composer Target Element	Transformation	Comments
Percentage Rule name.	Rule Name	None. Both the values are String.	Rule Name property is directly mapped. Rule Type property is set to "Load Balancing".

Statistics (Routing Rule) Object

The Statistics object uses a specified statistic for routing interactions. URS uses the values of defined statistics from Stat Server.

IRD Source Element	Composer Target Element	Transformation	Comments
Statistics Rule name.	Rule Name	None. Both the values are String.	Rule Name property is directly mapped. Rule Type property is set to "Statistic".

Segmentation Objects Migration

Used to cause interactions to take different paths in a strategy or subroutine. Note: The 8.1 release of Composer does not support the following IRD Segmentation objects: Business, Classify, and Screen.

Generic Segmentation Object

Composer migrates the IRD Generic object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	None. Both source and target properties are a list of expressions.	Refer to the Functions page.

Date Segmentation Object

Composer migrates the Date object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	Refer to the Functions section	
Time Zone Property	Function argument	Refer to the Functions section	Composer representation, i.e., 'PST'

Time Segmentation Object

Composer migrates the Time object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	None. Both source and target properties are a list of expressions.	
Time Zone Property	Function argument	Refer to the Functions section	Composer representation, i.e., 'PST'

Day of Week Segmentation Object

Composer migrates the Day of Week object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	None. Both source and target properties are a list of expressions.	
Time Zone Property	Function argument	Refer to the Functions section	Composer representation, i.e., 'PST'
Advanced Time Zone	-	Boolean value will determine if there is a day range, "From" day and "To" day.	

ANI Segmentation Object

Composer migrates the ANI Segmentation object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	None. Both source and target properties are a list of expressions.	Refer to the Functions page.

DNIS Segmentation Object

Composer migrates the DNIS Segmentation object to an instance of the Branching block.

IRD Source Element	Composer Target Element	Transformation	Comments
Segment Property	Conditions Property	None. Both source and target properties are a list of expressions.	Refer to the Functions page.

Voice Treatment Objects Migration

For the most part, Composer has a 1:1 mapping with IRD Treatment objects; however, migration combines similar Treatment objects into one block. Busy treatment properties from IRD's Selection object are mapped to Treatment objects as per the mapping rules defined below.

- Note: Composer Busy Treatments are provided through the same Treatment blocks used for Mandatory Treatments.

Busy Object

Composer migrates IRD's Busy object to an instance of the PlaySound block.

IRD Source Element	Composer Target Element	Transformation	Comments
	Sound Type property = BusyTone	Implicit SoundType setting to reflect the IRD block type.	
Duration Property	Duration Property	None. Both source and target properties are integers.	

Fast Busy Object

Composer migrates IRD's Fast Busy object to an instance of the PlaySound block.

IRD Source Element	Composer Target Element	Transformation	Comments
	Sound Type property = FastBusyTone	Implicit SoundType setting to reflect the IRD block type.	
Duration Property	Duration Property	None. Both source and target properties are integers.	

Music Object

Composer migrate IRD's Music object to an instance of the Play Sound block.

IRD Source Element	Composer Target Element	Transformation	Comment
-	Sound Type property = Music	Implicit SoundType setting to reflect the IRD block type.	
Duration	Duration	Direct Mapping	
MUSIC_DN	Resource	Direct Mapping	

Ringback Object

Composer migrates IRD's Ringback object to an instance of the Play Sound Block.

IRD Source Element	Composer Target Element	Transformation	Comment
-	Sound Type property = Ringback	Implicit SoundType setting to reflect the IRD block type.	
Compatible Mode	Compatibility Mode	Direct Mapping	New property in Composer block
Duration	Duration	Direct Mapping	

Silence Object

Composer migrates IRD's Silence object to an instance of the Play Sound Block.

IRD Source Element	Composer Target Element	Transformation	Comment
-	Sound Type property = Silence	Implicit SoundType setting to reflect the IRD block type.	
Compatible Mode	Compatibility Mode	Direct Mapping	New property in Composer block
Duration	Duration	Direct Mapping	

Pause Object

Composer migrates IRD's Pause object to an instance of the Pause Block.

IRD Source Element	Composer Target Element	Transformation	Comment
Duration	Duration	Direct Mapping	

RAN Object

Composer migrates IRD's RAN object to an instance of the Play Sound Block.

IRD Source Element	Composer Target Element	Transformation	Comment
-	Sound Type property = RAN	Implicit SoundType setting to reflect the IRD block type.	
ROUTE	Resource	Direct Mapping	
Duration	Duration	Direct Mapping	

Set Default Destination Object

Composer migrates IRD's Set Default Destination object to an instance of the Set Default Route Block.

IRD Source Element	Composer Target Element	Transformation	Comment
Destination	Destination	Direct Mapping	

Play Announcement Object

Composer migrates IRD's Play Announcement object to an instance of the Play Message Block.

IRD Source Element	Composer Target Element	Transformation	Comment
-	Type Of Prompts property = Announcement		Type property implicitly set to Announcement since block supports functions of multiple IRD blocks
Language	Language	Direct Mapping	
Wait for Treatment End	-	Not supported	
MSGID	Map to Prompt element : MSGID	Direct Mapping	New type added to Composer.
MSGTXT	Map to Prompt element: MSGTXT	Direct Mapping	New type added to Composer.
PROMPT	See Prompts Element Migration	Direct Mapping	

Prompts Element

Many IRD objects use prompt elements. IRD prompt values are migrated to Composer prompt elements.

IRD Source Element	Composer Target Element	Transformation	Comments
Interruptible	Interruptible	Direct Mapping	
ID	Value	Type = Announcement	
Digits	Value	Type = Formatted Digits	
User_Ann_ID	Value	Type = User Announcement	
Text	Value	Type = Text	

Play Announcement and Collect Digits

Composer migrates IRD's Play Announcement & Collect Digits object to an instance of the User Input block.

In the User Input block, you must specify a variable to hold the collected digits.

IRD Source Element	Composer Target Element	Tranformation	Comments
-	Type of Prompts = Announcement	Implicit setting	User Input block wraps multiple IRD objects
Wait for Treatment End	Wait for Treatment End	Direct Mapping	
Language	Language	Direct Mapping	
MAX_DIGITS	Number of Digits		
ABORT_DIGITS	Abort Digits	Direct Mapping	
IGNORE_DIGITS	Ignore Digits	Direct Mapping	
BACKSPACE_DIGITS	Backspace Digits	Direct Mapping	
TERM_DIGITS	Termination Digits	Direct Mapping	
RESET_DIGITS	Reset Digits	Direct Mapping	
CLEAR_DIGITS	Clear Input	Direct Mapping	
START_TIMEOUT	Start Timeout	Direct Mapping	
DIGIT_TIMEOUT	Digit Timeout	Direct Mapping	
TOTAL_TIMEOUT	Total Timeout	Direct Mapping	
MSGID	Map to Prompt element : MSGID	Direct Mapping	
MSGTXT	Map to Prompt element: MSGTXT	Direct Mapping	

IRD Source Element	Composer Target Element	Tranformation	Comments
Prompt	See Prompts Element Migration		

Text to Speech & Collect Digits

Composer migrates IRD's Text to Speech & Collect Digits object to an instance of the User Input block.

IRD Source Element	Composer Target Element	Transformation	Comments
-	Type of Prompts = Text to Speech	Implicit setting	User Input block wraps muliple IRD objects
Language	Language	Direct Mapping	
MAX_DIGITS	Number of Digits	Direct Mapping	
ABORT_DIGITS	Abort Digits	Direct Mapping	
IGNORE_DIGITS	Ignore Digits	Direct Mapping	
BACKSPACE_DIGITS	Backspace Digits	Direct Mapping	
TERM_DIGITS	Termination Digits	Direct Mapping	
RESET_DIGITS	Reset Digits	Direct Mapping	
CLEAR_DIGITS	Clear Input	Direct Mapping	
START_TIMEOUT	Start Timeout	Direct Mapping	
DIGIT_TIMEOUT	Digit Timeout	Direct Mapping	
TOTAL_TIMEOUT	Total Timeout	Direct Mapping	
Prompt	See Prompts Element Migration above		

Play Application Object

Composer migrates IRD's Play Application object to an instance of the Play Application block.

IRD Source Element	Composer Target Element	Transformation	Comments
Wait for Treatment End	Wait for Treatment End	Direct Mapping	
Language	Language	Direct Mapping	
APP_ID	Resource (Type = Id)	Direct Mapping	
Parameters List<>	Parameters List<>	Direct Mapping	

Record User Announcement Object

Composer migrates IRD's Record User Announcement object to an instance of the Create User Announcement block.

IRD Source Element	Composer Target Element	Transformation	Comments
Wait for Treatment End	Wait for Treatment End	Direct Mapping	
ABORT_DIGITS	Abort Digits	Direct Mapping	
TERM_DIGITS	Termination Digits	Direct Mapping	
RESET_DIGITS	Reset Digits	Direct Mapping	
START_TIMEOUT	Start Timeout	Direct Mapping	
TOTAL_TIMEOUT	Total Timeout	Direct Mapping	
Prompt	See Prompts Element		

Delete User Announcement

Composer migrates IRD's Delete User Announcement object to an instance of the Delete User Announcement block.

IRD Source Element	Composer Target Element	Transformation	Comments
USER_ID	User Id	Direct Mapping	
USER_ANN_ID	Announcement Id	Direct Mapping	

IVR Object

Composer migrates IRD's IVR object to an instance of the IVR block.

IRD Source Element	Composer Target Element	Transformation	Comments
Compatible Mode	Compatibility Mode	Direct Mapping	
LABEL	Remote Resource	Direct Mapping	Compatible Mode = False
DNIS	Failover Resource	Direct Mapping	Compatible Mode = False
SCRIPT	Application	Direct Mapping	Compatible Mode = True
TARGET	Remote Resource	Direct Mapping	Compatible Mode = True
DURATION	Treatment Duration	Direct Mapping	Compatible Mode = True

Text to Speech Object

Composer migrates IRD's Text to Speech object to an instance of the Play Message block.

IRD Source Element	Composer Target Element	Tranformation	Comments
Language	Language	Direct Mapping	
Prompt	Prompt	See Prompts Element above	

Verify Digits Object

Composer migrates IRD's Verify Digits object to an instance of the User Input block.

IRD Source Element	Composer Target Element	Transformation	Comments
Wait for Treatment End	Wait for Treatment End	Direct Mapping	
Language	Language	Direct Mapping	
COMPARE_DIGITS	Verification Data	DTMF Verification Option = Compare Digits	
COMPARE_USER_ID	Verification Data	DTMF Verification Option = Local Table Lookup	
COMPARE_PLAN_ID	Verification Data	DTMF Verification Option = Compare Dialing Plan Format	
NUM_ATTEMPTS	Verification Attempts	Direct Mapping	
NUM_DIGITS	Number of Digits	Direct Mapping	
ABORT_DIGITS	Abort Digits	Direct Mapping	
IGNORE_DIGITS	Ignore Digits	Direct Mapping	
BACKSPACE_DIGITS	Backspace Digits	Direct Mapping	
TERM_DIGITS	Termination Digits	Direct Mapping	
RESET_DIGITS	Reset Digits	Direct Mapping	
CLEAR_DIGITS	Clear Input	Direct Mapping	
START_TIMEOUT	Start Timeout	Direct Mapping	
DIGIT_TIMEOUT	Digit Timeout	Direct Mapping	
TOTAL_TIMEOUT	Total Timeout	Direct Mapping	
Prompt	Prompts	See Prompts Element above	
Reprompt	Retry Prompts	See Prompts Element above	

IRD Source Element	Composer Target Element	Transformation	Comments
Success	Success Prompts	See Prompts Element above	
Failure	Failure Prompts	See Prompts Element above	

IRD Variable Handling

In IRD 8.0.1, additional variable scopes have been introduced that do not exist within Orchestration. These new scopes for variables are not supported because of the significant ramifications on how variables can be accessed and/or defined with Orchestration Server's ability to operate in a fully distributed, multi-threaded environment.

Orchestration only supports LOCAL scoped variables as these map directly to variables that can be created within SCXML. For the remaining sets of more global scopes exposed by IRD, the following table provides the current recommended approach. In general, the use cases for these are seen to be more business application related and as such should be stored within a centralized database that can manage concurrent access and transactional-based locking, or by other similar means that can provide global access control to data.

Variable Scope	Description	Migration Notes
SCRIPT	A SCRIPT variable is created when the strategy is preloaded and is destroyed when the strategy is released from URS's memory. Every run of the strategy can change the value of the same SCRIPT variable. Use SCRIPT variables with caution. (One possible use is as counters over all interactions processed by the same strategy.) Values are shared within a single strategy.	If the SCRIPT variable is only used for rea purposes, then this could be populated w the provision definition and provided whe script is invoked. However, this variable then need to be passed as a parameter t subroutines or other workflows. If the va is not read only, then a DB block or othe mechanism would need to be used.
USER	A USER variable is created when the tenant is active and is destroyed when the tenant is released from URS's memory. Everything running within the tenant can change the value of the same USER variable. Values are shared within the current tenant.	Since a single ORS node may provide su for multiple tenants, it is recommended t this variable be stored and operated upo a centralized DB accessed via the DB blo
GLOBAL	A GLOBAL variable is created when the particular instance of URS runs and is destroyed when the URS stops running or is released from memory. Everything running on this instance of URS can change the value of the same GLOBAL variable. Values are shared within the entire URS instance.	Since Sessions may be swapped betwee Orchestration nodes, this is not really a r concept within Orchestration because se are not sticky. Any session can be execut a multitude of nodes through its life cycl There is currently no other recommenda this other than possibly a centralized DB via DB blocks
INTERACTION	An INTERACTION variable is created when a particular interaction is active and is destroyed when the interaction ends. The value of an INTERACTION variable for one interaction has no effect on the value of the same INTERACTION variable for another interaction. Values are shared for the current interaction only	Since this is related to the actual interac udata can be used to accomplish this tas would ensure that the data is shared acr Orchestration Nodes as well as any other component whenever an interaction is o on.

Composer Block and Exception Naming

IRD objects and exceptions do not have names whereas Composer has Block names and Exception names. Hence the following naming convention will be followed during the process of migration for naming migrated blocks.

Composer blocks created for every IRD object will be named using the default naming convention which follows the format <Composer block type>n. If the diagram has multiple blocks of the same category the block name will gets incremented. e.g. Target1, Target2,... TargetN

If an IRD Object has an Exception/Error port, the corresponding Composer block will get the major exception added in the Block. For e.g. When "Selection" block in IRD gets migrated to "Target" block "error.queue.submit" will get added, if the "Selection" Object has the Error port connected to another Object. Please refer to the Major Exception table at the end of this section. It would be upto the user to check exception handling and hook up the appropriate exception event. A new property "Notes" will be introduced in all Composer blocks. Migrated blocks in Composer will have their "Notes" property set to the Notes for their IRD equivalent. Additionally, "Notes" property of migrated blocks will also specify the type of IRD block it was migrated from. Major Exceptions describes the default major exception for the Composer Workflow blocks.

Major Exceptions

The table below describes the default major exceptions assigned in Composer workflow blocks. The exceptions appear in the Exceptions dialog box, which opens from the block's Exception Property.

Assignment of Major Exceptions

Composer Block	Exception Assigned	Comments
Target	error.queue.submit	
Default Route	error.queue.default	
Force Route	error.interaction.redirect	
Routing Rule	error.queue.submit	
Play Sound	error.dialog.playsound	
Play Application	error.dialog.start	
Play Message	error.dialog.play	
User Input	error.dialog.collect	
Set Default Route	error.dialog.setdialogdefaultdest	
Pause	error.send.failed	
CreateUserAnnouncement	error.dialog.createann	
DeleteUserAnnouncement	error.dialog.deleteann	
IVR	error.dialog.remote	
Cancel Call	error.dialog.stop	-