Working with the iWD Business Process in Composer

intelligent Workload Distribution 9.0.0

12/30/2021

Table of Contents

Working with the IWD Business Process in Composer

This Document

The topics in this document describe how to work with the iWD Business Process (`iwd_bp_comp`) for Composer/ORS that is supplied with intelligent Workload Distribution in release 9.0.x.

Working with the iWDBP

The iWD Business Process is made up of a set of Interaction Queues that map to the iWD state model.

- `iwd_bp_comp.Main.iWD_New`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Captured`
- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_ErrorHeld`
- `iwd_bp_comp.Main.iWD_Queued`
- `iwd_bp_comp.Main.iWD_Rejected`

The Interaction Queues that are included in the out-of-box `iwd_bp_comp` business process must be present. The names can be changed, but it is necessary to appropriately configure iWD with new names. The Global Task List looks for the defined Interaction Queue names. If you modify the business process to add additional queues or rename existing queues, and the change is not reflected in the configuration of iWD, the interactions display in the Global Task List with the status `Queued`. In this document it is assumed that original queue names are used.

Within this Business Process, from within a routing strategy, Composer blocks are used to invoke Genesys Rules Engine (GRE) functions. This approach is used to apply classification and prioritization rules to the interaction. When a user goes to the Global Task List (GTL) view in iWD Manager, to monitor the interactions that are in various states, this component communicates with Interaction Server to retrieve the list of interactions and their attributes.

This out-of-the-box Genesys iWD Business Process maps to the iWD state model, allowing you to use iWD-based reporting for other interaction types (for example, you might want to track Genesys emails along with other task types, under the same Department or Process).

This Genesys iWD Business Process is completely optional for iWD customers who are using Genesys Email, Genesys Chat, Genesys SMS, or even third-party email, SMS, or chat. If the Genesys iWD Business Process is not used, iWD Runtime Node/Data Mart and iWD Global Task List functionality may be limited.

For Genesys eServices customers, the Genesys iWD Business Process can be left unchanged if you want to use business rules only. In this scenario, what would change would be the routing strategies. The strategies would use Composer blocks to invoke the Genesys Rules Engine. This means that existing Genesys Email, Chat or SMS/MMS customers can use the business rules within iWD without having to change their Genesys Business Processes; or, to access some additional functionality, changes can be made to the Business Processes.

There is a summary of the differences between Genesys Interaction Routing Designer (IRD) and Composer **here** (new document).

Cloning the iWDBP to Create New Business Processes

You can create new business processes (under the same Tenant) that can support clear logical distinctions between processes and departments For example, interactions from different media types (email, chat, SMS and so on) can be handled by separate business processes with their own customized queue names, and this in turn can provide clear logical distinctions in reporting, because the queue name is the basis for handling reporting requirements.

This is achieved by by cloning and editing the IWDBP, and making the interaction queues (supplied out-of-box with iWD) configurable for your needs, together with some additional configuration changes.

If you configure more than one business process, customized queues must be configured for each Solution in the iWD GAX Plug-in. Only the existing queues may be used. The custom queue names will then be used by both iWD Manager and iWD Data Mart instead of the default ones.

Other Information Resources

- The **Universal Routing 8.1 Business Process User's Guide** provides an in-depth discussion of business processes.
- The **Composer 8.1.4 Deployment Guide** provides information on installing Composer, and post-installation configuration.
- The **IRD to Composer Migration 8.1.2** describes how to migrate IRD routing strategies into Composer Projects as SCXML-based workflow diagrams.
- The **Composer Help 8.1.4** describes how to create voice applications for GVP and routing applications for the Orchestration Platform.

Important

In Composer, a business process is referred to as an Interaction Process Diagram or **IPD**.

New Features by Release

9.0.011

- ORS Business Process improvements
    - Error handling: fixed several cases where error messages were not provided.

9.0.010

- ORS Business Process improvements
    - The maximum number of retries to connect GRE/UCS is now restricted for the InvokeGRE and InvokeUCS strategies in order to prevent an infinite loop if GRE/UCS is down.
    - The iWD Business Process now properly handles ORS timeouts and GRE/UCS unavailability.

9.0.004

- ORS Business Process improvements
    - A **Pause** block with a configurable delay has been added to the InvokeGRE workflow to guarantee that interaction updates will be received. By default, the delay value is set to `0`. Previously during the task life cycle, when the Genesys Rules Engine changed interaction properties, ORS did not always have enough time to receive an acknowledgement of the changes from Interaction Server and so it continued executing the workflow. This could lead to unexpected behavior. For example, tasks might go to the `ErrorHeld` queue sporadically without visible reasons. There is more information here:
    - IWDBP Strategies & Subroutines.

Prerequisites

Software Requirements

The general software prequisites for iWD must be in place, plus ORS/Composer components as follows:

- Orchestration Server 8.1.4 or higher.
- Composer 8.1.410.09 or higher.

IWDBP Contents

Strategies

The iWD business process (`iwd_bp_comp`) contains the following strategies:

- `Classification`
- `Prioritization`
- `Invoke UCS`
- `Invoke GRE`
- `Distribution`
- `MarkInteractionAsDone`
- `Removal`
- `Finish`

Subroutines

The iWD business process contains the following subroutines:

- `DetermineESPServerName`
- `DetermineRulePackageName`
- `CheckBusinessValueAndPriority`

Queues

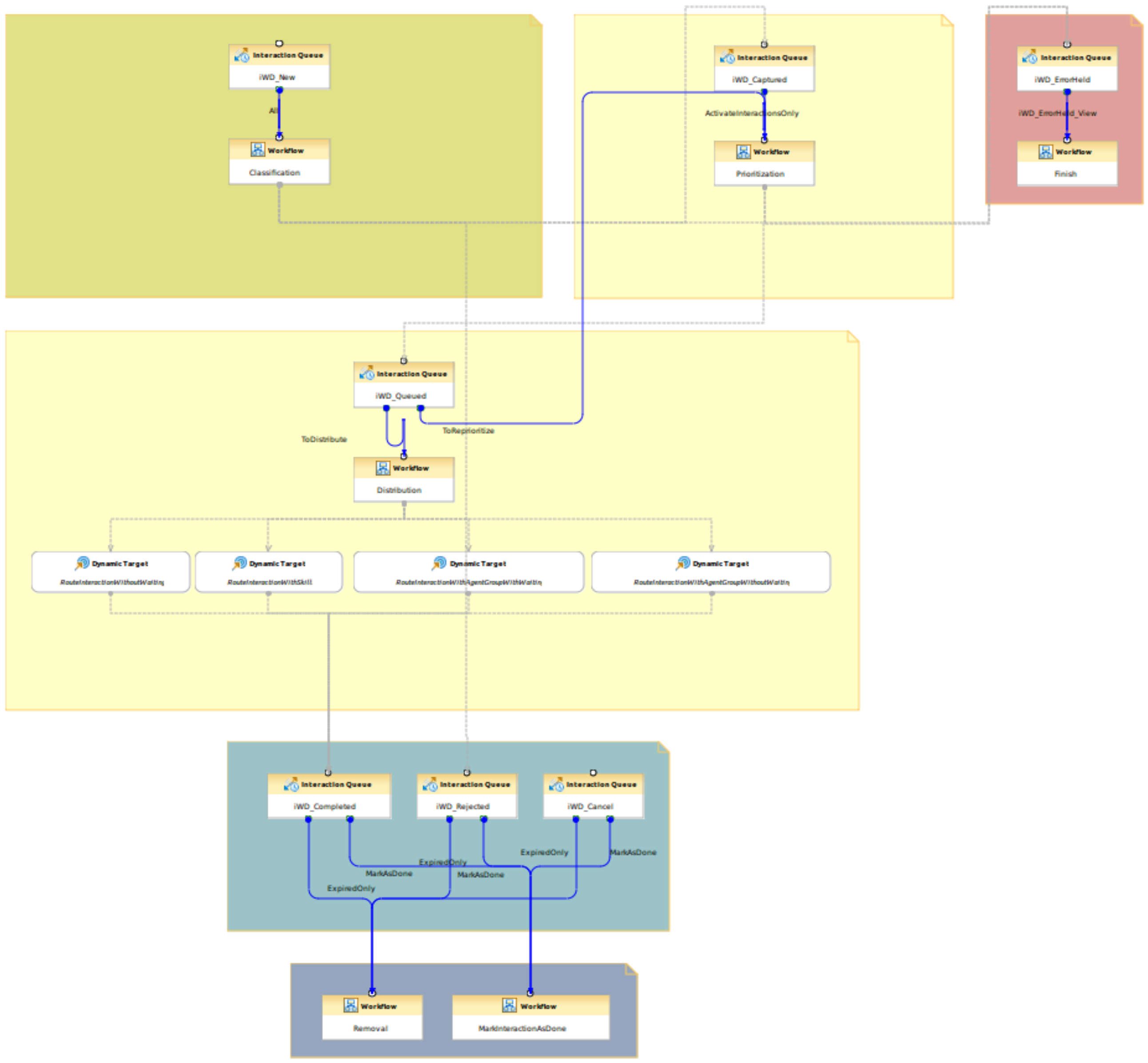
The iWD business process contains the following queues:

- `iwd_bp_comp.Main.iWD_New`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Captured`
- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_ErrorHeld`
- `iwd_bp_comp.Main.iWD_Queued`
- `iwd_bp_comp.Main.iWD_Rejected`

The Interaction Queues that are included in the out of the box `iwd_bp_comp` business process must be present. The names can be changed, but it is necessary to appropriately configure iWD with new names. The Global Task List looks for the defined Interaction Queue names. If you modify the business process to add additional queues or rename existing queues, and the change is not reflected in the configuration of iWD, the *interactions* display in the Global Task List with the status `Queued`. In this document it is assumed that original queue names are used.

IWD Main Process in Composer

The screenshot below shows the entire business process as it appears in Genesys Composer.



IWDBP Strategies & Subroutines

IWDBP Strategies & Subroutines from 9.0.011

Global variables

Timeout for userdata changes

During a task life cycle, its interaction properties are changed asynchronously. Without sufficient timeout to receive confirmation event from Interaction Server, ORS might continue workflow execution with unfinished property updates. This could lead to unexpected behavior—for example, tasks could go to the **ErrorHeld** queue sporadically without visible reasons.

To address such issues, there is a configurable timeout within UserData blocks and other places where interaction properties are changed (for example, the **AfterEspCallActivities** block in the InvokeGRE strategy). If a timeout event occurs, an interaction goes to the **ErrorHeld** queue with a corresponding error message assigned to its Userdata. This timeout could be set globally using the `vWaitTimeoutSec` variable on the project level. The default value is 10 seconds. The timeout should be calculated individually and specifically, depending on the delay in the Interaction Server response. To set its value, do the following:

1. Click on the **Entry** block of any strategy.
2. Select the **Properties** tab at the bottom of the Composer window.
3. Click on the dots next to **Global Settings -> Variables** to open the **Application Variables** window.
4. Expand the **Project Variables** item and set the `vWaitTimeoutSec` value.

Classification Strategy

The purpose of this strategy is to invoke corresponding classification rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

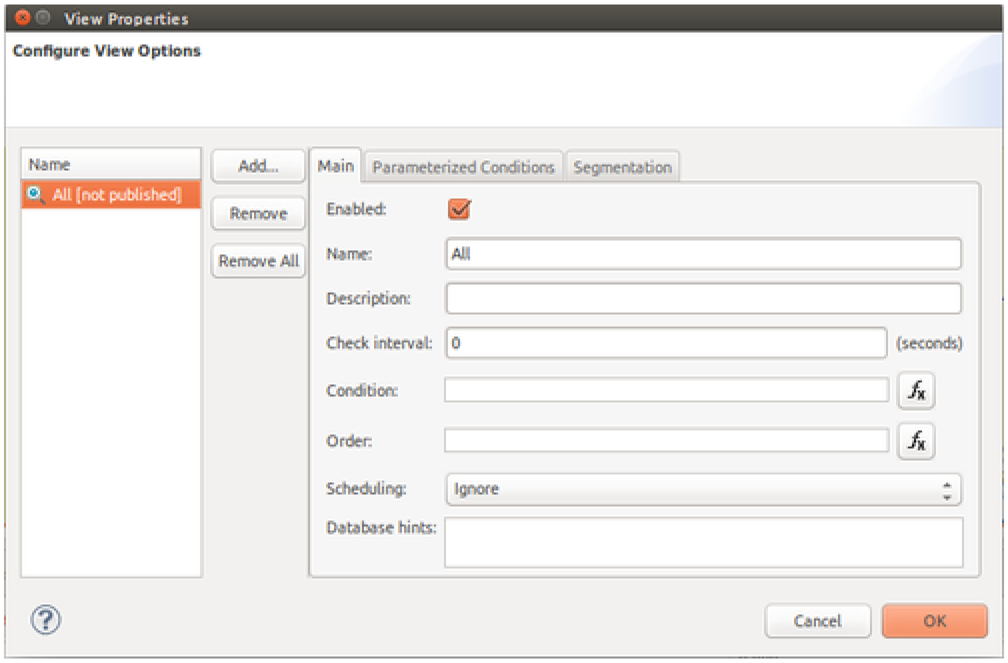
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_New`

Interactions have to satisfy the following conditions:

- There are no conditions here.
- Interactions are taken in order they were submitted.

Composer Configuration



Enabled: ✓

Name: All

Description:

Check interval: 0 (seconds)

Condition:

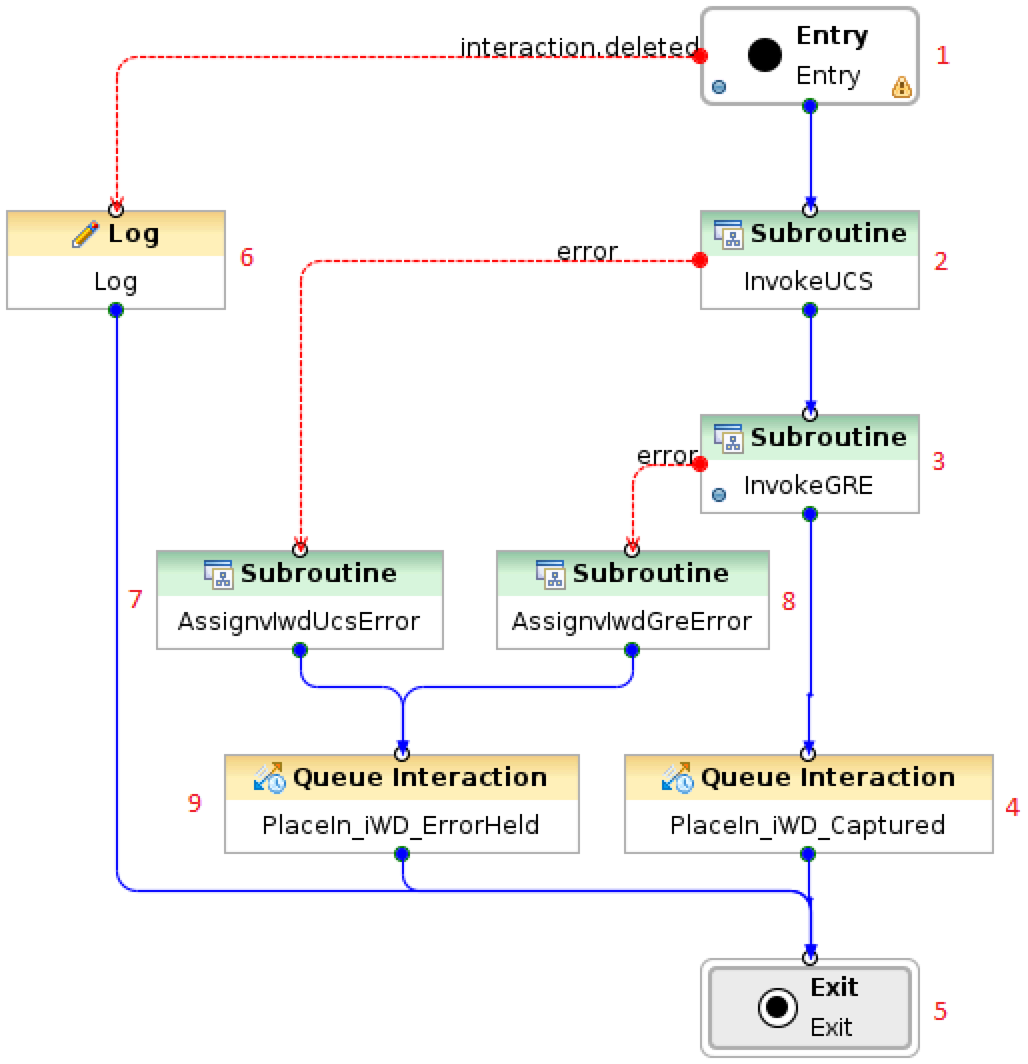
Order:

Scheduling: Ignore

Database hints:

The Composer configuration for this strategy block shows that all interactions are distributed to the `iwd_bp_comp.Main.iWD_New` queue without conditions.

Flow Summary



Flow Detail

1. Entry to Classification workflow.
2. The InvokeUCS subroutine is invoked to create new interaction in the UCS database.
3. The InvokeGRE subroutine is invoked.
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
5. Exit Classification workflow.
6. Log message in case if interaction was from some reasons deleted.

7. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`.
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine.
8. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error description that occured in InvokeGRE subroutine.
9. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

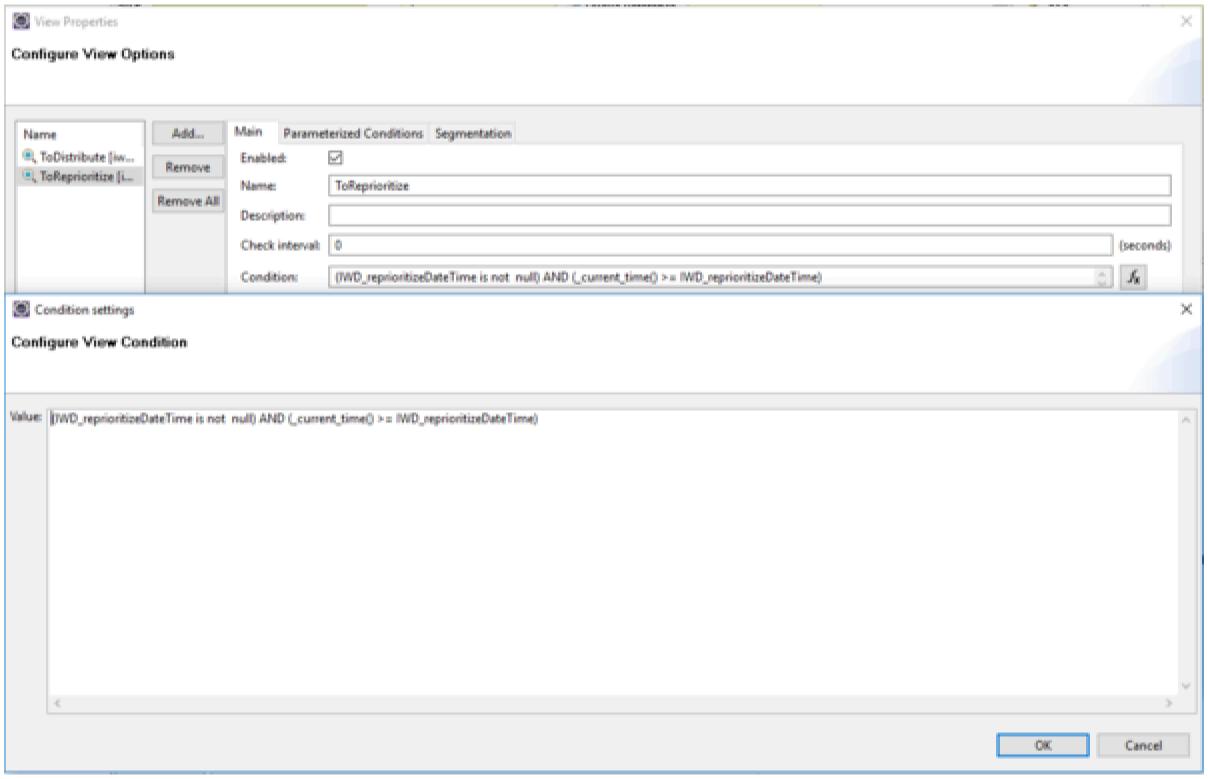
Prioritization Strategy

The purpose of this strategy is to invoke the corresponding prioritization rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Captured`—Interactions have to satisfy the following conditions:
    - Active interactions only (interactions which do not have the property `IWD_activationDateTime` set, or this property has a time stamp which is in the past.
    - Interactions are taken in the order they were submitted.
- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are subject for immediate reprioritization (interactions that have the property `IWD_reprioritizeDateTime` set to a time stamp which is in the past).
    - Interactions are taken in order of `IWD_reprioritizationDateTime` (oldest first).

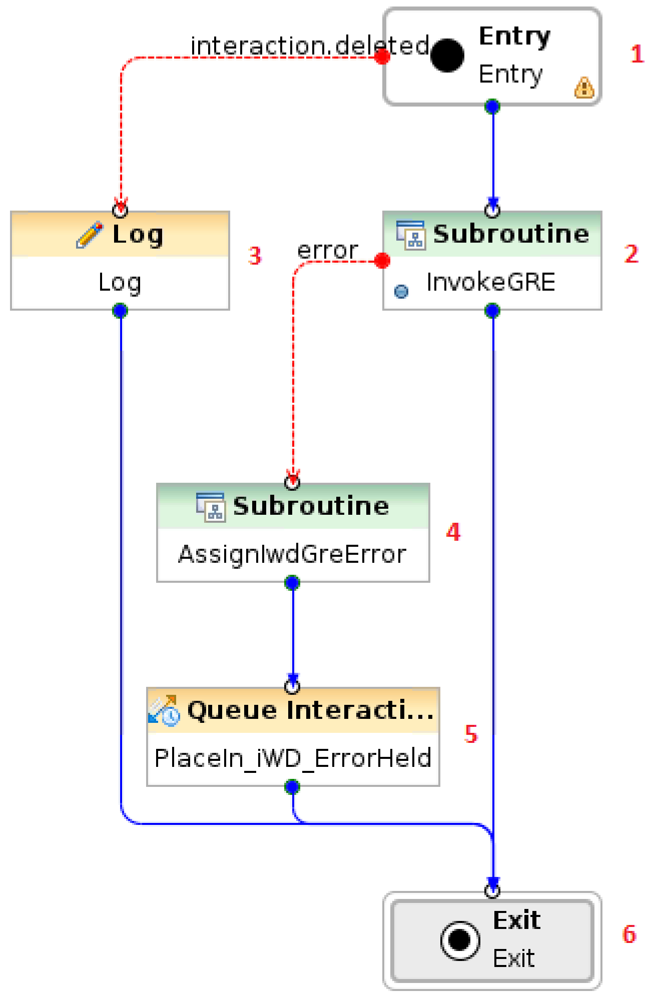
Composer Configuration



Enabled: ✓
Name: ToReprioritize
Description:
Check interval: 0 (seconds)
Condition: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Value: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Flow Summary



Flow Detail

1. Entry to Prioritization workflow.
2. The `InvokeGRE` subroutine is invoked.
3. Log message in case if interaction was from some reasons deleted.
4. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error description that occurred in `InvokeGRE` subroutine.
5. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

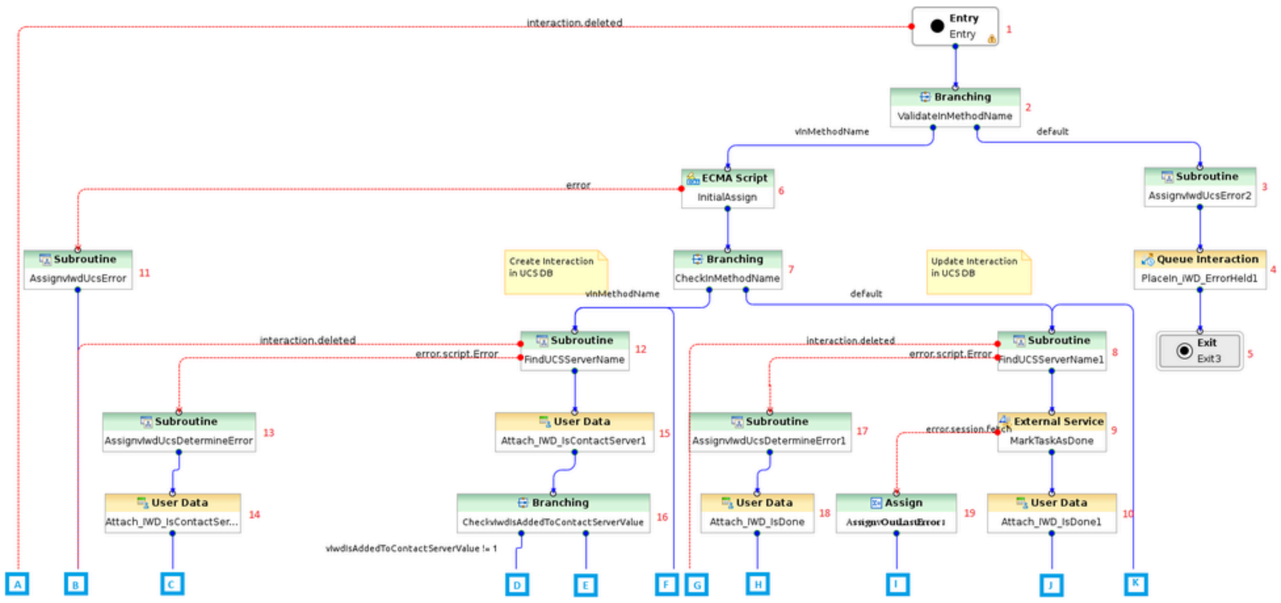
6. Exit Prioritization workflow.

Invoke UCS Strategy

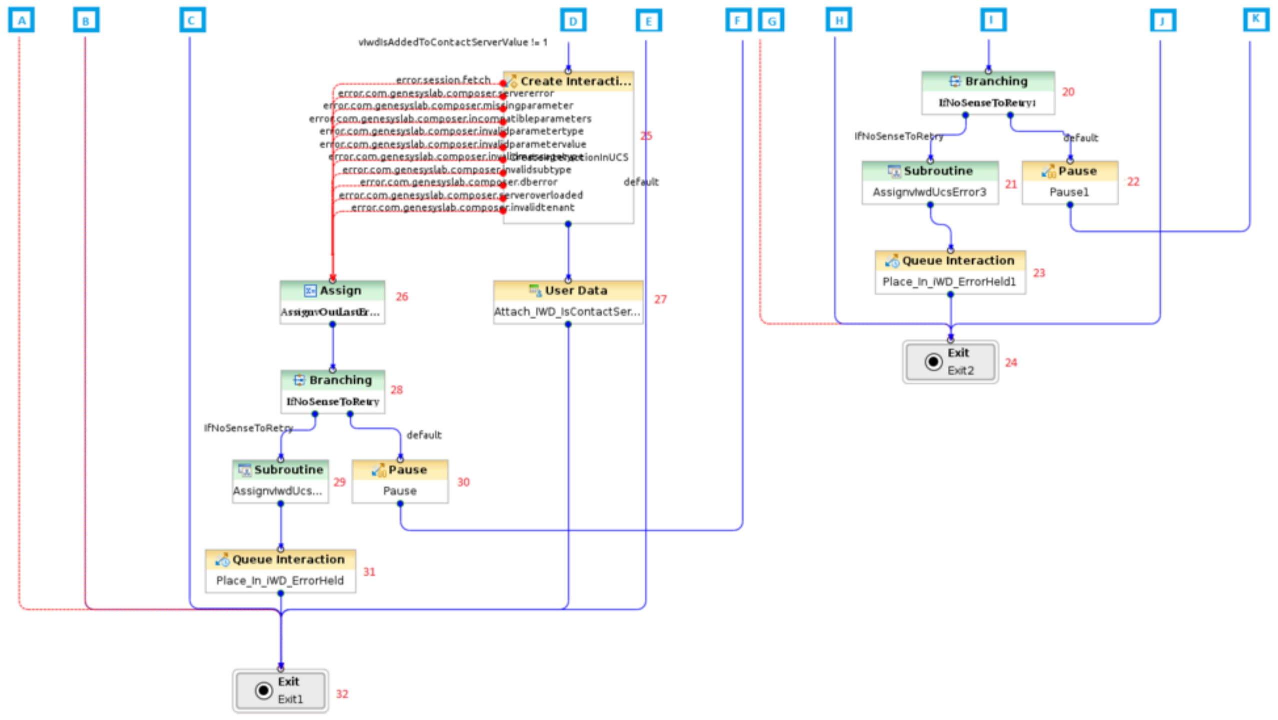
The purpose of this workflow is to create an interaction in the UCS database if UCS is configured.

Flow Summary

Part 1



Part 2



Flow Detail

1. Entry InvokeUCS strategy.
2. Check if `in_method_name = 'Create' | in_method_name = 'OMInteractions'`.
3. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Error informs that: `vInMethodName + ' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeUCS workflow.
6. Variables are initialized:
    - `vExternalId`—Read from task attribute `ExternalId`
    - `vMediaType`—Read from task attribute
    - `vSubmittedBy`—Read from task attribute `attr_itx_submitted_by`
    - `vType`—Read from task attribute `'InteractionType'`
    - `vSubType`—Read from task attribute `'InteractionSubtype'`
    - `vIwdIsAddedToContactServerValue`—Read from task attribute `'IWD_isAddedToContactServer'`

7. Check if `in_method_name = 'Create'`.
8. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
   - `vInItemName—ContactServerList`
   - `vInListName—Iwd_Esp_List`
9. The strategy calls a method on the Universal Contact Server to set the status of the interaction to 3, indicating that the interaction is done.
10. The value of the user data key `IWD_isDone` is set to 1.
11. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`'
    - `vInLastErrorString`—Error description that occurred when variables were initialized
12. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
    - `vInItemName—ContactServerList`
    - `vInListName—Iwd_Esp_List`
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`'
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
14. The value of the user data key `IWD_isContactServer` is set to `0`.
15. The value of the user data key `IWD_isContactServer` is set to 1.
16. Check if `IWD_isContactServer` is set to 1.
17. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
18. The value of the user data key `IWD_isDone` is set to `0`.
19. An error is extracted from user data and assigned in `vLastError` variable.
20. If it makes sense to retry updating the interaction record in UCS.
21. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry update interaction in UCS.
22. A delay is introduced into the processing. Flow returns to step 8.
23. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
24. Exit InvokeUCS workflow.
25. A new interaction is created in the UCS database, for this iWD task.
26. An error is extracted from user data and assigned in `vLastError` variable.

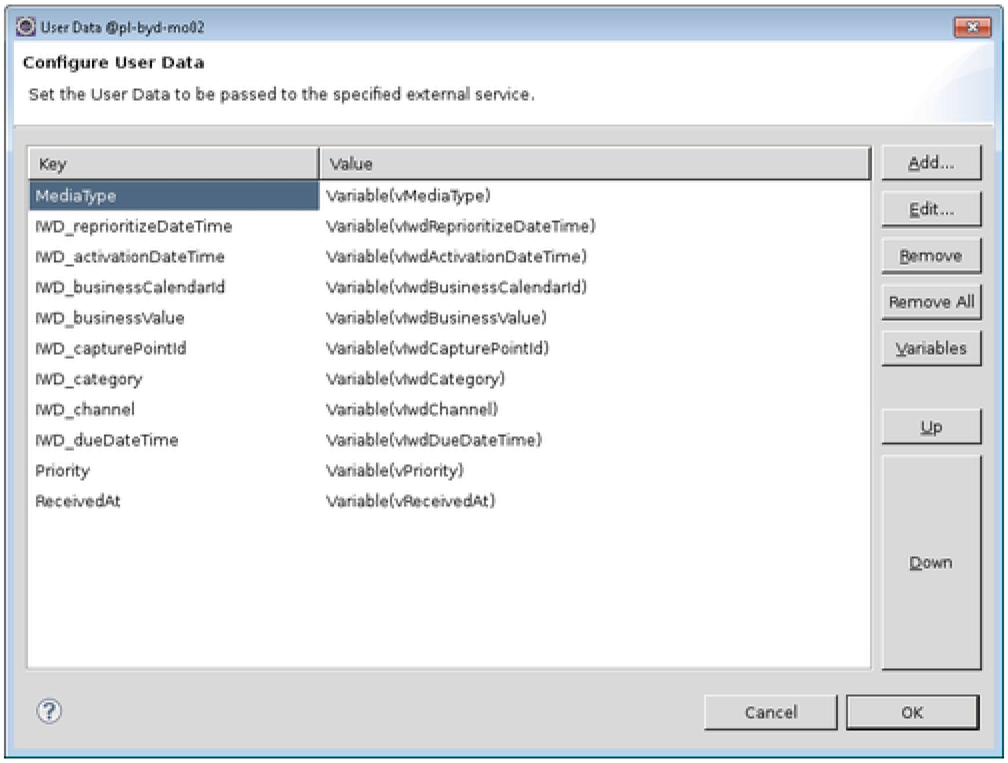
27. The user data key `IWD_isAddedToContactServer` is updated to 1 to indicate that the task was successfully added to the interaction history in UCS. The result returned from the ESP call to UCS (from See A new interaction is created in the UCS database, for this iWD task. If that function is successful is written to the variable `IWD_UCS_Result`.
28. If it makes sense to retry creating the interaction record in UCS.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry create interaction in UCS
30. A delay is introduced into the processing. Flow returns to step 12.
31. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
32. Exit InvokeUCS workflow.

InvokeGRE Strategy

Important

For Composer/ORS versions prior 8.1.400.48—If custom task attributes will be used in the Standard Rules Template, you must add them in the External Service block called `InvokeGRE` in the InvokeGRE workflow. All user-defined attributes need to be added in the User Data attribute, otherwise they will not be attached to the task and so will not be sent in the ESP request to the external ESP service.

Composer configuration



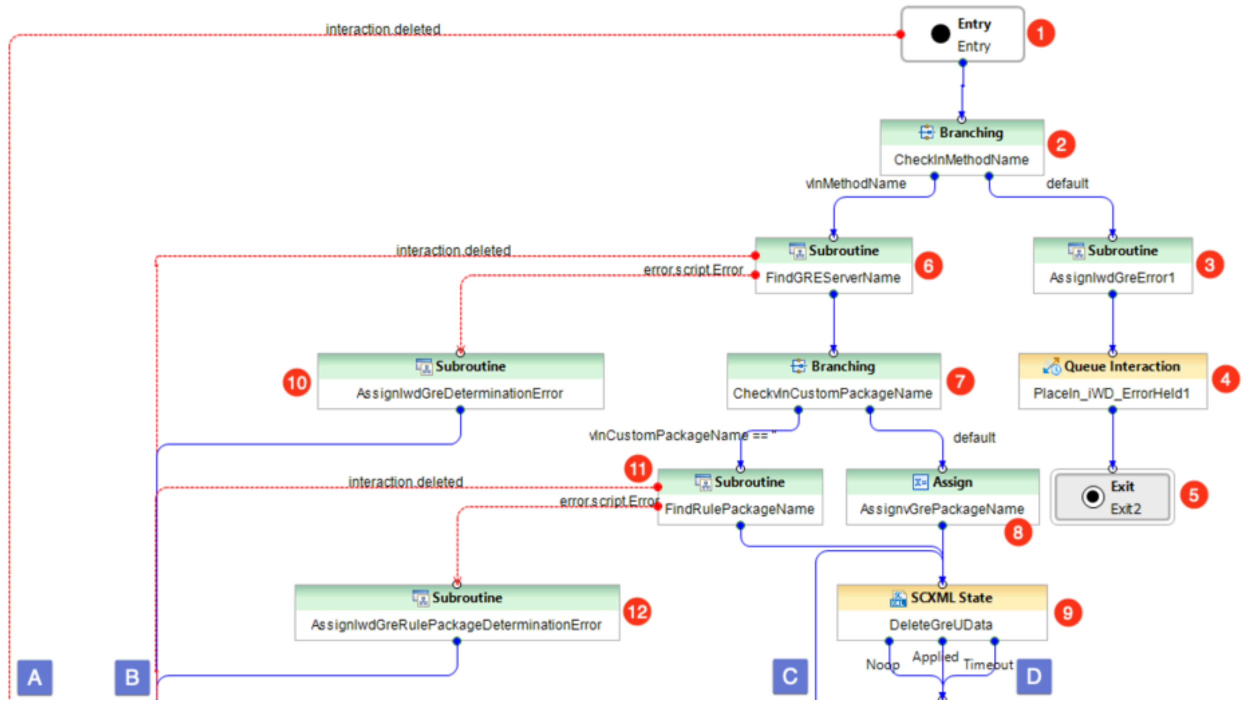
Configure User Data

Set the User Data to be passed to the specified external service.

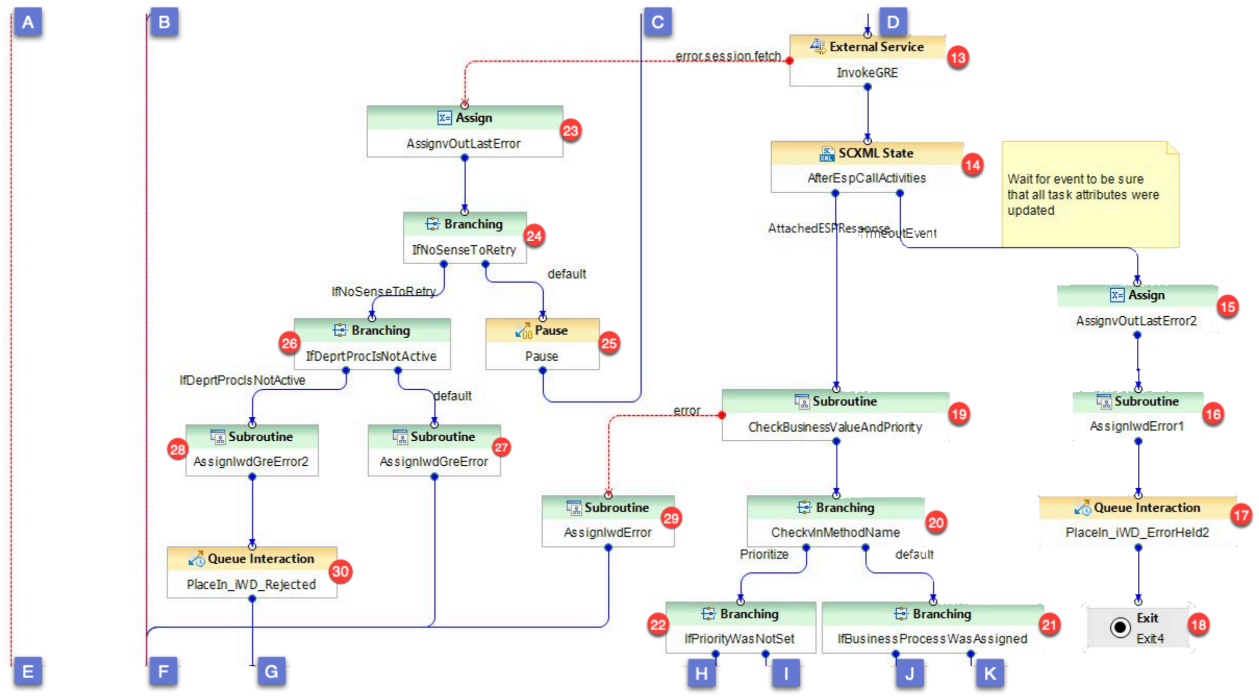
Key	Value
MediaType	Variable(vMediaType)
IWD_reprioritizeDateTime	Variable(vIwdReprioritizeDateTime)
IWD_activationDateTime	Variable(vIwdActivationDateTime)
IWD_businessCalendarId	Variable(vIwdBusinessCalendarId)
IWD_businessValue	Variable(vIwdBusinessValue)
IWD_capturePointId	Variable(vIwdCapturePointId)
IWD_category	Variable(vIwdCategory)
IWD_channel	Variable(vIwdChannel)
IWD_dueDateTime	Variable(vIwdDueDateTime)
Priority	Variable(vPriority)
ReceivedAt	Variable(vReceivedAt)

Flow Summary

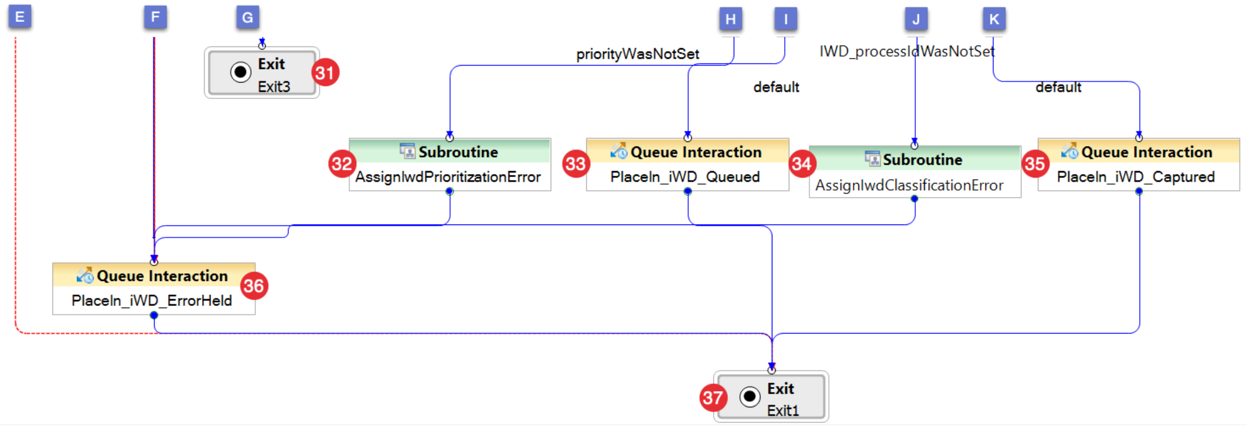
Part 1



Part 2



Part 3



Flow Detail

1. Entry to `InvokeGRE` strategy.
2. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.

3. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `vInMethodName` + `' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeGRE workflow.
6. The `FindListObjectItem` subroutine is invoked to determine the name of the Genesys Rules Engine Application. The subroutine uses the List Object list `GREServerList`:
    - `vInItemName`—`GREServerList`
    - `vInListName`—`Iwd_Esp_List`
7. Check if `vInCustomPackageName` was published to this subroutine. If it is set then `vInCustomPackageName` will be run. Otherwise package name needs to be found in `Iwd_Package_List`.
8. Assign `vInCustomPackageName` to `vGrePackageName`.
9. Delete GRE-related user data (`IWD_GRE_Result`, `IWD_GRE_Error`, `RulePhase`, `NumberOfRulesApplied`, `RulesApplied`, `PackagesApplied`) before Invoke GRE.
10. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
11. The `FindListObjectItem` subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules:
    - `vInItemName`—`RulePackageList`
    - `vInListName`—`Iwd_Package_List`
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Rule_Package_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
13. An ESP request is sent to the Genesys Rules Engine to evaluate the classification rules.

Important
All user data that needs to be added to ESP request must be added in User Data attributes.

14. Parse ESP result and attach to the interaction all attributes modified by the GRE.
15. Assign the string `AfterEspCallActivities timeout` to the `vLastError` variable.
16. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `Attach GreResult timeout`
17. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

18. Exit InvokeGRE workflow.
19. CheckBusinessValueAndPriority subroutine is called to verify if `IWD_businessValue` and `Priority` have correct values.
20. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
21. Check if `IWD_processId` was set by any rules or when task was created.
22. Check is made to see if this is the first time that prioritization rules are being evaluated for the interaction, and the priority was not set up by any rules.
23. Get last error that was occured in GRE call and assign it to `vLastError` variable.
24. A check is done to see if the error code is related to the ESP server communication.
25. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to step 11 to retry the connection to the ESP server.
26. The last Interaction Server-related error is extracted from a variable.
27. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
28. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable
30. The interaction is placed in the `iwd_bp_comp.Main.iWD_Rejected` queue.
31. Exit InvokeGRE workflow.
32. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Prioritization_Error`
    - `vInLastErrorString`—Error description: `Priority is not set up by rules`.
33. The interaction is placed in the `iwd_bp_comp.Main.iWD_Queued` queue.
34. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`vIwdClassificationError`
    - `vInLastErrorString`—Error description: `Process was not set by rules`.
35. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
36. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
37. Exit InvokeGRE workflow.

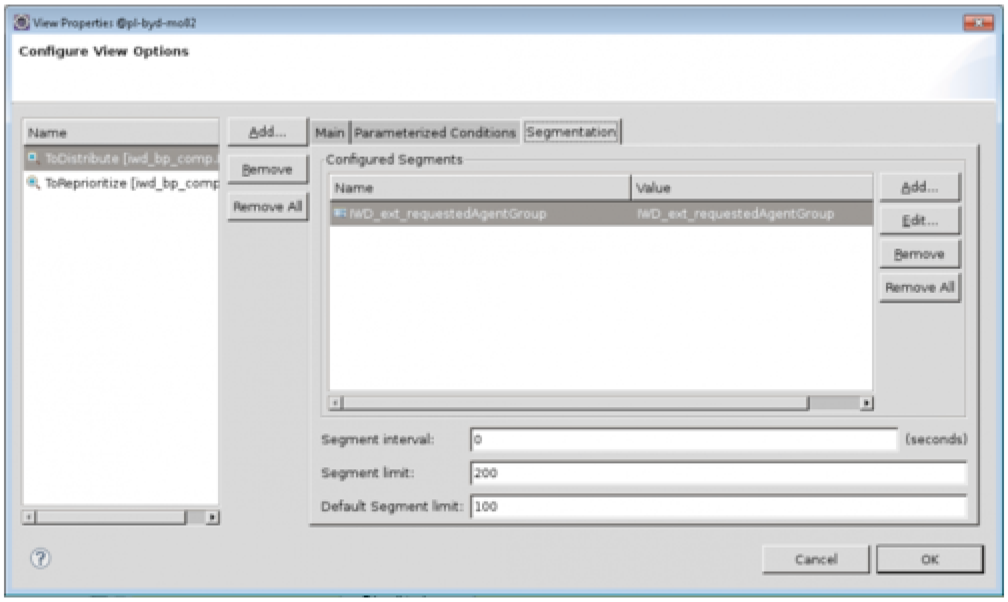
Distribution Strategy

This strategy routes interactions to a requested Agent, requested Agent Group, requested Skill, or to the default iWD Agent Group. This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are not subject for immediate reprioritization (interactions that do not have the property `IWD_reprioritizeDateTime` set, or that have this property set to a time stamp that is in the future).
    - Interactions are taken in order of priority (highest priority first)

A Segmentation feature ensures that all agents can be kept busy by distributing tasks in each segment separately. As a result, even in a Distribution strategy that is populated by high-priority tasks assigned to small groups of agents, the strategy will not become so saturated that distribution of tasks to other agents is blocked. Segmentation settings are found in the **ToDistribute** view of the Distribution routing strategy. The Distribution strategy can make a call to the segmentation setting and add an `IWD_Segment` attribute to the interaction data.

Composer Configuration - Segmentation View


View Properties: @pl-byd-mo82

Configure View Options

Name
ToDistribute [iwd_bp_comp.
ToReprioritize [iwd_bp_comp

Configured Segments

Name	Value
IWD_ext_requestedAgentGroup	IWD_ext_requestedAgentGroup

Segment interval: 0 (seconds)

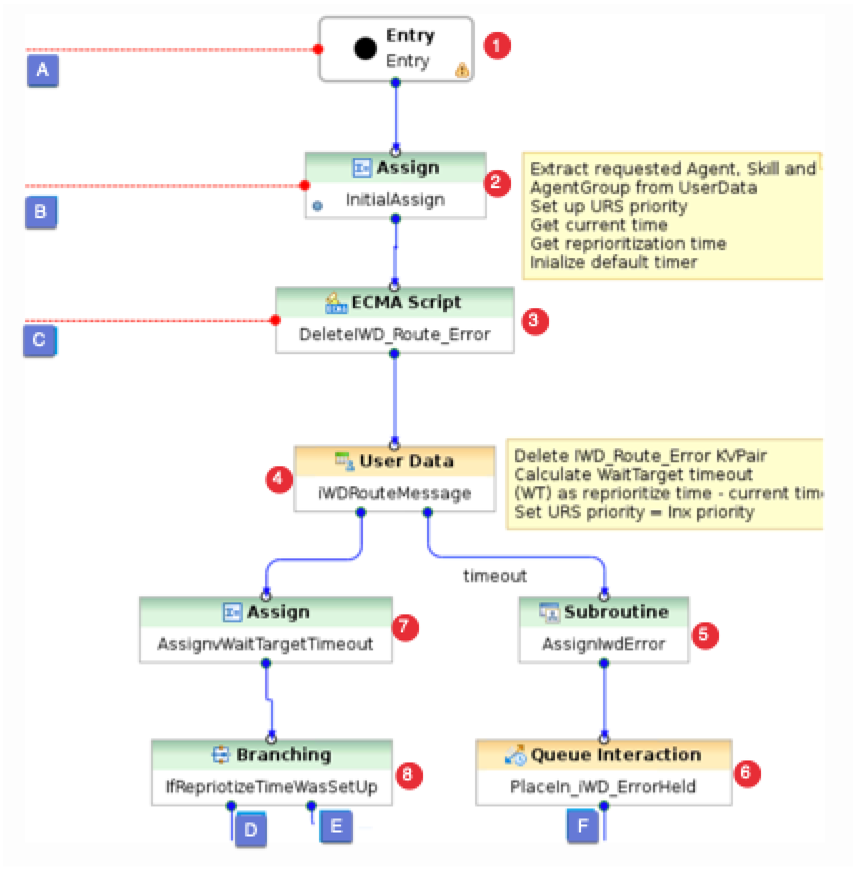
Segment limit: 200

Default Segment limit: 100

Flow Summary

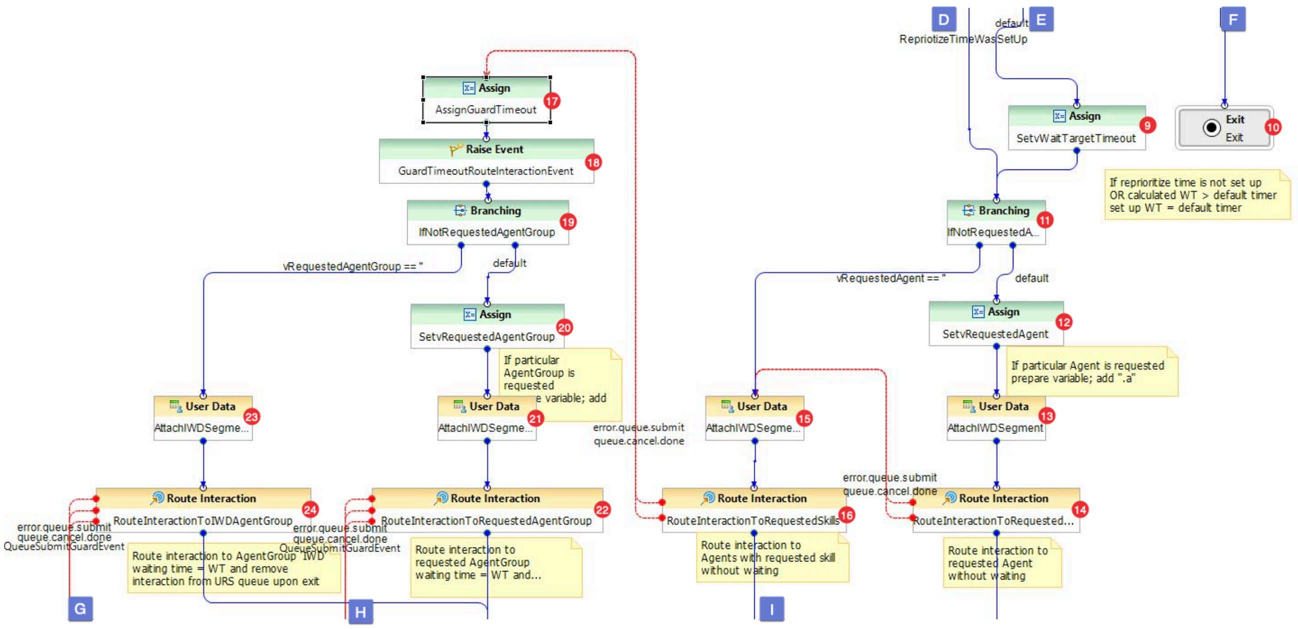
Part 1

Click to enlarge.



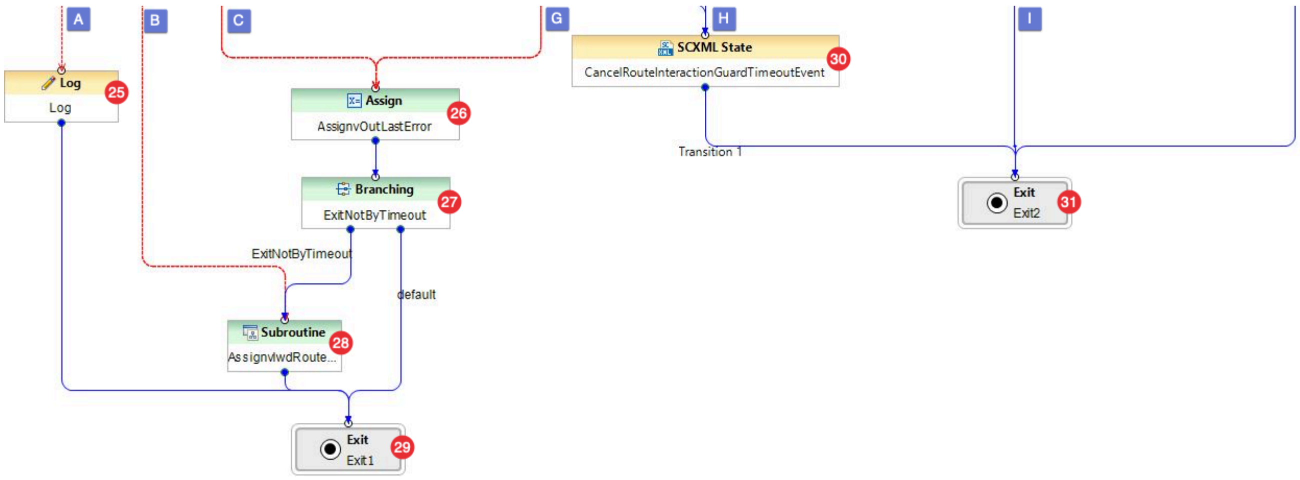
Part 2

Click to enlarge.


D
E
F
default
RepriotizeTimeWasSetUp
Assign
SetvWaitTargetTimeout
9
Exit
Exit
10
If reprioritize time is not set up OR calculated WT > default timer set up WT = default timer
Assign
AssignGuardTimeout
17
Raise Event
GuardTimeoutRouteInteractionEvent
18
Branching
IfNotRequestedAgentGroup
19
vRequestedAgentGroup == ''
default
Assign
SetvRequestedAgentGroup
20
If particular AgentGroup is requested prepare variable; add
Branching
IfNotRequestedA...
11
vRequestedAgent == ''
default
Assign
SetvRequestedAgent
12
If particular Agent is requested prepare variable; add ".a"
User Data
AttachIWDSegme...
23
User Data
AttachIWDSegme...
21
error.queue.submit
queue.cancel.done
User Data
AttachIWDSegme...
15
User Data
AttachIWDSegment
13
error.queue.submit
queue.cancel.done
Route Interaction
RouteInteractionToIWDAgentGroup
24
error.queue.submit
queue.cancel.done
QueueSubmitGuardEvent
Route interaction to AgentGroup 'IWD' waiting time = WT and remove interaction from URS queue upon exit
Route Interaction
RouteInteractionToRequestedAgentGroup
22
error.queue.submit
queue.cancel.done
QueueSubmitGuardEvent
Route interaction to requested AgentGroup waiting time = WT and...
Route Interaction
RouteInteractionToRequestedSkills
16
Route interaction to Agents with requested skill without waiting
Route Interaction
RouteInteractionToRequested...
14
Route interaction to requested Agent without waiting
G
H
I

Part 3

Click to enlarge.



Flow Detail

1. Entry to Distribution workflow.
2. Variables are initialized:
   - vRequestedAgentGroup—Read from task attribute `IWD_ext_requestedAgentGroup`
   - vRequestedAgentGroup—Read from task attribute `IWD_ext_requestedAgent`
   - vRequestedSkill—Read from task attribute `IWD_ext_requestedSkill`

- vCurrentTint—Current time in seconds
- vReprioritizeDint—Read from task attribute IWD_businessValue
- vDefaultTargetTimeout—Default target timeout set to 3600 seconds
- vInxPriority—Read from task attribute Priority

3. Delete IWD_Route_Error from attached data. Calculate WaitTarget timeout based on vReprioritizeDTInt and vCurrentDTInt. Sets URS priority.
4. Set information about clear IWD_Route_Error attribute.
5. Invoke AssignLastError subroutine with attributes:
   - vInLastErrorkey—IWD_Error
   - vInLastErrorString—Error description: 'Update IWD_Route_Error timeout'
6. The interaction is placed in the iwd_bp_comp.Main.iWD_ErrorHeld queue.
7. Calculate vWaitTargetTimeout.
8. Check if calculated vWaitTargetTimeout is in range (0, vDefaultTargetTimeout>.
9. Set vWaitTargetTimeout to vDefaultTargetTimeout.
10. Exit Distribution workflow.
11. Check if particular Agent is requested.
12. Assign vRequestedAgent + '.a' to vRequestedAgent variable.
13. Set vIWDSegment to '_requested_agent'.
14. Route interaction to requested vRequestedAgent without waiting.
15. Set vIWDSegment to '_requested_skill'.
16. Route interaction to requested vRequestedAgent with requested skill without waiting.
17. Set vRouteGuardTimeout to Route Interaction timeout (vWaitTargetTimeout) plus vRouteGuardDelay. vRouteGuardDelay is 15 seconds by default and can be configured using **User Variables** dialog for Distribution strategy.
18. Add event QueueSubmitGuardEvent to the event queue to be raised after vRouteGuardTimeout seconds.
19. Check if particular AgentGroup is requested.
20. Assign vRequestedAgentGroup + '.qa' to vRequestedAgentGroup variable.
21. Set vIWDSegment to '_requested_agent_group'.
22. Route interaction to requested vRequestedAgentGroup with vWaitTargetTimeout.
23. Set vIWDSegment to 'default'.
24. Route interaction to IWD Agent Group with vWaitTargetTimeout.
25. Log message in case if interaction was from some reasons deleted.
26. Assign last route interaction error to vLastError.
27. Check if route interaction finished with an error.
28. Invoke AssignLastError subroutine with attributes:

- `vInLastErrorkey`—`IWD_Route_Error`
- `vInLastErrorString`—Error description that occurred in route interaction

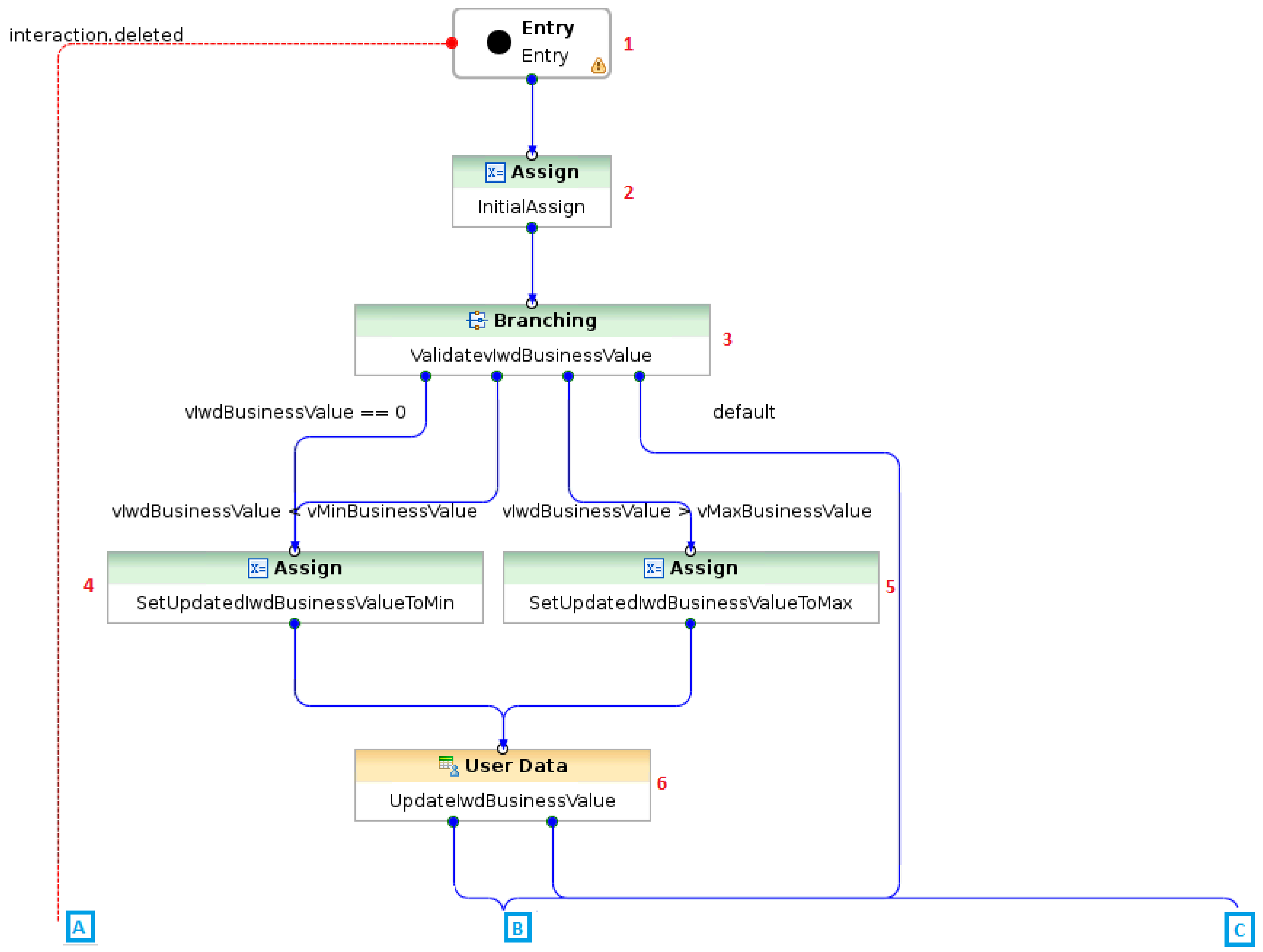
29. Exit Distribution workflow.
30. Cancel event `QueueSubmitGuardEvent` if it was not raised.
31. Exit Distribution workflow.

CheckBusinessValueandPriority Subroutine

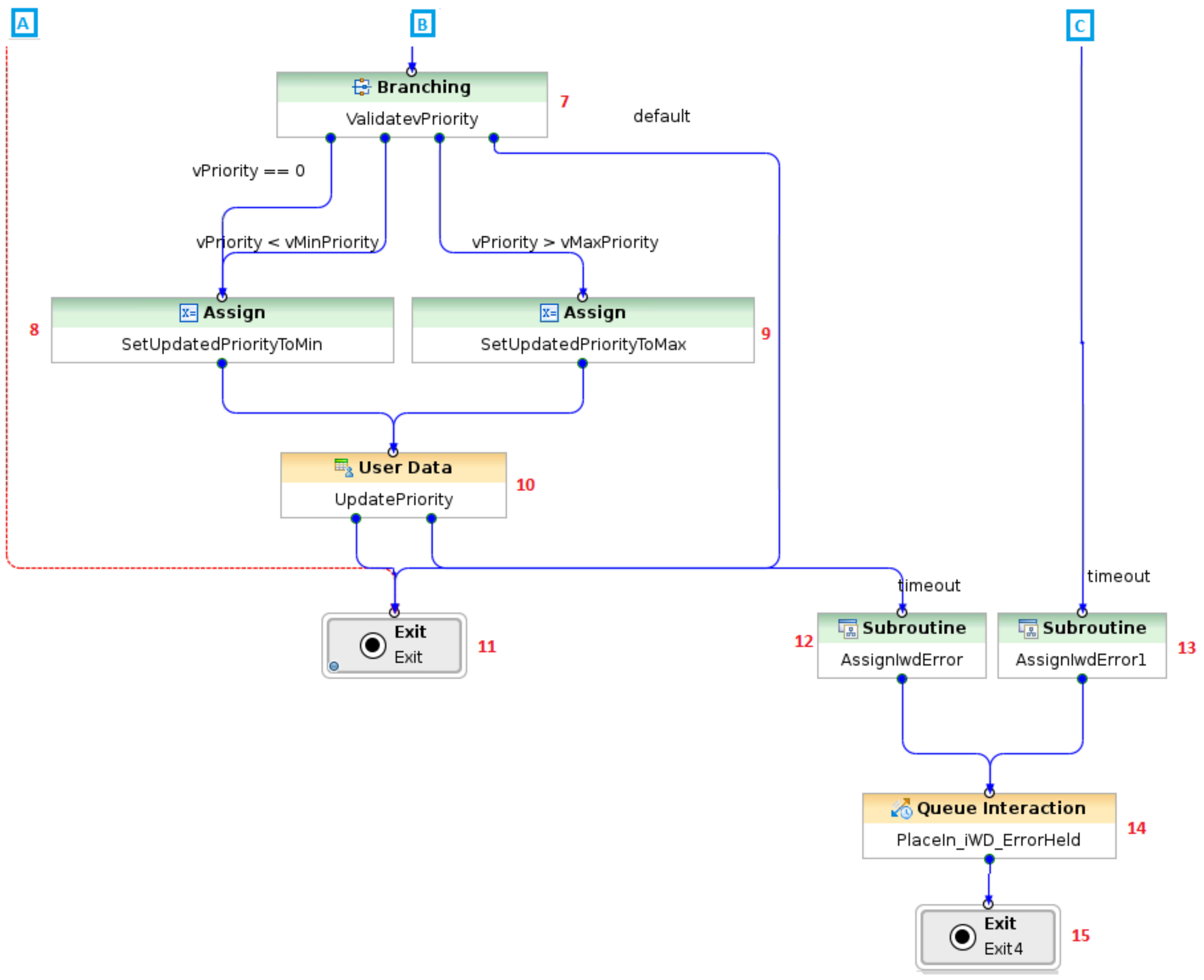
The purpose of this workflow is to verify if `Priority` and `IWD_businessValue` have correct values.

Flow Summary

Part 1


interaction.deleted
Entry
Entry
1
Assign
InitialAssign
2
Branching
ValidatevIwdBusinessValue
3
vIwdBusinessValue == 0
default
vIwdBusinessValue < vMinBusinessValue
vIwdBusinessValue > vMaxBusinessValue
4
Assign
SetUpdatedIwdBusinessValueToMin
Assign
SetUpdatedIwdBusinessValueToMax
5
User Data
UpdateIwdBusinessValue
6
A
B
C

Part 1



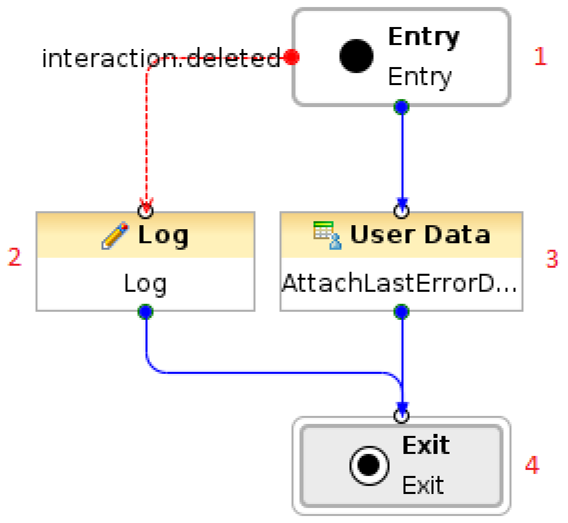
Flow Detail

1. Entry to CheckBusinessValueAndPriority workflow.
2. Variables are initialized:
   - `vIwdBusinessValue`—Read from task attribute `IWD_businessValue`
   - `vIwdPriority`—Read from task attribute `Priority`
3. Validate if `vIwdBusinessValue` is valid.
4. Set `vIwdBusinessValue` to `vMinBusinessValue`.
5. Set `vIwdBusinessValue` to `vMaxBusinessValue`.
6. Update `IWD_businessValue` to `vIwdBusinessValue`.

7. Validate if `vIwdPriority` is valid.
8. Set `vIwdPriority` to `vMinPriority`.
9. Set `vIwdPriority` to `vMaxPriority`.
10. Update Priority to `vIwdPriority`.
11. Exit `CheckBusinessValueAndPriority` workflow.
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString` - Error description: '`Update Priority timeout`'
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString`— Error description: '`Update iWD_businessValue timeout`'
14. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
15. Exit `CheckBusinessValueAndPriority` workflow.

AssignLastError Subroutine

Flow Summary



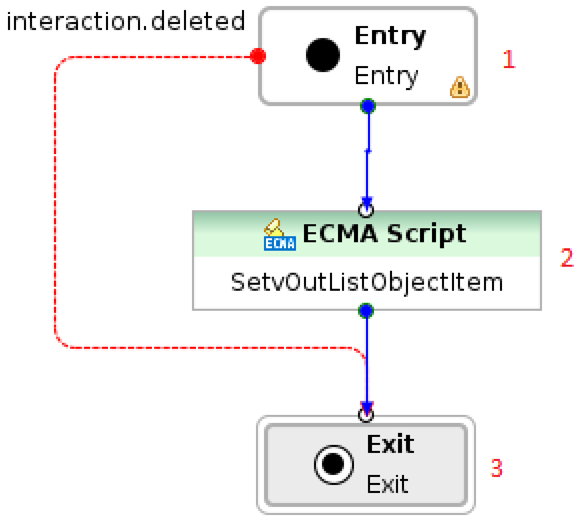
Flow Detail

1. Entry to AssignLastError workflow.
2. The last error is attached to user data as a key-value pair with the key `vInLastErrorkey` and value `vInLastErrorString`.

   `vInLastErrorkey` and `vInLastErrorString` are workflow attributes that need to be set before calling this workflow.
3. Log message if interaction was from some reasons deleted.
4. Exit AssignLastError workflow.

FindListObjectItem Subroutine

Flow Summary



Flow Detail

1. Entry to FindListObjectItem workflow.
2. Search `vKeyToFindInListObject` in `vInListName`.
   - `vInItemName`—Section in `vInListName`
   - `vInListName`—List object where `vKeyToFindInListObject` should be searched

- `vKeyToFindInListObject`—Option in `vInItemName` that should be found

3. When `vKeyToFindInListObject` is found in `vInListName`, then the value assigned to this option will be assigned to `vOutListObjectItem`.
4. Exit FindListObjectItem workflow.

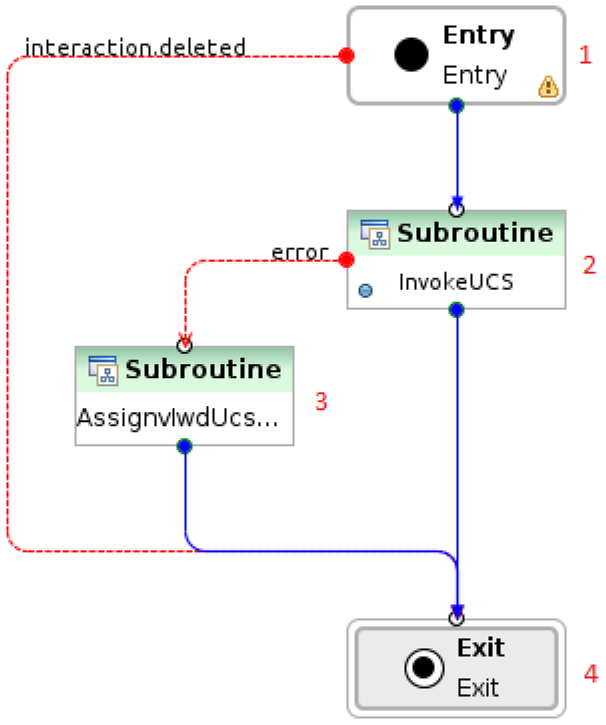
MarkInteractionAsDone Strategy

The purpose of this strategy is to update the Universal Contact Server (UCS) database to mark the interaction as done. This equates to setting the value in the Status column of the Interactions table to 3. UCS clients, such as Interaction Workspace, will then display the status of this interaction as done when the user looks at interactions they have previously processed.

Interactions have to satisfy the following conditions:

- The value of the attached data key `IWD_isContactServer` is `1`
- The value of the attached data key `IWD_isDone` is either null or `0` (zero)

Flow Summary



Flow Detail

1. Entry to MarkInteractionAsDone workflow.
2. The InvokeUCS subroutine is invoked to complete interaction in the UCS database.
3. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine
4. Exit MarkInteractionAsDone workflow.

Removal Strategy

The purpose of this strategy is to delete expired interactions from the Interaction Server database.

A key-value pair in user data with the key `IWD_expirationDateTime` contains information about when an interaction has to be deleted.

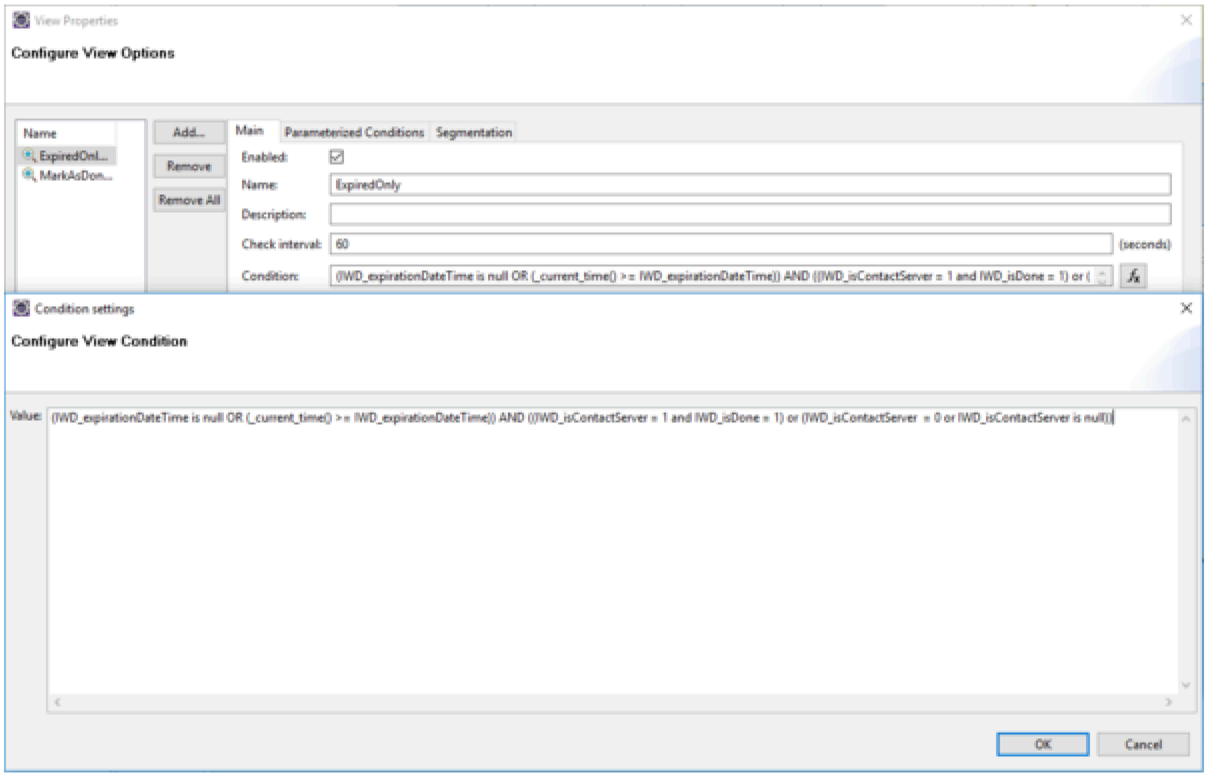
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Rejected`

Interactions have to satisfy the following conditions:

- Interactions must either have the property `IWD_expirationDateTime` not set, or this property must have a time stamp which is in the past.
- If UCS is available, interactions must be marked as done in UCS.
- Interactions are taken in the order they were submitted.

Composer Configuration



ExpiredOnl...
MarkAsDon...

Enabled: ✓

Name: ExpiredOnly

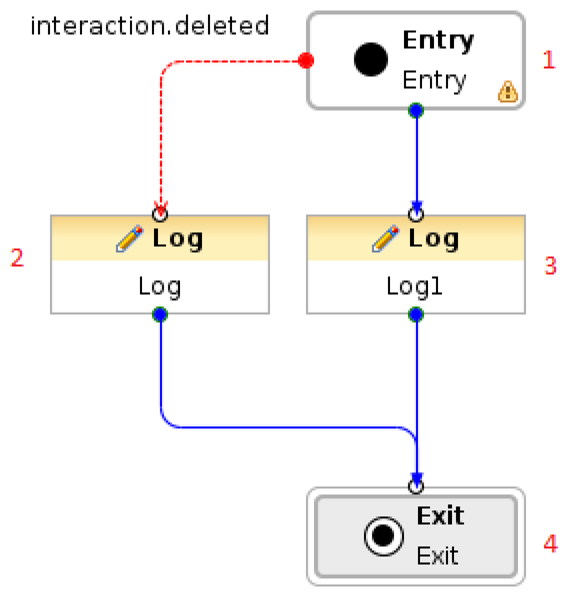
Description:

Check interval: 60 (seconds)

Condition: (IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (

Value: (IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (IWD_isContactServer = 0 or IWD_isContactServer is null))

Flow Summary



Flow Detail

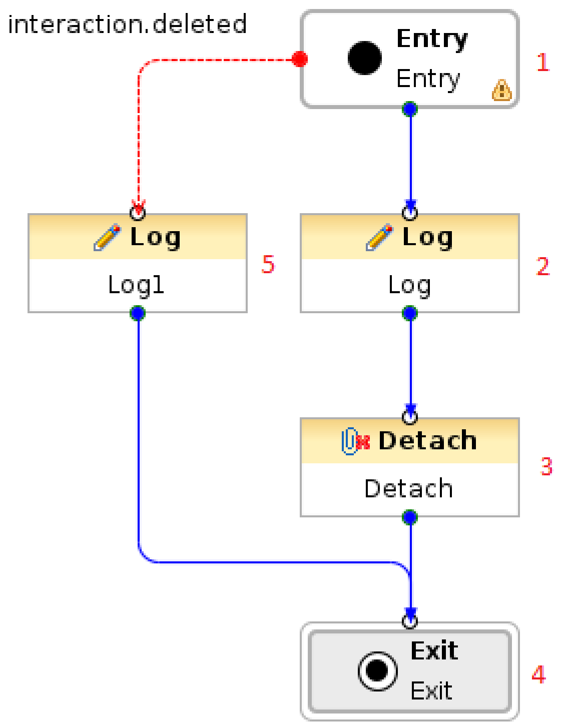
1. Entry to Removal workflow.
2. Log message in case if interaction was from some reasons deleted.
3. Log message: `Task will be terminated on exit`
4. Exit Removal workflow.

Finish Strategy

This workflow detaches interactions from the session. This workflow processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Errorheld`

Flow Summary



Flow Detail

1. Entry to Finish workflow.
2. Log message :'Task processing completed'.
3. Detach interaction. After this operation, the interaction will not be processed any more.
4. Log message in case if interaction was for some reason deleted.
5. Exit Finish workflow.

IWDBP Strategies & Subroutines from 9.0.009

Global variables

Timeout for userdata changes

During a task life cycle, its interaction properties are changed asynchronously. Without sufficient timeout to receive confirmation event from Interaction Server, ORS might continue workflow execution with unfinished property updates. This could lead to unexpected behavior—for example, tasks could go to the **ErrorHeld** queue sporadically without visible reasons.

To address such issues, there is a configurable timeout within UserData blocks and other places where interaction properties are changed (for example, the **AfterEspCallActivities** block in the InvokeGRE strategy). If a timeout event occurs, an interaction goes to the **ErrorHeld** queue with a corresponding error message assigned to its Userdata. This timeout could be set globally using the `vWaitTimeoutSec` variable on the project level. The default value is 10 seconds. The timeout should be calculated individually and specifically, depending on the delay in the Interaction Server response. To set its value, do the following:

1. Click on the **Entry** block of any strategy.
2. Select the **Properties** tab at the bottom of the Composer window.
3. Click on the dots next to **Global Settings -> Variables** to open the **Application Variables** window.
4. Expand the **Project Variables** item and set the `vWaitTimeoutSec` value.

Classification Strategy

The purpose of this strategy is to invoke corresponding classification rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

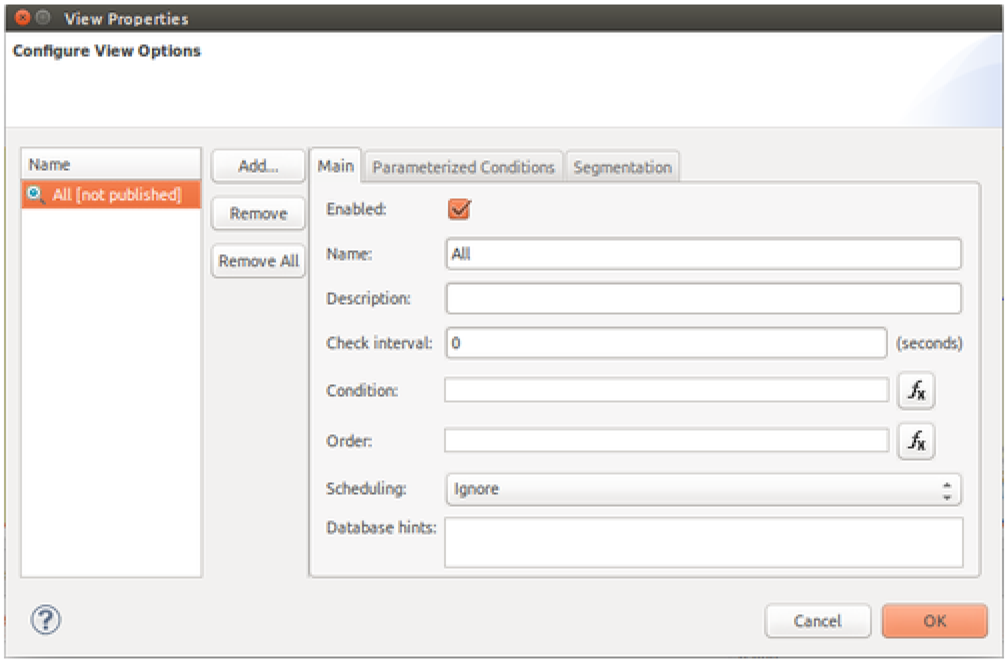
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_New`

Interactions have to satisfy the following conditions:

- There are no conditions here.
- Interactions are taken in order they were submitted.

Composer Configuration

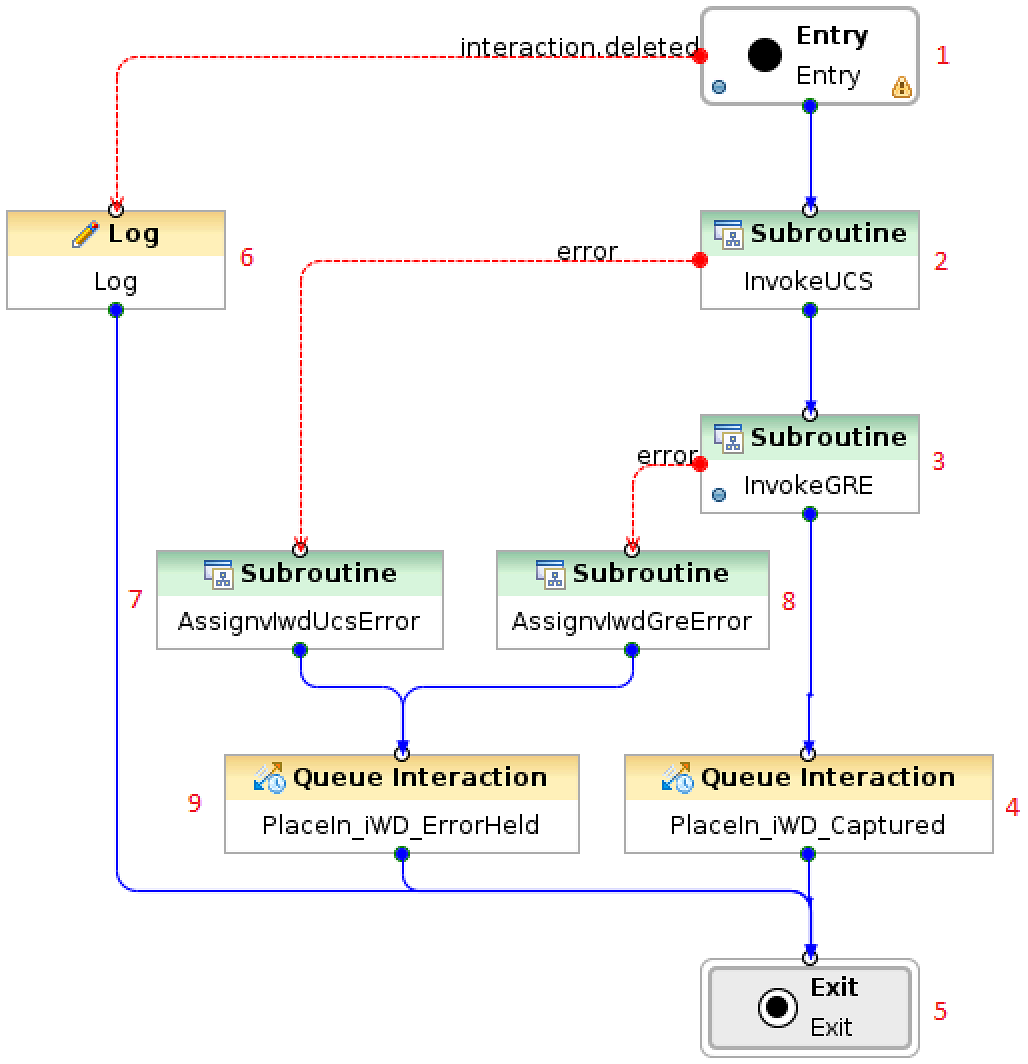


All [not published]

Enabled: ✓
Name: All
Description:
Check interval: 0 (seconds)
Condition:
Order:
Scheduling: Ignore
Database hints:

The Composer configuration for this strategy block shows that all interactions are distributed to the `iwd_bp_comp.Main.iWD_New` queue without conditions.

Flow Summary



Flow Detail

1. Entry to Classification workflow.
2. The InvokeUCS subroutine is invoked to create new interaction in the UCS database.
3. The InvokeGRE subroutine is invoked.
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
5. Exit Classification workflow.
6. Log message in case if interaction was from some reasons deleted.

7. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_UCS_Error`.
   - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine.
8. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_GRE_Error`
   - `vInLastErrorString`—Error description that occured in InvokeGRE subroutine.
9. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

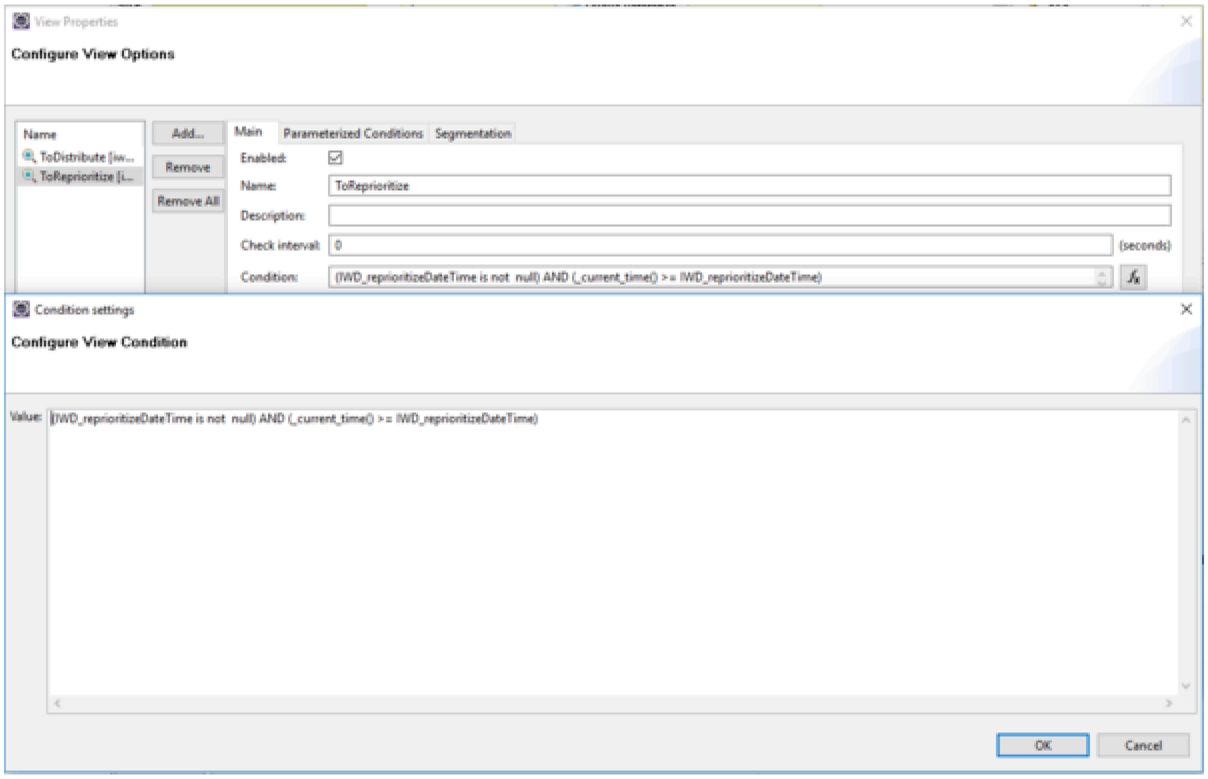
Prioritization Strategy

The purpose of this strategy is to invoke the corresponding prioritization rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Captured`—Interactions have to satisfy the following conditions:
  - Active interactions only (interactions which do not have the property `IWD_activationDateTime` set, or this property has a time stamp which is in the past.
  - Interactions are taken in the order they were submitted.
- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
  - Interactions that are subject for immediate reprioritization (interactions that have the property `IWD_reprioritizeDateTime` set to a time stamp which is in the past).
  - Interactions are taken in order of `IWD_reprioritizationDateTime` (oldest first).

Composer Configuration

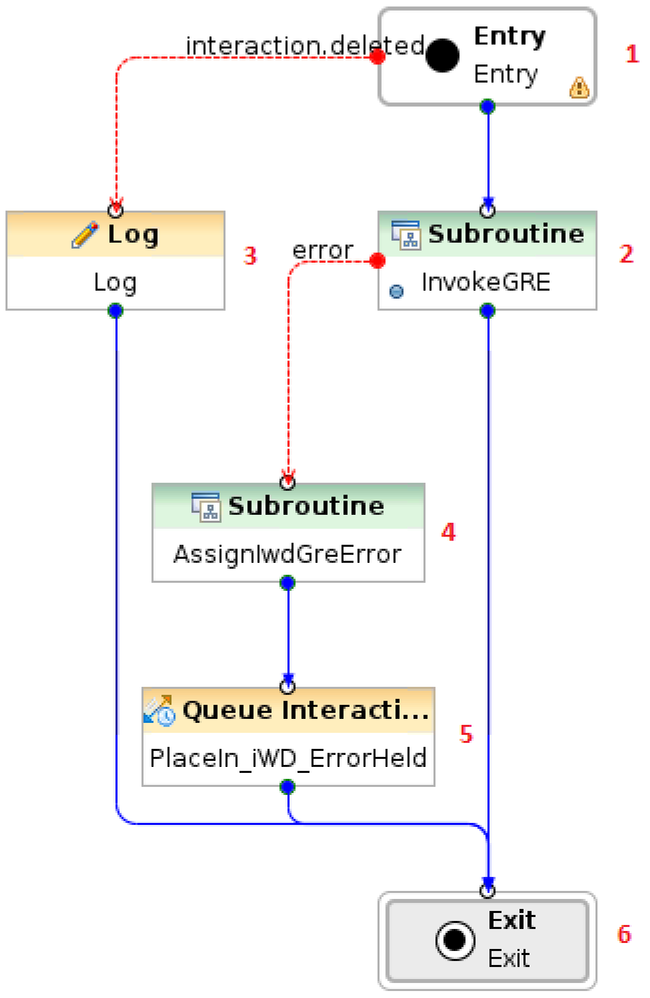


Name
ToDistribute [iw...
ToReprioritize [i...

Enabled: ✓
Name: ToReprioritize
Description:
Check interval: 0 (seconds)
Condition: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Value: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Flow Summary



Flow Detail

1. Entry to Prioritization workflow.
2. The `InvokeGRE` subroutine is invoked.
3. Log message in case if interaction was from some reasons deleted.
4. Invoke `AssignLastError` subroutine with attributes:
   - `vInLastErrorkey`—`IWD_GRE_Error`
   - `vInLastErrorString`—Error description that occurred in `InvokeGRE` subroutine.
5. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

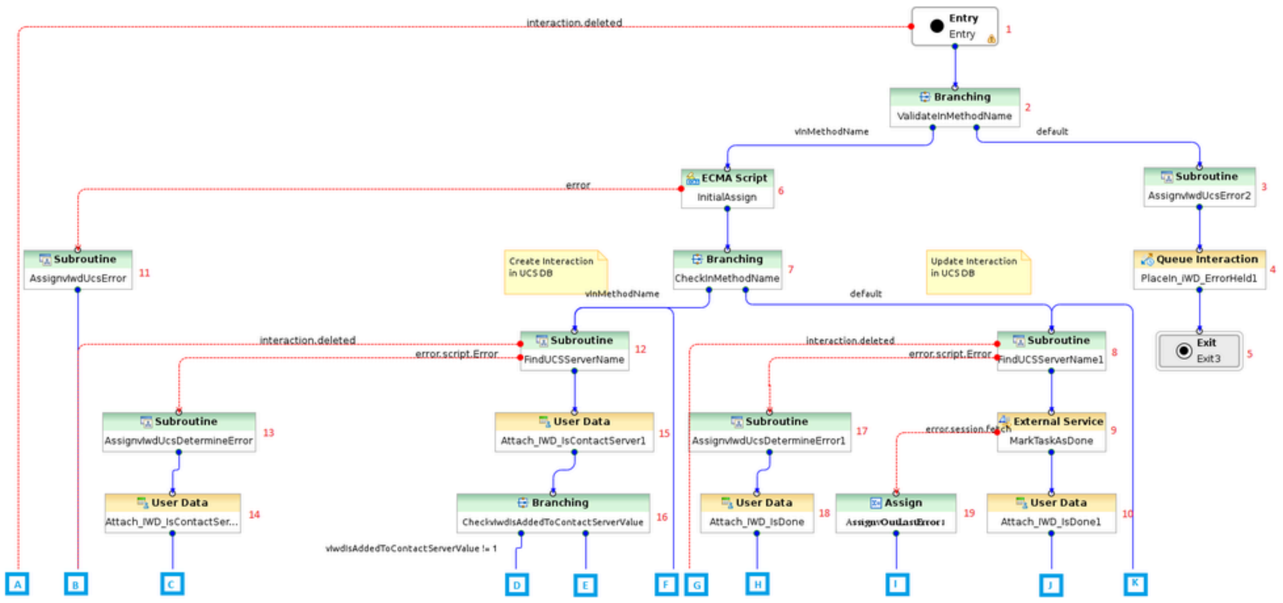
6. Exit Prioritization workflow.

Invoke UCS Strategy

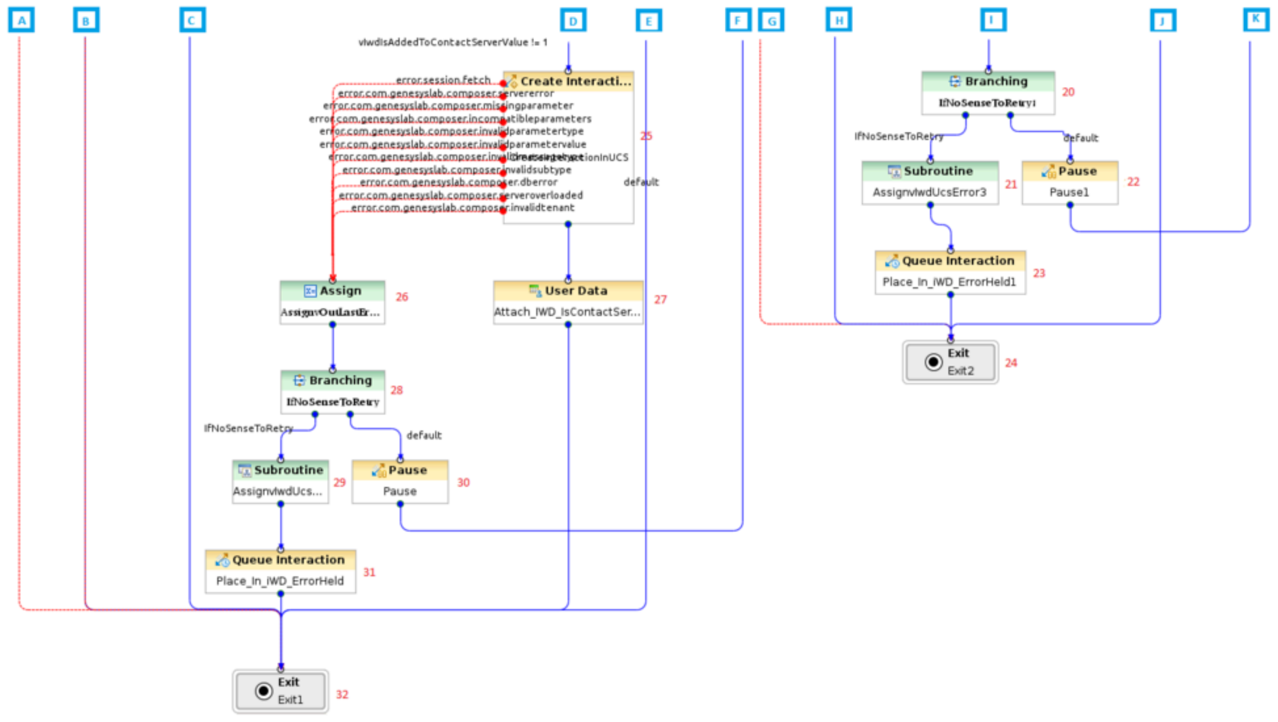
The purpose of this workflow is to create an interaction in the UCS database if UCS is configured.

Flow Summary

Part 1



Part 2


A
B
C
D
E
F
G
H
I
J
K
vIwdIsAddedToContactServerValue != 1
Create Interacti...
error.session.fetch
error.com.genesyslab.composer.servererror
error.com.genesyslab.composer.missingparameter
error.com.genesyslab.composer.incompatibleparameters
error.com.genesyslab.composer.invalidparametertype
error.com.genesyslab.composer.invalidparametervalue
error.com.genesyslab.composer.invalidresponsetype
CreateInteractionInUCS
error.com.genesyslab.composer.invalidsubtype
error.com.genesyslab.composer.dberror
error.com.genesyslab.composer.serveroverloaded
error.com.genesyslab.composer.invalidtenant
default
25
Assign
AssignvOutLastEr...
26
User Data
Attach_IWD_IsContactSer...
27
Branching
IfNoSenseToRetry
28
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcs...
29
Pause
Pause
30
Queue Interaction
Place_In_iWD_ErrorHeld
31
Exit
Exit1
32
Branching
IfNoSenseToRetry1
20
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcsError3
21
Pause
Pause1
22
Queue Interaction
Place_In_iWD_ErrorHeld1
23
Exit
Exit2
24

Flow Detail

1. Entry InvokeUCS strategy.
2. Check if `in_method_name = 'Create' | in_method_name = 'OMInteractions'`.
3. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_UCS_Error`
   - `vInLastErrorString`—Error informs that: `vInMethodName + ' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeUCS workflow.
6. Variables are initialized:
   - `vExternalId`—Read from task attribute `ExternalId`
   - `vMediaType`—Read from task attribute
   - `vSubmittedBy`—Read from task attribute `attr_itx_submitted_by`
   - `vType`—Read from task attribute `'InteractionType'`
   - `vSubType`—Read from task attribute `'InteractionSubtype'`
   - `vIwdIsAddedToContactServerValue`—Read from task attribute `'IWD_isAddedToContactServer'`

7. Check if `in_method_name = 'Create'`.
8. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
   - `vInItemName`—`ContactServerList`
   - `vInListName`—`Iwd_Esp_List`
9. The strategy calls a method on the Universal Contact Server to set the status of the interaction to 3, indicating that the interaction is done.
10. The value of the user data key `IWD_isDone` is set to 1.
11. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`'
    - `vInLastErrorString`—Error description that occurred when variables were initialized
12. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
    - `vInItemName`—`ContactServerList`
    - `vInListName`—`Iwd_Esp_List`
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Determination_Error`'
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
14. The value of the user data key `IWD_isContactServer` is set to `0`.
15. The value of the user data key `IWD_isContactServer` is set to 1.
16. Check if `IWD_isContactServer` is set to 1.
17. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
18. The value of the user data key `IWD_isDone` is set to `0`.
19. An error is extracted from user data and assigned in `vLastError` variable.
20. If it makes sense to retry updating the interaction record in UCS.
21. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry update interaction in UCS.
22. A delay is introduced into the processing. Flow returns to step 8.
23. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
24. Exit InvokeUCS workflow.
25. A new interaction is created in the UCS database, for this iWD task.
26. An error is extracted from user data and assigned in `vLastError` variable.

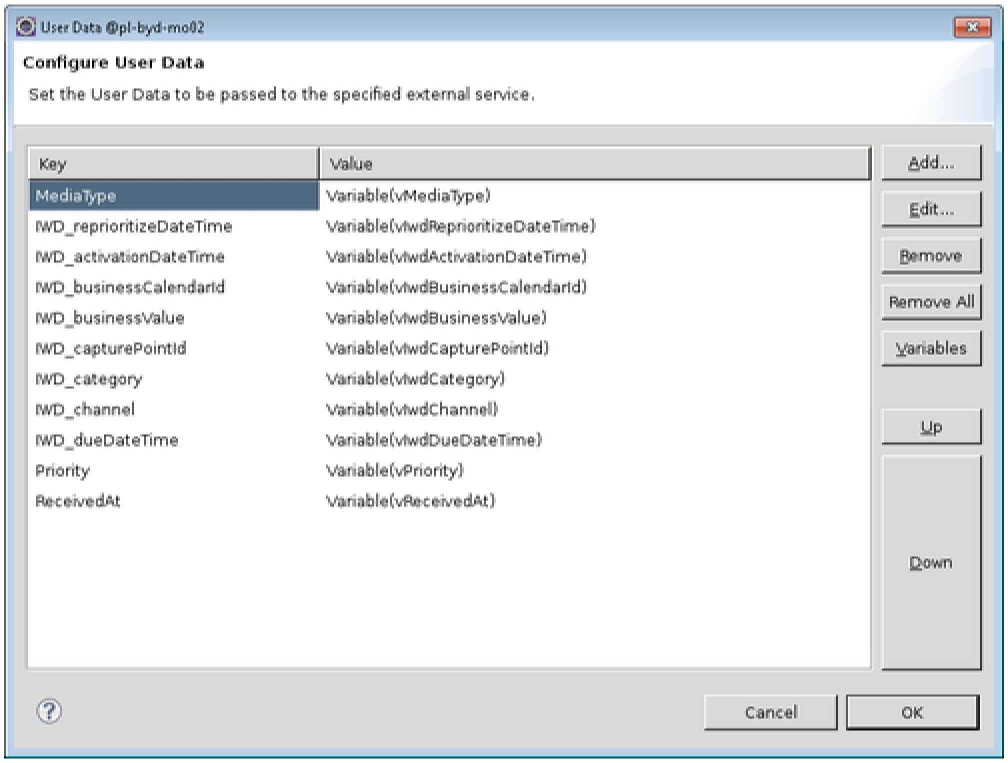
27. The user data key `IWD_isAddedToContactServer` is updated to 1 to indicate that the task was successfully added to the interaction history in UCS. The result returned from the ESP call to UCS (from See A new interaction is created in the UCS database, for this iWD task. If that function is successful is written to the variable `IWD_UCS_Result`.
28. If it makes sense to retry creating the interaction record in UCS.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry create interaction in UCS
30. A delay is introduced into the processing. Flow returns to step 12.
31. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
32. Exit InvokeUCS workflow.

InvokeGRE Strategy

Important

For Composer/ORS versions prior 8.1.400.48—If custom task attributes will be used in the Standard Rules Template, you must add them in the External Service block called `InvokeGRE` in the InvokeGRE workflow. All user-defined attributes need to be added in the User Data attribute, otherwise they will not be attached to the task and so will not be sent in the ESP request to the external ESP service.

Composer configuration



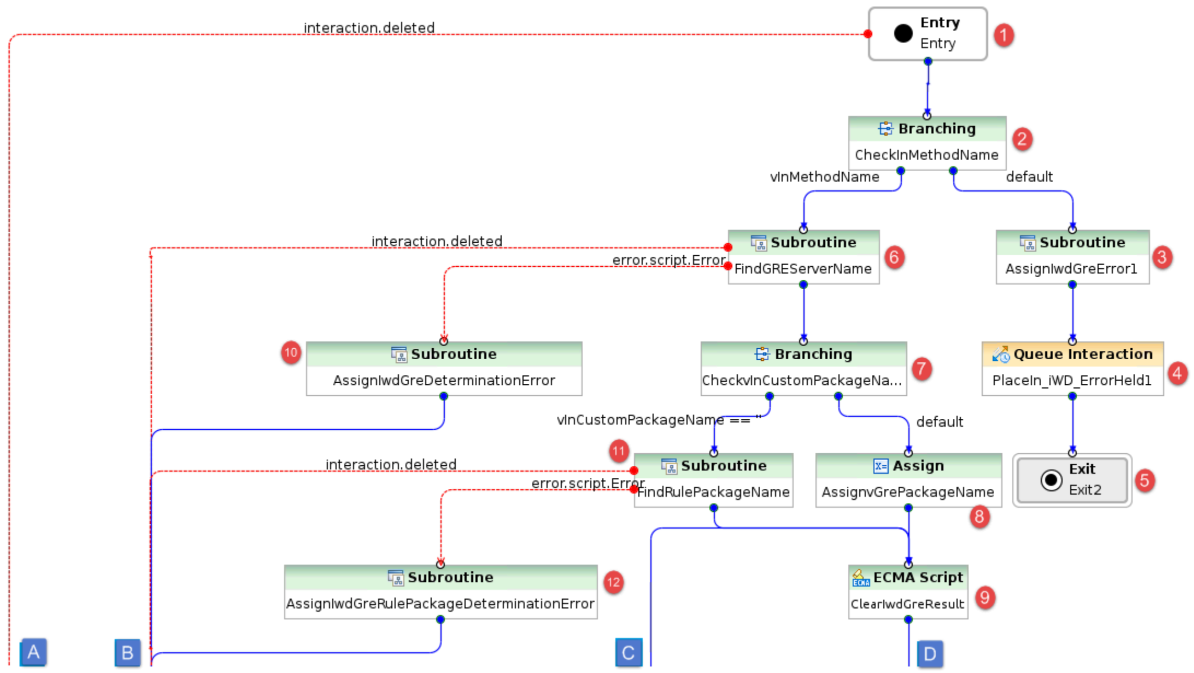
Configure User Data

Set the User Data to be passed to the specified external service.

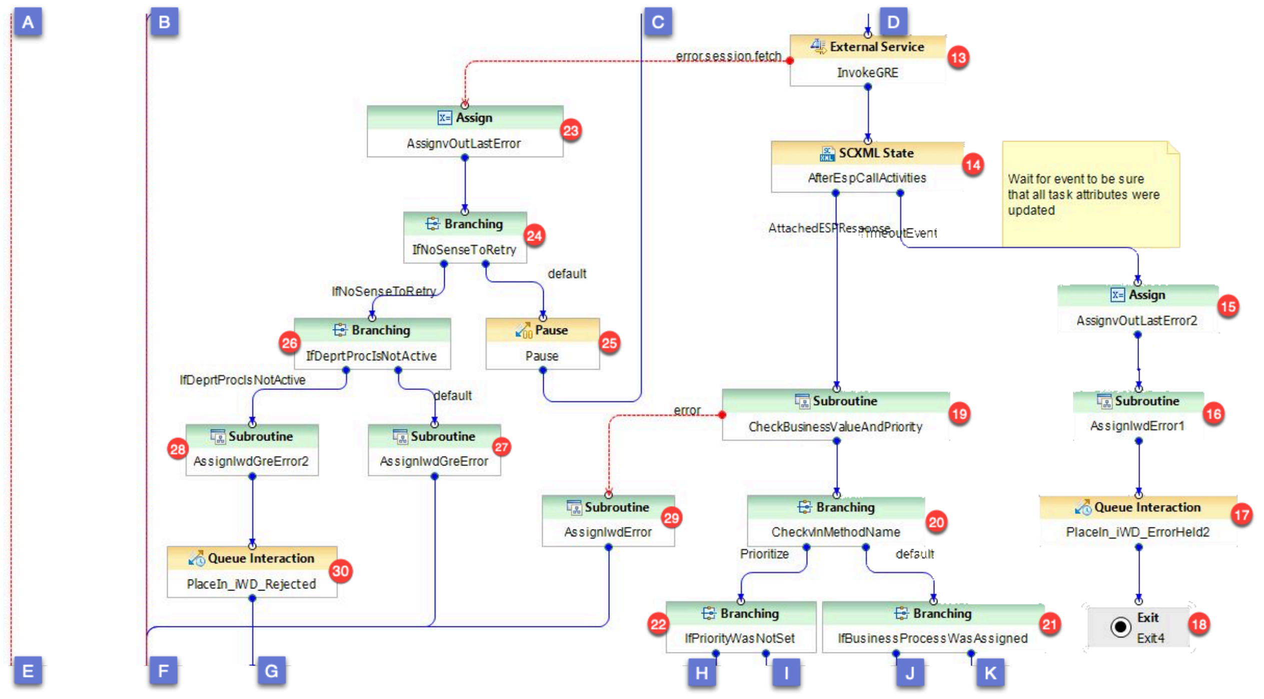
Key	Value
MediaType	Variable(vMediaType)
IWD_reprioritizeDateTime	Variable(vIwdReprioritizeDateTime)
IWD_activationDateTime	Variable(vIwdActivationDateTime)
IWD_businessCalendarId	Variable(vIwdBusinessCalendarId)
IWD_businessValue	Variable(vIwdBusinessValue)
IWD_capturePointId	Variable(vIwdCapturePointId)
IWD_category	Variable(vIwdCategory)
IWD_channel	Variable(vIwdChannel)
IWD_dueDateTime	Variable(vIwdDueDateTime)
Priority	Variable(vPriority)
ReceivedAt	Variable(vReceivedAt)

Flow Summary

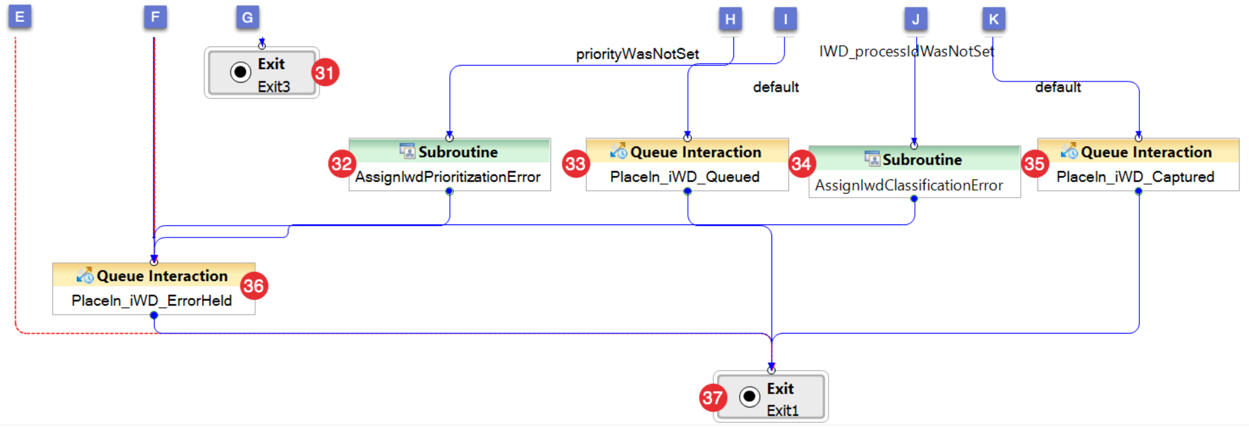
Part 1



Part 2



Part 3



Flow Detail

1. Entry to `InvokeGRE` strategy.
2. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.

3. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `vInMethodName` + `' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeGRE workflow.
6. The `FindListObjectItem` subroutine is invoked to determine the name of the Genesys Rules Engine Application. The subroutine uses the List Object list `GREServerList`:
    - `vInItemName`—`GREServerList`
    - `vInListName`—`Iwd_Esp_List`
7. Check if `vInCustomPackageName` was published to this subroutine. If it is set then `vInCustomPackageName` will be run. Otherwise package name needs to be found in `Iwd_Package_List`.
8. Assign `vInCustomPackageName` to `vGrePackageName`.
9. Delete `IWD_GRE_Result`, `IWD_Error`, `RulePhase` before Invoke GRE.
10. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
11. The `FindListObjectItem` subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules:
    - `vInItemName`—`RulePackageList`
    - `vInListName`—`Iwd_Package_List`
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Rule_Package_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
13. An ESP request is sent to the Genesys Rules Engine to evaluate the classification rules.

> **Important**
> All user data that needs to be added to ESP request must be added in User Data attributes.

14. Parse ESP result and attach to the interaction all attributes modified by the GRE.
15. Assign the string `AfterEspCallActivities timeout` to the `vLastError` variable.
16. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `Attach GreResult timeout`
17. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
18. Exit InvokeGRE workflow.

19. CheckBusinessValueAndPriority subroutine is called to verify if `IWD_businessValue` and `Priority` have correct values.
20. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
21. Check if `IWD_processId` was set by any rules or when task was created.
22. Check is made to see if this is the first time that prioritization rules are being evaluated for the interaction, and the priority was not set up by any rules.
23. Get last error that was occured in GRE call and assign it to `vLastError` variable.
24. A check is done to see if the error code is related to the ESP server communication.
25. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to step 11 to retry the connection to the ESP server.
26. The last Interaction Server-related error is extracted from a variable.
27. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
28. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable
30. The interaction is placed in the `iwd_bp_comp.Main.iWD_Rejected` queue.
31. Exit InvokeGRE workflow.
32. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Prioritization_Error`
    - `vInLastErrorString`—Error description: `Priority is not set up by rules`.
33. The interaction is placed in the `iwd_bp_comp.Main.iWD_Queued` queue.
34. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`vIwdClassificationError`
    - `vInLastErrorString`—Error description: `Process was not set by rules`.
35. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
36. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
37. Exit InvokeGRE workflow.

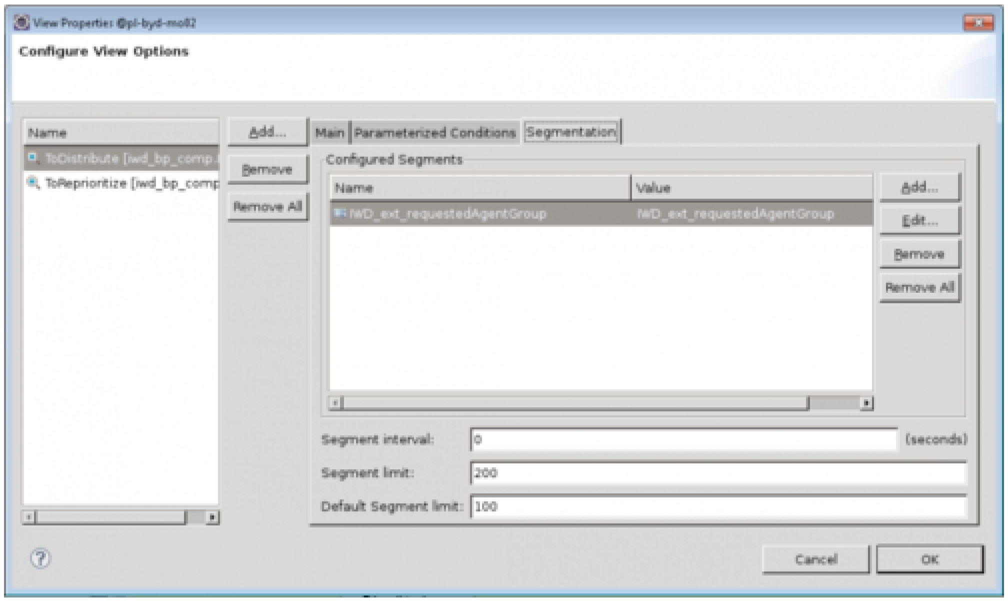
Distribution Strategy

This strategy routes interactions to a requested Agent, requested Agent Group, requested Skill, or to the default iWD Agent Group. This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are not subject for immediate reprioritization (interactions that do not have the property `IWD_reprioritizeDateTime` set, or that have this property set to a time stamp that is in the future).
    - Interactions are taken in order of priority (highest priority first)

A Segmentation feature ensures that all agents can be kept busy by distributing tasks in each segment separately. As a result, even in a Distribution strategy that is populated by high-priority tasks assigned to small groups of agents, the strategy will not become so saturated that distribution of tasks to other agents is blocked. Segmentation settings are found in the **ToDistribute** view of the Distribution routing strategy. The Distribution strategy can make a call to the segmentation setting and add an `IWD_Segment` attribute to the interaction data.

Composer Configuration - Segmentation View



Name
ToDistribute [iwd_bp_comp.
ToReprioritize [iwd_bp_comp

Configured Segments

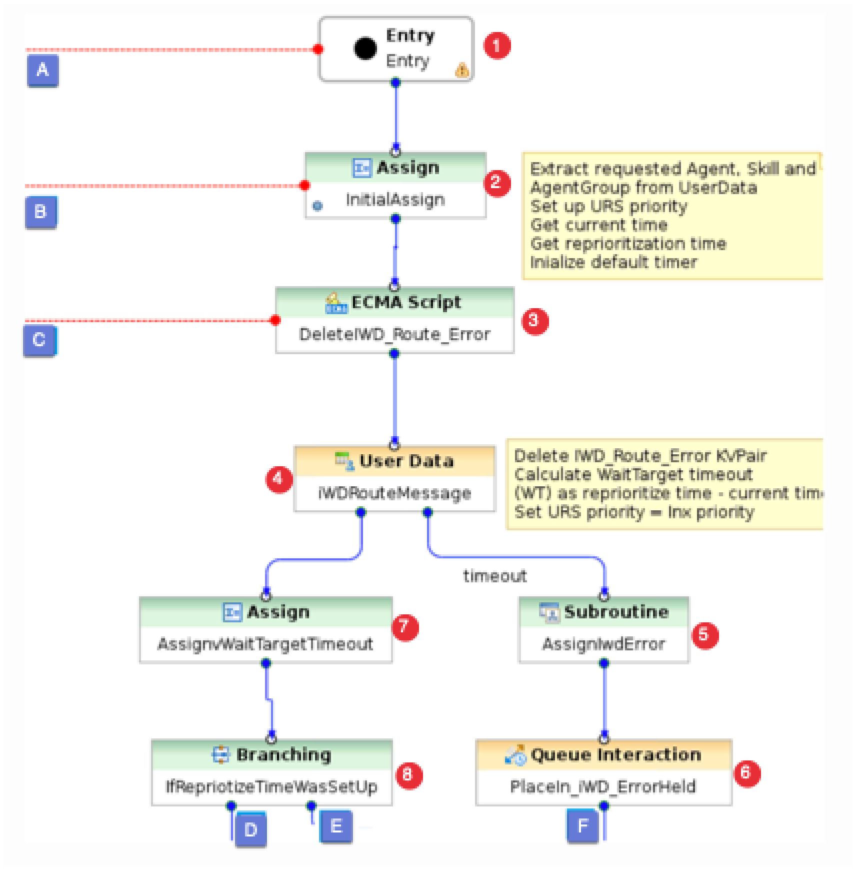
Name	Value
IWD_ext_requestedAgentGroup	IWD_ext_requestedAgentGroup

Segment interval: 0 (seconds)
Segment limit: 200
Default Segment limit: 100

Flow Summary

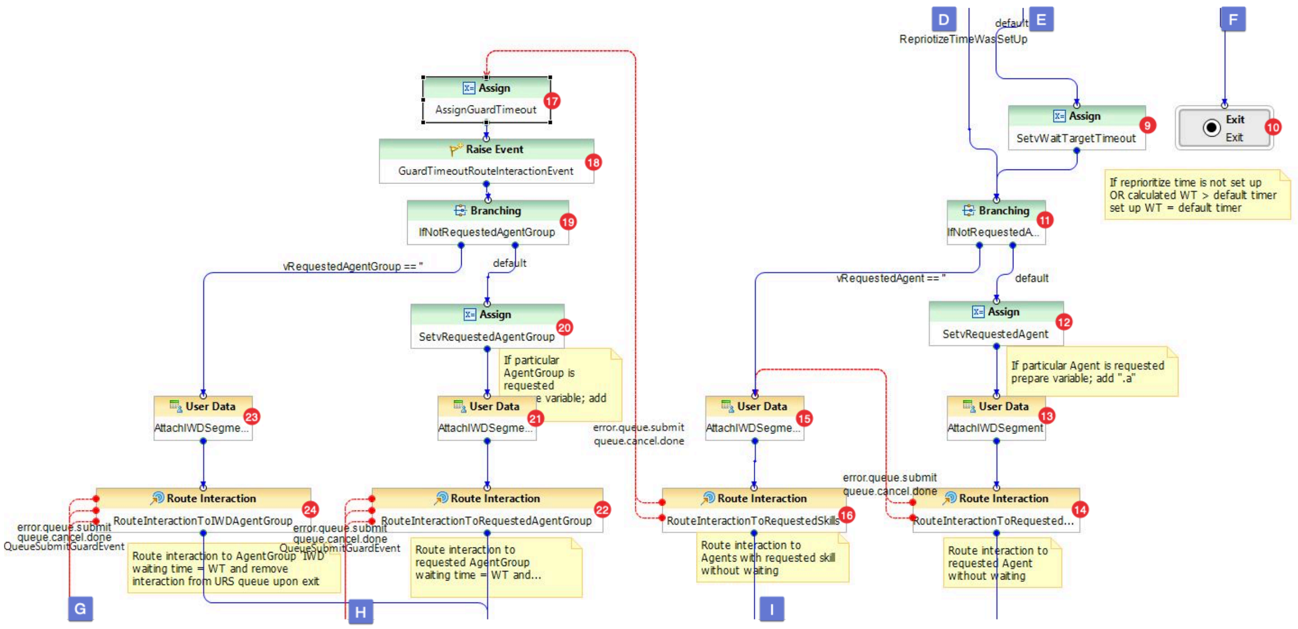
Part 1

Click to enlarge.



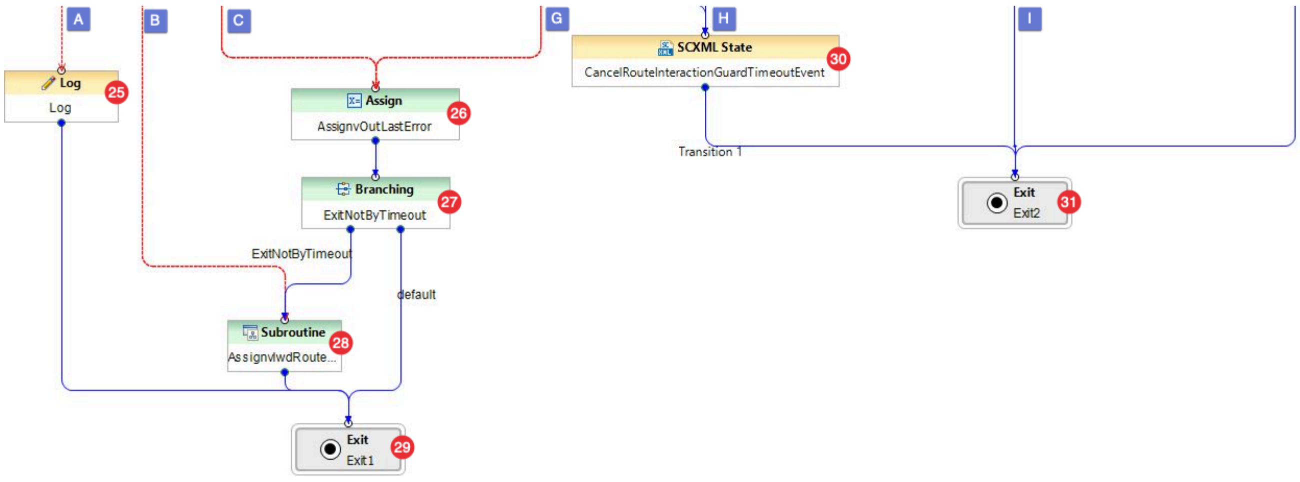
Part 2

Click to enlarge.


D
E
F
RepriotizeTimeWasSetUp
default
Assign
AssignGuardTimeout
17
Raise Event
GuardTimeoutRouteInteractionEvent
18
Branching
IfNotRequestedAgentGroup
19
vRequestedAgentGroup == ''
default
Assign
SetvRequestedAgentGroup
20
If particular AgentGroup is requested prepare variable; add
User Data
AttachIWDSegme...
23
User Data
AttachIWDSegme...
21
Route Interaction
RouteInteractionToIWDAgentGroup
24
Route Interaction
RouteInteractionToRequestedAgentGroup
22
error.queue.submit
queue.cancel.done
QueueSubmitGuardEvent
Route interaction to AgentGroup 'IWD' waiting time = WT and remove interaction from URS queue upon exit
Route interaction to requested AgentGroup waiting time = WT and...
Assign
SetvWaitTargetTimeout
9
Exit
Exit
10
If reprioritize time is not set up OR calculated WT > default timer set up WT = default timer
Branching
IfNotRequestedA...
11
vRequestedAgent == ''
default
Assign
SetvRequestedAgent
12
If particular Agent is requested prepare variable; add ".a"
User Data
AttachIWDSegme...
15
User Data
AttachIWDSegment
13
Route Interaction
RouteInteractionToRequestedSkills
16
Route Interaction
RouteInteractionToRequested...
14
Route interaction to Agents with requested skill without waiting
Route interaction to requested Agent without waiting
G
H
I

Part 3

Click to enlarge.



Flow Detail

1. Entry to Distribution workflow.
2. Variables are initialized:
   - vRequestedAgentGroup—Read from task attribute `IWD_ext_requestedAgentGroup`
   - vRequestedAgentGroup—Read from task attribute `IWD_ext_requestedAgent`
   - vRequestedSkill—Read from task attribute `IWD_ext_requestedSkill`

- vCurrentTint—Current time in seconds
- vReprioritizeDint—Read from task attribute IWD_businessValue
- vDefaultTargetTimeout—Default target timeout set to 3600 seconds
- vInxPriority—Read from task attribute Priority

3. Delete IWD_Route_Error from attached data. Calculate WaitTarget timeout based on vReprioritizeDTInt and vCurrentDTInt. Sets URS priority.
4. Set information about clear IWD_Route_Error attribute.
5. Invoke AssignLastError subroutine with attributes:
   - vInLastErrorkey—IWD_Error
   - vInLastErrorString—Error description: 'Update IWD_Route_Error timeout'
6. The interaction is placed in the iwd_bp_comp.Main.iWD_ErrorHeld queue.
7. Calculate vWaitTargetTimeout.
8. Check if calculated vWaitTargetTimeout is in range (0, vDefaultTargetTimeout>.
9. Set vWaitTargetTimeout to vDefaultTargetTimeout.
10. Exit Distribution workflow.
11. Check if particular Agent is requested.
12. Assign vRequestedAgent + '.a' to vRequestedAgent variable.
13. Set vIWDSegment to '_requested_agent'.
14. Route interaction to requested vRequestedAgent without waiting.
15. Set vIWDSegment to '_requested_skill'.
16. Route interaction to requested vRequestedAgent with requested skill without waiting.
17. Set vRouteGuardTimeout to Route Interaction timeout (vWaitTargetTimeout) plus vRouteGuardDelay. vRouteGuardDelay is 15 seconds by default and can be configured using **User Variables** dialog for Distribution strategy.
18. Add event QueueSubmitGuardEvent to the event queue to be raised after vRouteGuardTimeout seconds.
19. Check if particular AgentGroup is requested.
20. Assign vRequestedAgentGroup + '.qa' to vRequestedAgentGroup variable.
21. Set vIWDSegment to '_requested_agent_group'.
22. Route interaction to requested vRequestedAgentGroup with vWaitTargetTimeout.
23. Set vIWDSegment to 'default'.
24. Route interaction to IWD Agent Group with vWaitTargetTimeout.
25. Log message in case if interaction was from some reasons deleted.
26. Assign last route interaction error to vLastError.
27. Check if route interaction finished with an error.
28. Invoke AssignLastError subroutine with attributes:

- `vInLastErrorkey`—`IWD_Route_Error`
- `vInLastErrorString`—Error description that occurred in route interaction

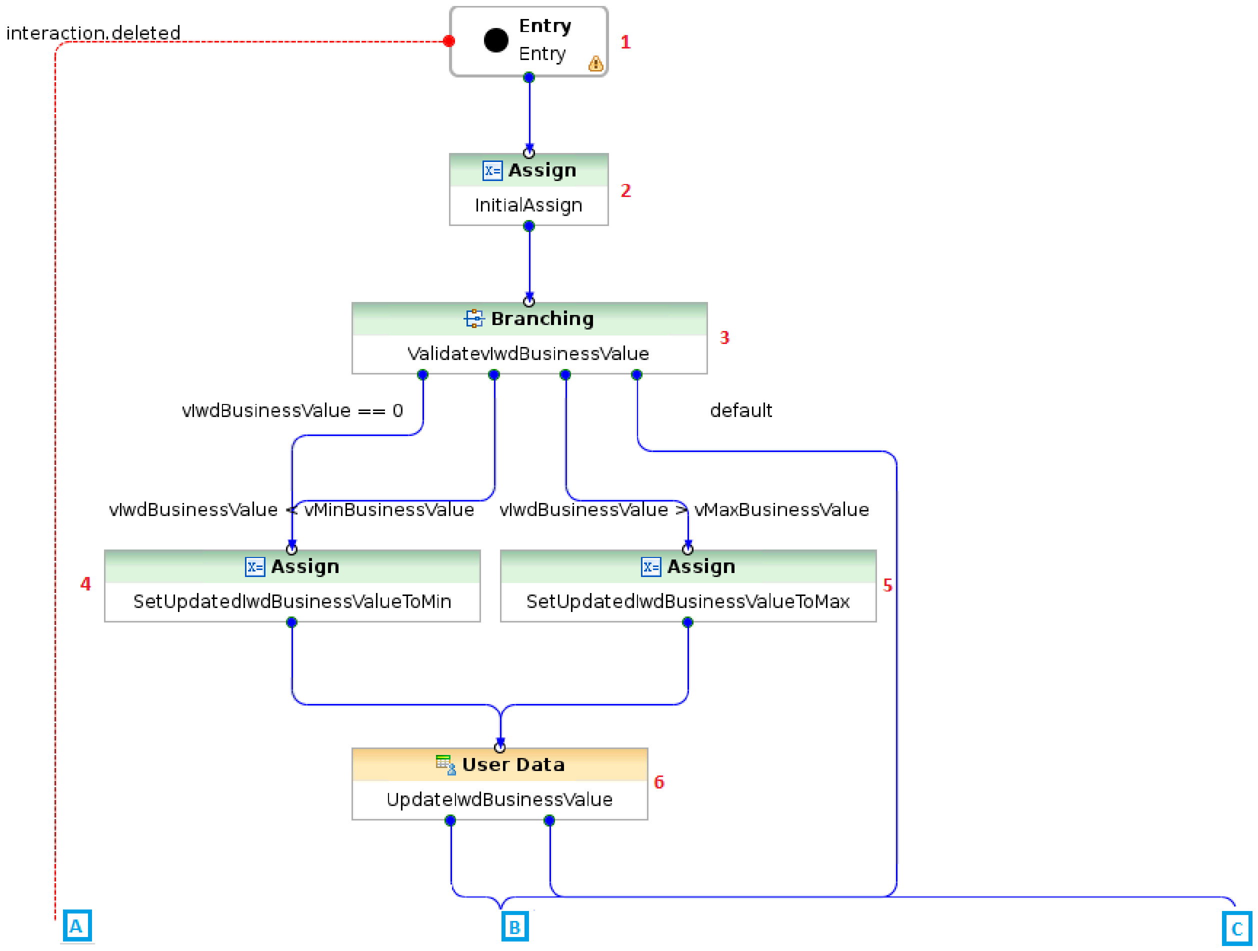
29. Exit Distribution workflow.
30. Cancel event `QueueSubmitGuardEvent` if it was not raised.
31. Exit Distribution workflow.

CheckBusinessValueandPriority Subroutine

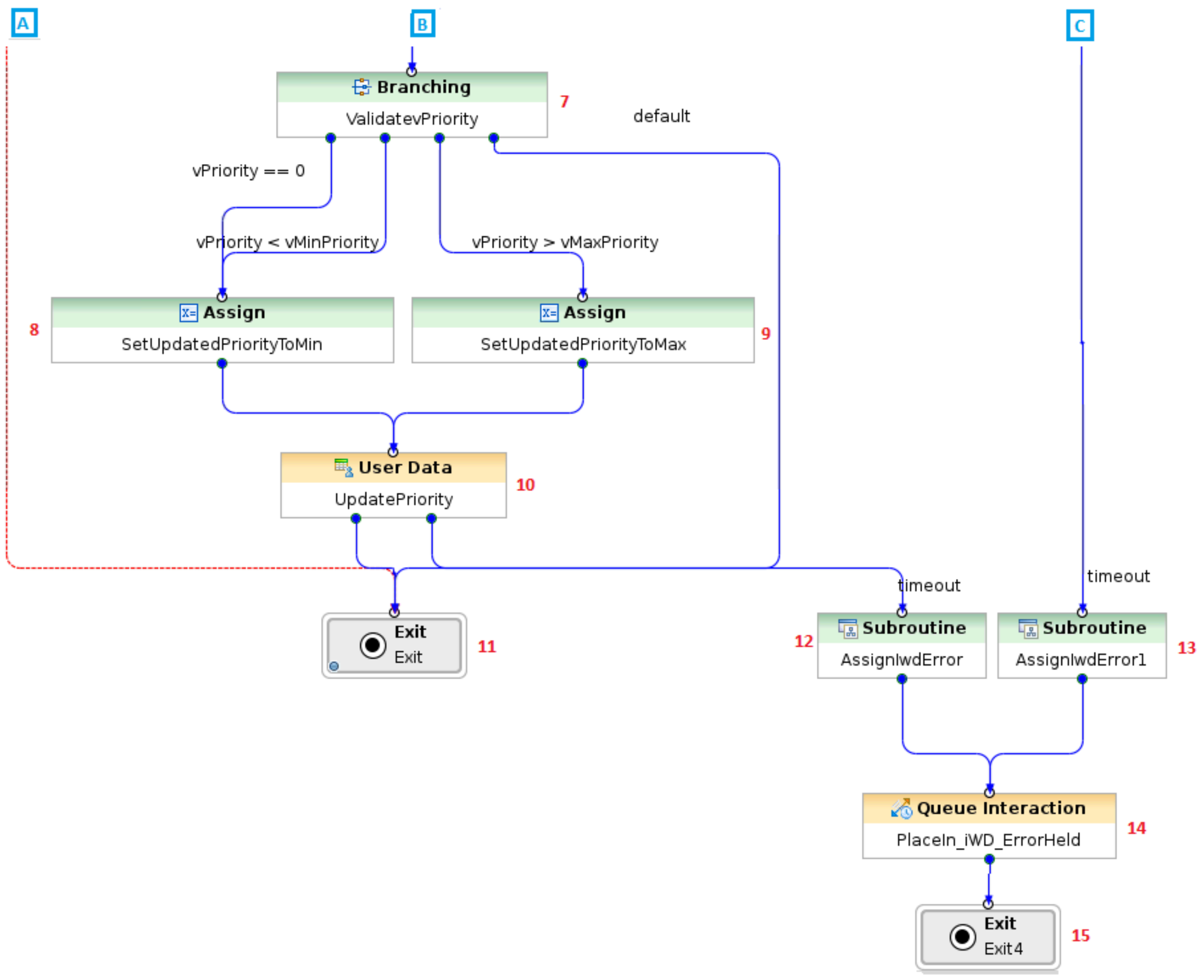
The purpose of this workflow is to verify if `Priority` and `IWD_businessValue` have correct values.

Flow Summary

Part 1


interaction.deleted
Entry
Entry
1
Assign
InitialAssign
2
Branching
ValidatevIwdBusinessValue
3
vIwdBusinessValue == 0
default
vIwdBusinessValue < vMinBusinessValue
vIwdBusinessValue > vMaxBusinessValue
4
Assign
SetUpdatedIwdBusinessValueToMin
Assign
SetUpdatedIwdBusinessValueToMax
5
User Data
UpdateIwdBusinessValue
6
A
B
C

Part 1



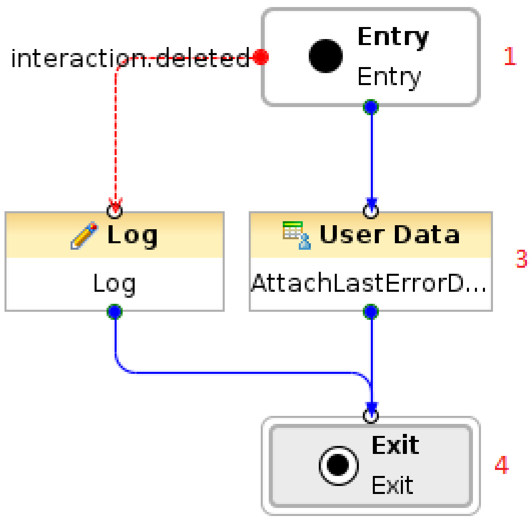
Flow Detail

1. Entry to CheckBusinessValueAndPriority workflow.
2. Variables are initialized:
   - `vIwdBusinessValue`—Read from task attribute `IWD_businessValue`
   - `vIwdPriority`—Read from task attribute `Priority`
3. Validate if `vIwdBusinessValue` is valid.
4. Set `vIwdBusinessValue` to `vMinBusinessValue`.
5. Set `vIwdBusinessValue` to `vMaxBusinessValue`.
6. Update `IWD_businessValue` to `vIwdBusinessValue`.

7. Validate if `vIwdPriority` is valid.
8. Set `vIwdPriority` to `vMinPriority`.
9. Set `vIwdPriority` to `vMaxPriority`.
10. Update Priority to `vIwdPriority`.
11. Exit `CheckBusinessValueAndPriority` workflow.
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString` - Error description: '`Update Priority timeout`'
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString`— Error description: '`Update iWD_businessValue timeout`'
14. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
15. Exit `CheckBusinessValueAndPriority` workflow.

AssignLastError Subroutine

Flow Summary



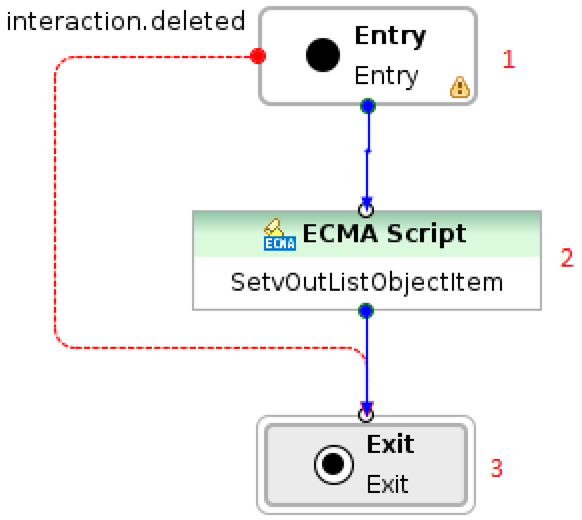
Flow Detail

1. Entry to AssignLastError workflow.
2. The last error is attached to user data as a key-value pair with the key `vInLastErrorkey` and value `vInLastErrorString`.

   `vInLastErrorkey` and `vInLastErrorString` are workflow attributes that need to be set before calling this workflow.
3. Log message if interaction was from some reasons deleted.
4. Exit AssignLastError workflow.

FindListObjectItem Subroutine

Flow Summary



Flow Detail

1. Entry to FindListObjectItem workflow.
2. Search `vKeyToFindInListObject` in `vInListName`.
   - `vInItemName`—Section in `vInListName`
   - `vInListName`—List object where `vKeyToFindInListObject` should be searched

  - `vKeyToFindInListObject`—Option in `vInItemName` that should be found

3. When `vKeyToFindInListObject` is found in `vInListName`, then the value assigned to this option will be assigned to `vOutListObjectItem`.
4. Exit FindListObjectItem workflow.

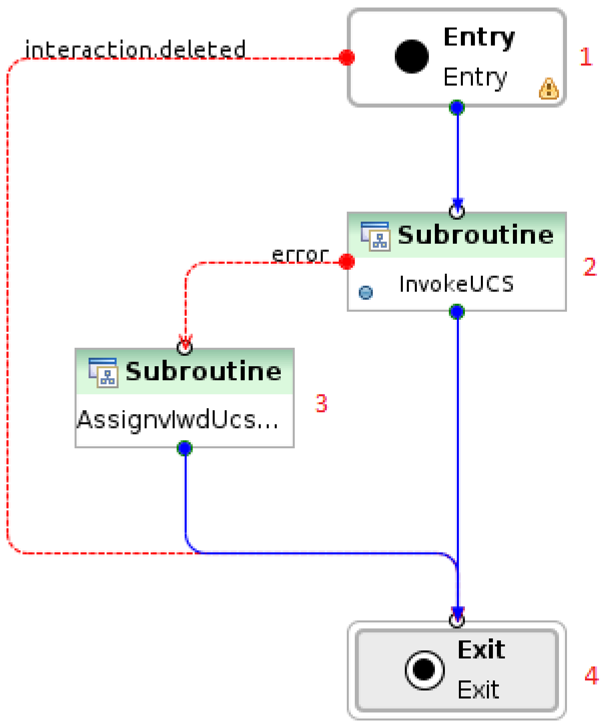
MarkInteractionAsDone Strategy

The purpose of this strategy is to update the Universal Contact Server (UCS) database to mark the interaction as done. This equates to setting the value in the Status column of the Interactions table to 3. UCS clients, such as Interaction Workspace, will then display the status of this interaction as done when the user looks at interactions they have previously processed.

Interactions have to satisfy the following conditions:

- The value of the attached data key `IWD_isContactServer` is `1`
- The value of the attached data key `IWD_isDone` is either null or `0` (zero)

Flow Summary



Flow Detail

1. Entry to MarkInteractionAsDone workflow.
2. The InvokeUCS subroutine is invoked to complete interaction in the UCS database.
3. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_UCS_Error`
   - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine
4. Exit MarkInteractionAsDone workflow.

Removal Strategy

The purpose of this strategy is to delete expired interactions from the Interaction Server database.

A key-value pair in user data with the key `IWD_expirationDateTime` contains information about when an interaction has to be deleted.

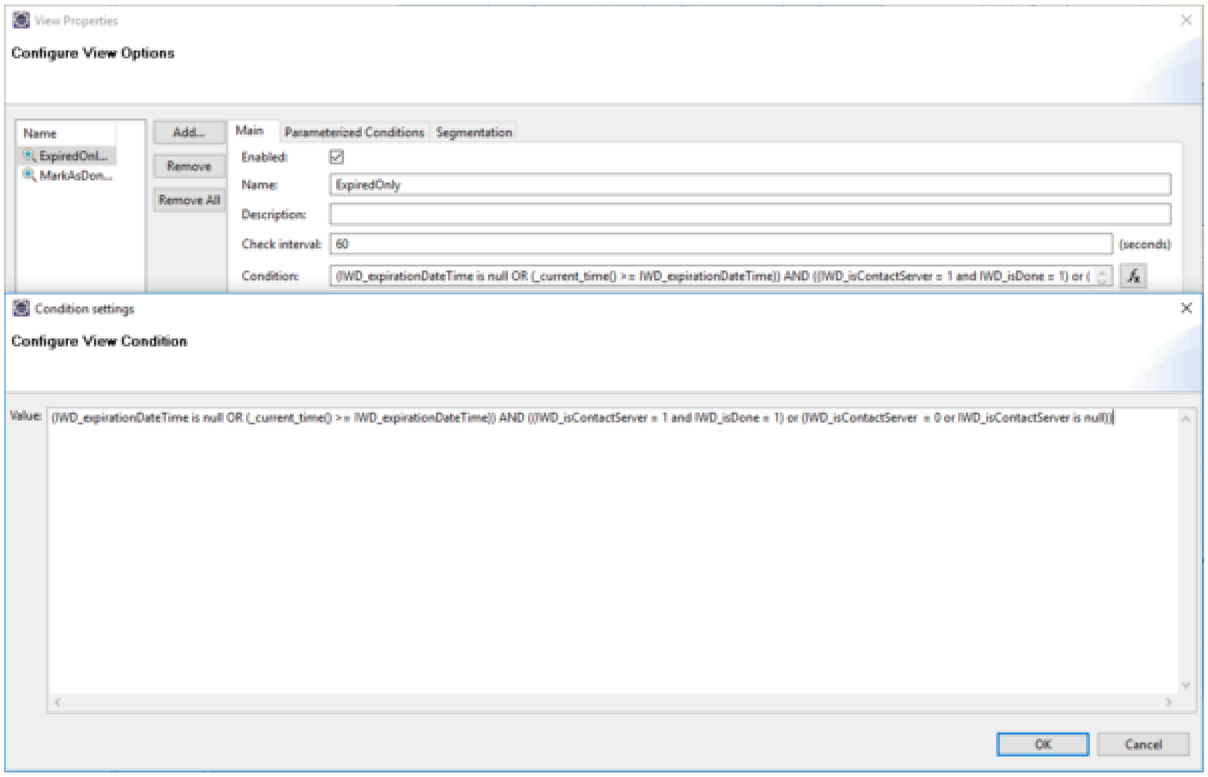
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Rejected`

Interactions have to satisfy the following conditions:

- Interactions must either have the property `IWD_expirationDateTime` not set, or this property must have a time stamp which is in the past.
- If UCS is available, interactions must be marked as done in UCS.
- Interactions are taken in the order they were submitted.

Composer Configuration



Enabled: ✓

Name: ExpiredOnly

Description:

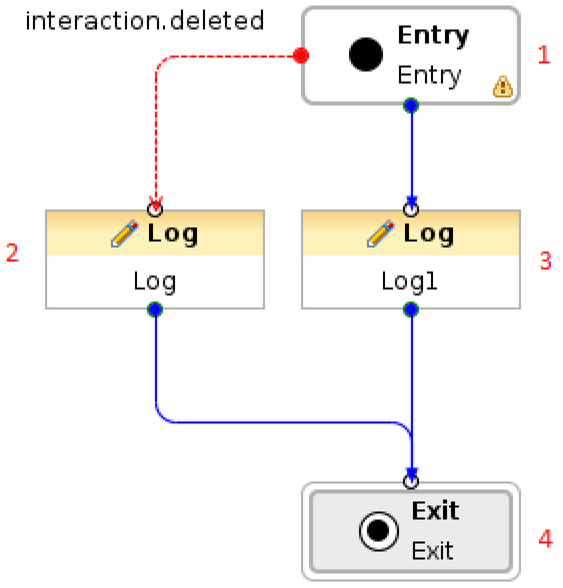
Check interval: 60 (seconds)

Condition: (IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (

Value:

```
(IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (IWD_isContactServer  = 0 or IWD_isContactServer is null))
```

Flow Summary



Flow Detail

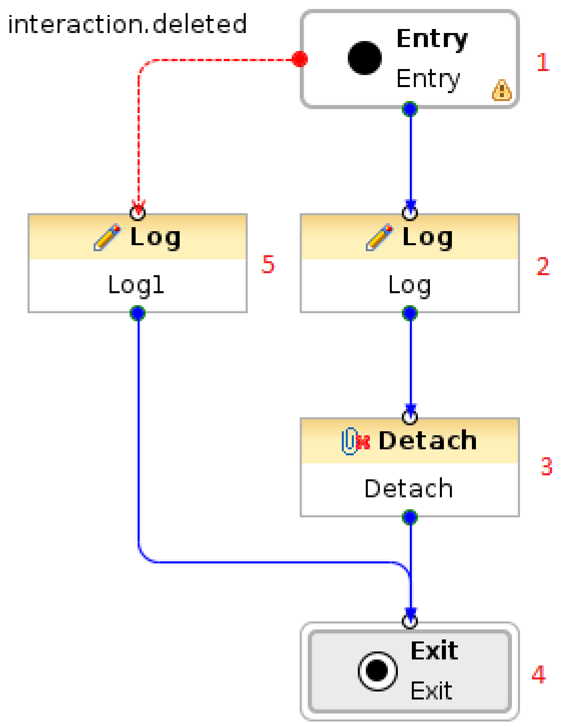
1. Entry to Removal workflow.
2. Log message in case if interaction was from some reasons deleted.
3. Log message: `Task will be terminated on exit`
4. Exit Removal workflow.

Finish Strategy

This workflow detaches interactions from the session. This workflow processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Errorheld`

Flow Summary



Flow Detail

1. Entry to Finish workflow.
2. Log message :'Task processing completed'.
3. Detach interaction. After this operation, the interaction will not be processed any more.
4. Log message in case if interaction was for some reason deleted.
5. Exit Finish workflow.

IWDBP Strategies & Subroutines from 9.0.004 to 9.0.008

Classification Strategy

The purpose of this strategy is to invoke corresponding classification rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

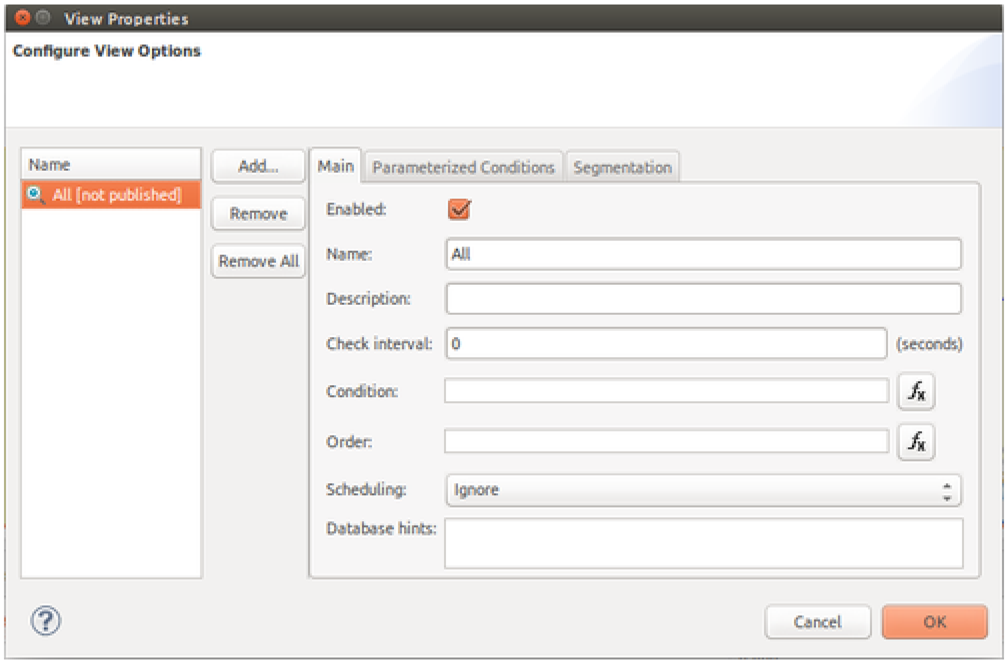
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_New`

Interactions have to satisfy the following conditions:

- There are no conditions here.
- Interactions are taken in order they were submitted.

Composer Configuration



All [not published]

Enabled: ✓

Name: All

Description:

Check interval: 0 (seconds)

Condition:

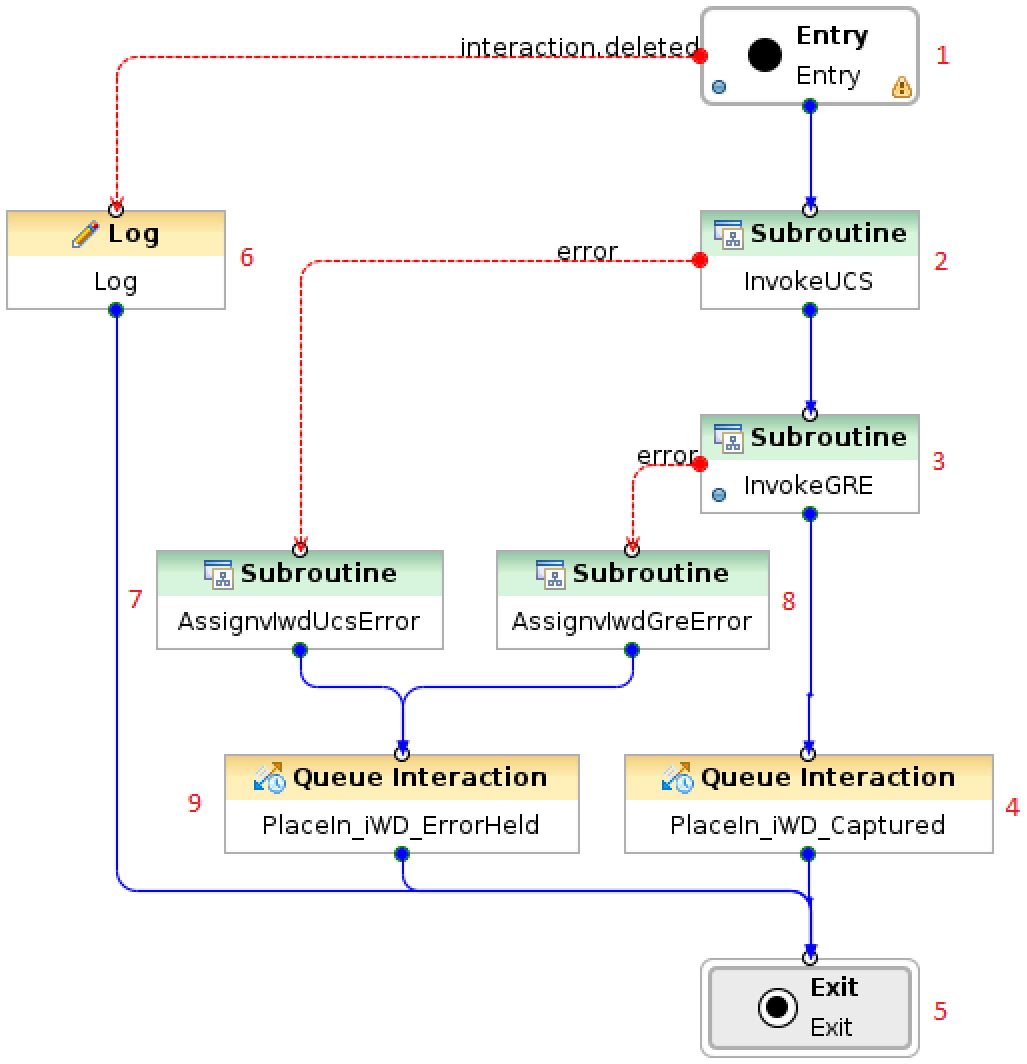
Order:

Scheduling: Ignore

Database hints:

The Composer configuration for this strategy block shows that all interactions are distributed to the `iwd_bp_comp.Main.iWD_New` queue without conditions.

Flow Summary



Flow Detail

1. Entry to Classification workflow.
2. The InvokeUCS subroutine is invoked to create new interaction in the UCS database.
3. The InvokeGRE subroutine is invoked.
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
5. Exit Classification workflow.
6. Log message in case if interaction was from some reasons deleted.

7. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`.
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine.
8. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error description that occured in InvokeGRE subroutine.
9. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

Prioritization Strategy

The purpose of this strategy is to invoke the corresponding prioritization rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

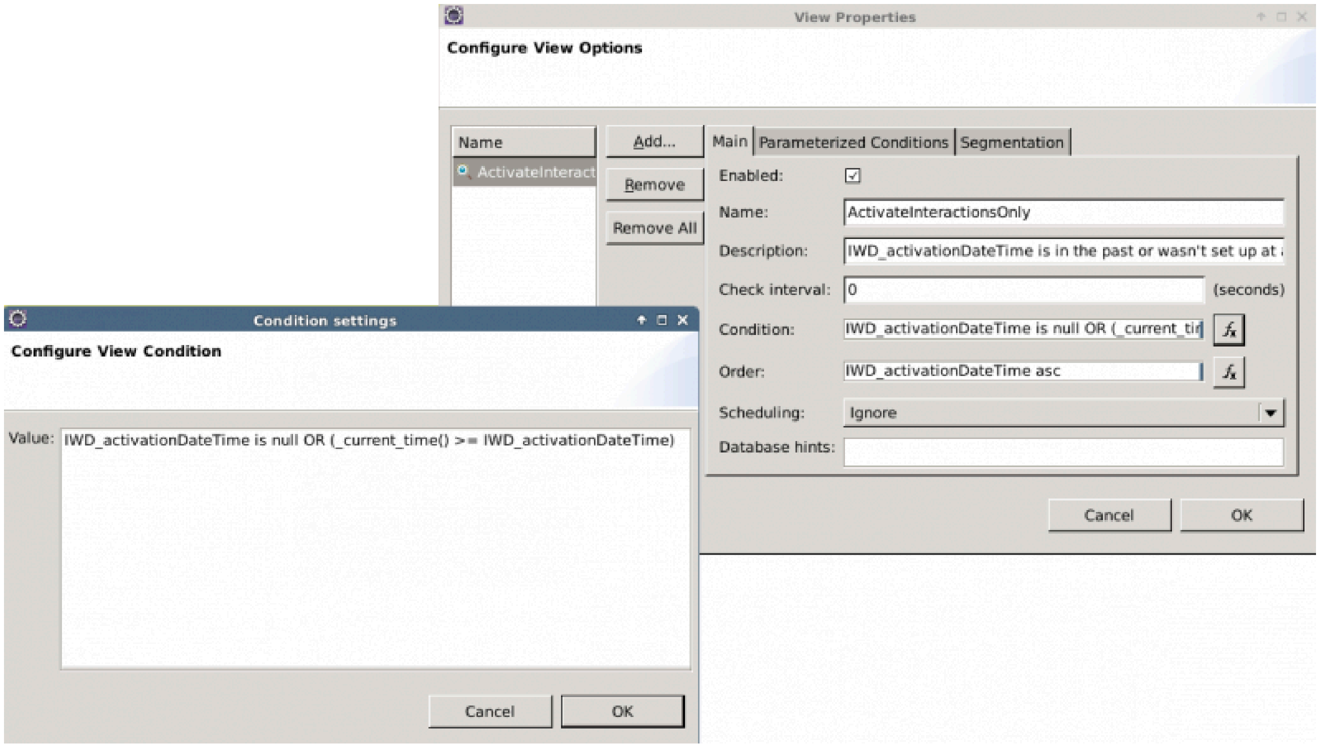
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Captured`

Interactions have to satisfy the following conditions:

- Active interactions only (interactions which do not have the property `IWD_activationDateTime` set, or this property has a time stamp which is in the past.
- Interactions are taken in the order they were submitted.

Composer Configuration



Enabled: ✓

Name: ActivateInteractionsOnly

Description: IWD_activationDateTime is in the past or wasn't set up at

Check interval: 0 (seconds)

Condition: IWD_activationDateTime is null OR (_current_ti

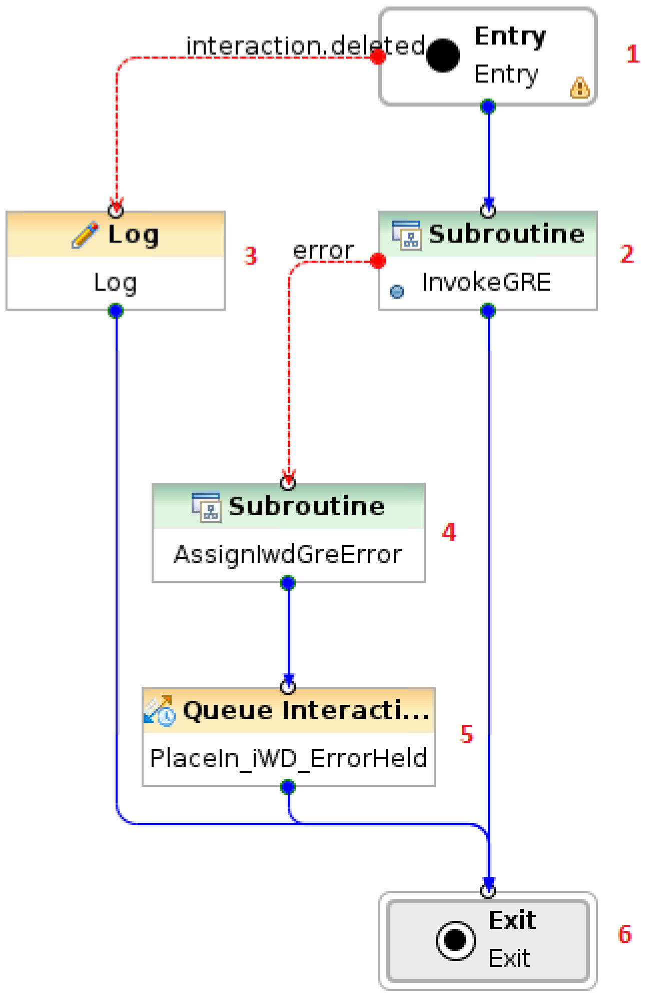
Order: IWD_activationDateTime asc

Scheduling: Ignore

Database hints:

Value: IWD_activationDateTime is null OR (_current_time() >= IWD_activationDateTime)

Flow Summary



Flow Detail

1. Entry to Prioritization workflow.
2. The `InvokeGRE` subroutine is invoked.
3. Log message in case if interaction was from some reasons deleted.
4. Invoke `AssignLastError` subroutine with attributes:
   - `vInLastErrorkey`—`IWD_GRE_Error`
   - `vInLastErrorString`—Error description that occurred in `InvokeGRE` subroutine.
5. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

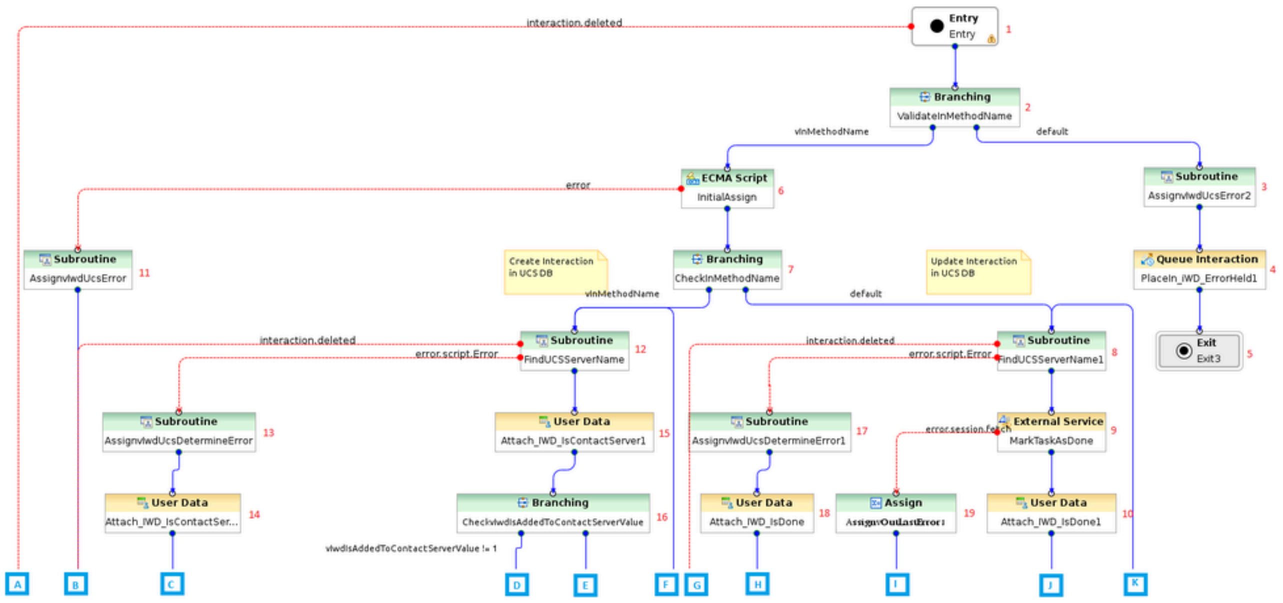
6. Exit Prioritization workflow.

Invoke UCS Strategy

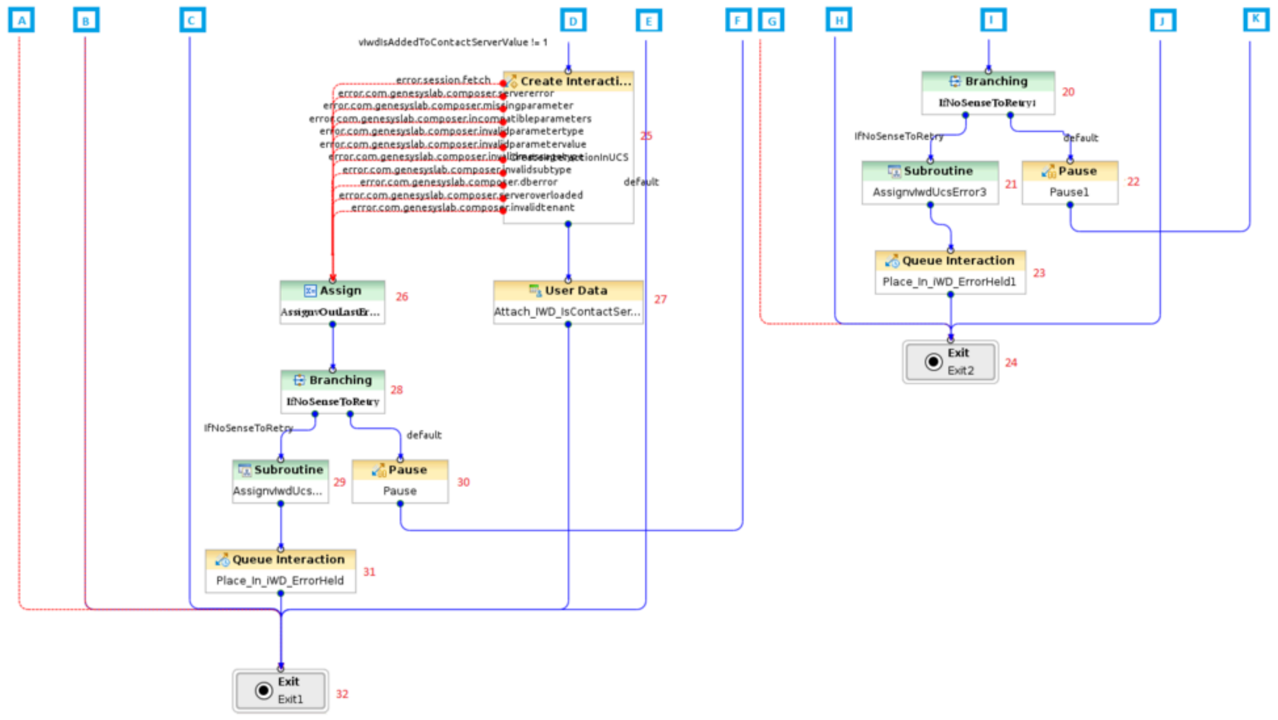
The purpose of this workflow is to create an interaction in the UCS database if UCS is configured.

Flow Summary

Part 1



Part 2


A
B
C
D
E
F
G
H
I
J
K
vIwdIsAddedToContactServerValue != 1
error.session.fetch
error.com.genesyslab.composer.servererror
error.com.genesyslab.composer.missingparameter
error.com.genesyslab.composer.incompatibleparameters
error.com.genesyslab.composer.invalidparametertype
error.com.genesyslab.composer.invalidparametervalue
error.com.genesyslab.composer.invalidresponsetype
error.com.genesyslab.composer.invalidsubtype
error.com.genesyslab.composer.dberror
error.com.genesyslab.composer.serveroverloaded
error.com.genesyslab.composer.invalidtenant
Create Interacti...
CreateInteractionInUCS
default
25
Assign
AssignvOutLastEr...
26
User Data
Attach_IWD_IsContactSer...
27
Branching
IfNoSenseToRetry
28
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcs...
29
Pause
Pause
30
Queue Interaction
Place_In_iWD_ErrorHeld
31
Exit
Exit1
32
Branching
IfNoSenseToRetry
20
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcsError3
21
Pause
Pause1
22
Queue Interaction
Place_In_iWD_ErrorHeld1
23
Exit
Exit2
24

Flow Detail

1. Entry InvokeUCS strategy.
2. Check if `in_method_name = 'Create' | in_method_name = 'OMInteractions'`.
3. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_UCS_Error`
   - `vInLastErrorString`—Error informs that: `vInMethodName + ' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeUCS workflow.
6. Variables are initialized:
   - `vExternalId`—Read from task attribute `ExternalId`
   - `vMediaType`—Read from task attribute
   - `vSubmittedBy`—Read from task attribute `attr_itx_submitted_by`
   - `vType`—Read from task attribute '`InteractionType`'
   - `vSubType`—Read from task attribute '`InteractionSubtype`'
   - `vIwdIsAddedToContactServerValue`—Read from task attribute '`IWD_isAddedToContactServer`'

7. Check if `in_method_name = 'Create'`.
8. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
   - `vInItemName—ContactServerList`
   - `vInListName—Iwd_Esp_List`
9. The strategy calls a method on the Universal Contact Server to set the status of the interaction to 3, indicating that the interaction is done.
10. The value of the user data key `IWD_isDone` is set to `1`.
11. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`'
    - `vInLastErrorString`—Error description that occurred when variables were initialized
12. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
    - `vInItemName—ContactServerList`
    - `vInListName—Iwd_Esp_List`
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`'
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
14. The value of the user data key `IWD_isContactServer` is set to `0`.
15. The value of the user data key `IWD_isContactServer` is set to `1`.
16. Check if `IWD_isContactServer` is set to `1`.
17. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
18. The value of the user data key `IWD_isDone` is set to `0`.
19. An error is extracted from user data and assigned in `vLastError` variable.
20. If it makes sense to retry updating the interaction record in UCS.
21. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry update interaction in UCS.
22. A delay is introduced into the processing. Flow returns to step 8.
23. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
24. Exit InvokeUCS workflow.
25. A new interaction is created in the UCS database, for this iWD task.
26. An error is extracted from user data and assigned in `vLastError` variable.

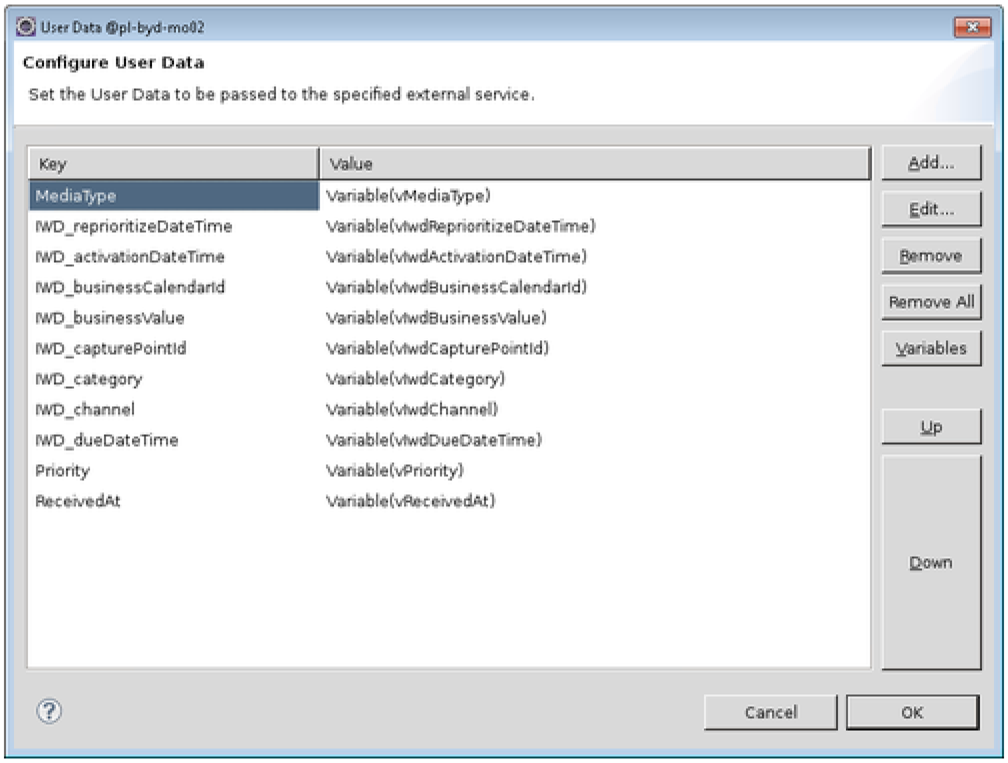
27. The user data key `IWD_isAddedToContactServer` is updated to 1 to indicate that the task was successfully added to the interaction history in UCS. The result returned from the ESP call to UCS (from See A new interaction is created in the UCS database, for this iWD task. If that function is successful is written to the variable `IWD_UCS_Result`.
28. If it makes sense to retry creating the interaction record in UCS.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry create interaction in UCS
30. A delay is introduced into the processing. Flow returns to step 12.
31. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
32. Exit InvokeUCS workflow.

InvokeGRE Strategy

Important

For Composer/ORS versions prior 8.1.400.48—If custom task attributes will be used in the Standard Rules Template, you must add them in the External Service block called `InvokeGRE` in the InvokeGRE workflow. All user-defined attributes need to be added in the User Data attribute, otherwise they will not be attached to the task and so will not be sent in the ESP request to the external ESP service.

Composer configuration



Configure User Data

Set the User Data to be passed to the specified external service.

Key	Value
MediaType	Variable(vMediaType)
IWD_reprioritizeDateTime	Variable(vIwdReprioritizeDateTime)
IWD_activationDateTime	Variable(vIwdActivationDateTime)
IWD_businessCalendarId	Variable(vIwdBusinessCalendarId)
IWD_businessValue	Variable(vIwdBusinessValue)
IWD_capturePointId	Variable(vIwdCapturePointId)
IWD_category	Variable(vIwdCategory)
IWD_channel	Variable(vIwdChannel)
IWD_dueDateTime	Variable(vIwdDueDateTime)
Priority	Variable(vPriority)
ReceivedAt	Variable(vReceivedAt)

Pause block configuration

During the task lifecycle, GRE changes interaction properties. This is done asynchronously, so sometimes ORS does not have enough time to receive confirmation event from Interaction Server and continue execution of the workflow. This could lead to unexpected behavior—for example, tasks could go to the `ErrorHeld` queue sporadically without visible reasons.

A **Pause** block with a configurable delay has been added into the InvokeGRE workflow after the **AfterESPCallActivities** block to guarantee that interaction updates will be received. This delay is set in milliseconds. By default it is set to `0`.

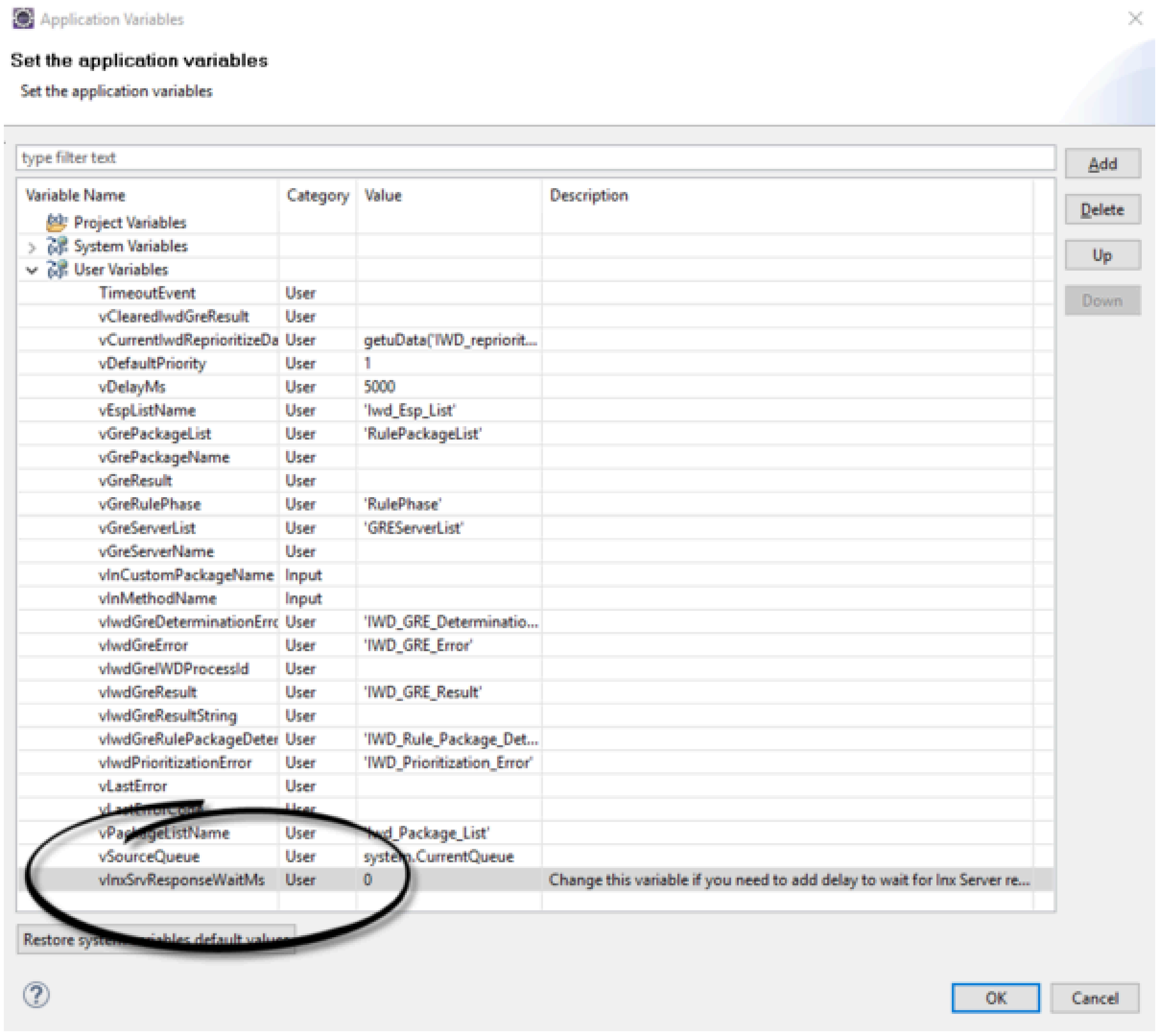
If it is observed that ORS has already moved to the next workflow step but has still not received a confirmation event from Interaction Server, then this delay needs to be configured. The value should be calculated individually and specifically, depending on the delay in the Interaction Server response.

Important

The delay must be set accurately, taking into account that this pause will happen during each InvokeGRE execution. If there are many interactions, the total delay could be substantial.

The delay can be set via the `vInxSrvResponseWaitMs` variable. To set its value, do the following:

1. Click on the **Entry** block of the **InvokeGRE** strategy.
2. Select the **Properties** tab at the bottom of the Composer window.
3. Click on the dots next to **Global Settings -> Variables** to open the **Application Variables** window.
4. Expand the **User Variables** item and set the `vInxSrvResponseWaitMs` value.



Set the application variables

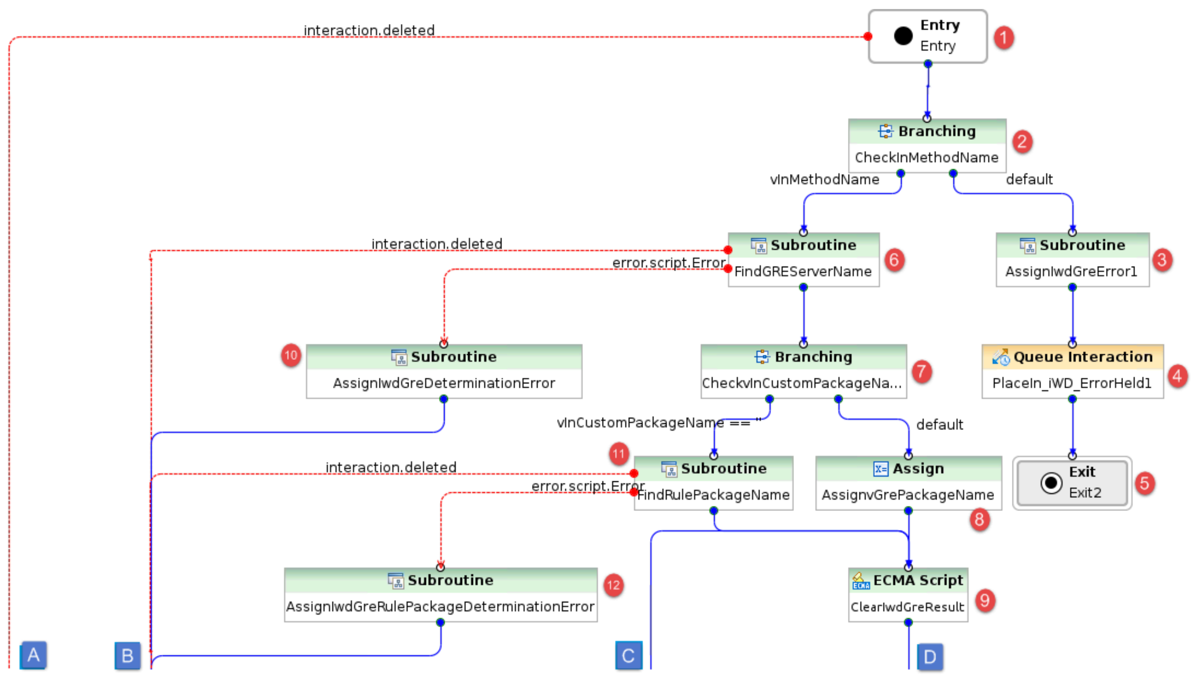
Set the application variables

type filter text

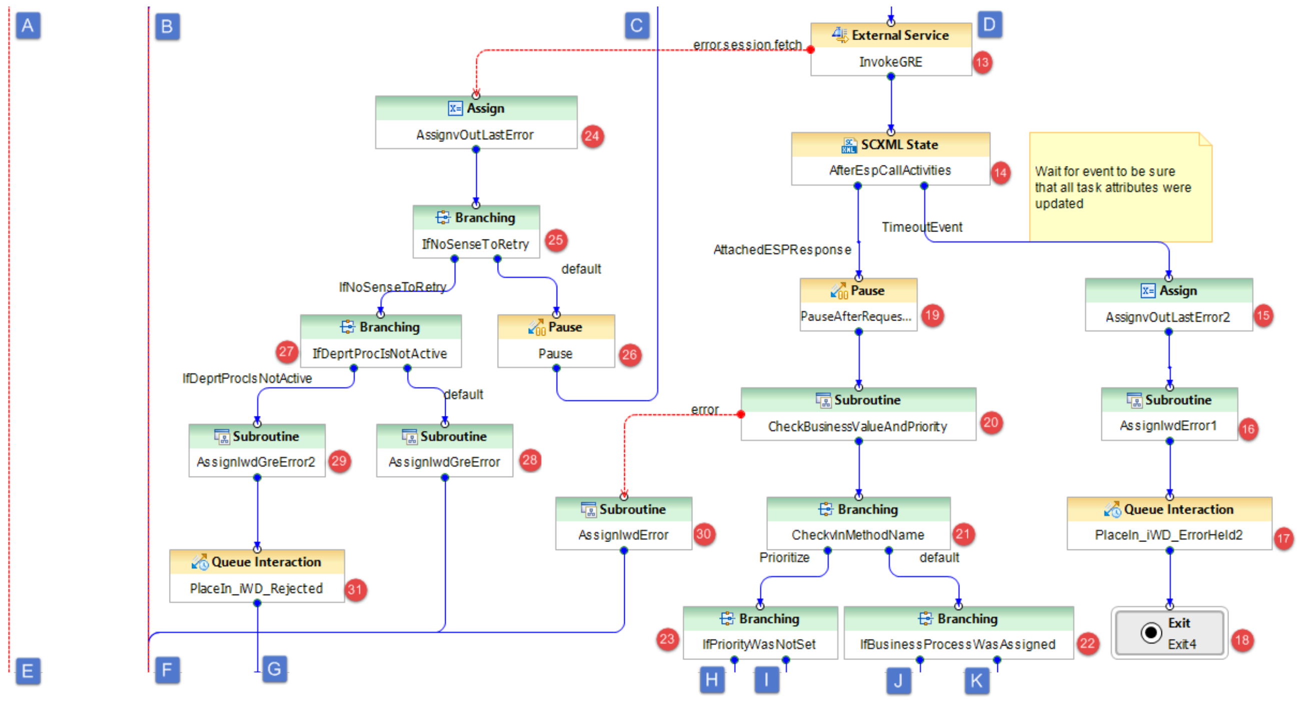
Variable Name	Category	Value	Description
Project Variables			
System Variables			
User Variables			
TimeoutEvent	User		
vClearedIwdGreResult	User		
vCurrentIwdReprioritizeDa	User	getuData('IWD_repriorit...	
vDefaultPriority	User	1	
vDelayMs	User	5000	
vEspListName	User	'Iwd_Esp_List'	
vGrePackageList	User	'RulePackageList'	
vGrePackageName	User		
vGreResult	User		
vGreRulePhase	User	'RulePhase'	
vGreServerList	User	'GREServerList'	
vGreServerName	User		
vInCustomPackageName	Input		
vInMethodName	Input		
vIwdGreDeterminationErrc	User	'IWD_GRE_Determinatio...	
vIwdGreError	User	'IWD_GRE_Error'	
vIwdGreIWDProcessId	User		
vIwdGreResult	User	'IWD_GRE_Result'	
vIwdGreResultString	User		
vIwdGreRulePackageDeter	User	'IWD_Rule_Package_Det...	
vIwdPrioritizationError	User	'IWD_Prioritization_Error'	
vLastError	User		
vLastErrorCode	User		
vPackageListName	User	'Iwd_Package_List'	
vSourceQueue	User	system.CurrentQueue	
vInxSrvResponseWaitMs	User	0	Change this variable if you need to add delay to wait for Inx Server re...

Flow Summary

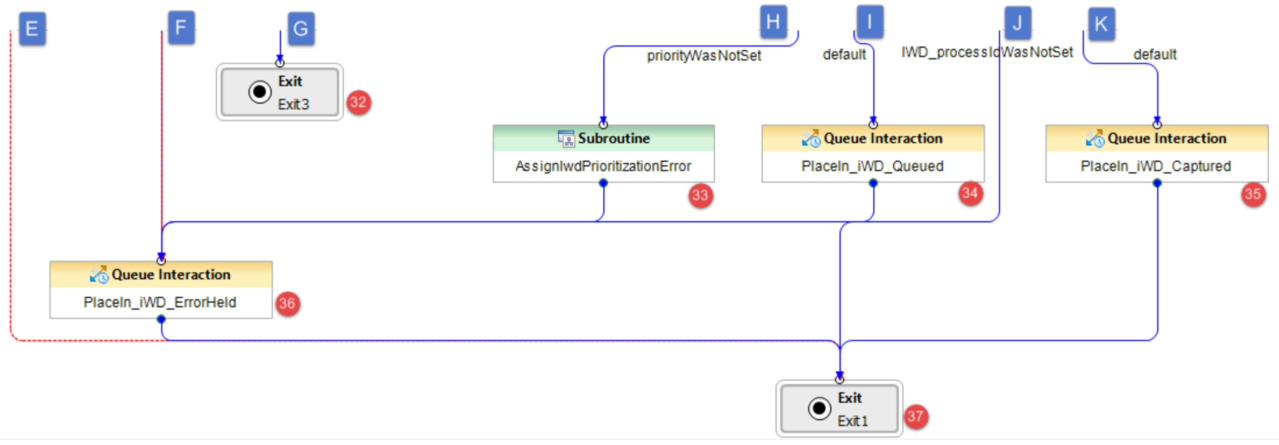
Part 1



Part 2



Part 3



Flow Detail

1. Entry to `InvokeGRE` strategy.
2. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
3. Invoke `AssignLastError` subroutine with attributes:

- `vInLastErrorkey`—`IWD_GRE_Error`
- `vInLastErrorString`—Error informs that: `vInMethodName` + `' is not valid'`

4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeGRE workflow.
6. The `FindListObjectItem` subroutine is invoked to determine the name of the Genesys Rules Engine Application. The subroutine uses the List Object list `GREServerList`:
    - `vInItemName`—`GREServerList`
    - `vInListName`—`Iwd_Esp_List`
7. Check if `vInCustomPackageName` was published to this subroutine. If it is set then `vInCustomPackageName` will be run. Otherwise package name needs to be found in `Iwd_Package_List`.
8. Assign `vInCustomPackageName` to `vGrePackageName`.
9. Delete `IWD_GRE_Result`, `IWD_Error`, `RulePhase` before Invoke GRE.
10. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
11. The `FindListObjectItem` subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules:
    - `vInItemName`—`RulePackageList`
    - `vInListName`—`Iwd_Package_List`
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Rule_Package_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
13. An ESP request is sent to the Genesys Rules Engine to evaluate the classification rules.

> **Important**
> All user data that needs to be added to ESP request must be added in User Data attributes.

14. Parse ESP result and attach to the interaction all attributes modified by the GRE.
15. Assign the string `AfterEspCallActivities timeout` to the `vLastError` variable.
16. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `Attach GreResult timeout`
17. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
18. Exit InvokeGRE workflow.
19. A delay is introduced, based on the value of the `vInxSrvResponseWaitMs` variable which can be set in

the Entry block.

20. CheckBusinessValueAndPriority subroutine is called to verify if `IWD_businessValue` and `Priority` have correct values.
21. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
22. Check if `IWD_processId` was set by any rules or when task was created.
23. Check is made to see if this is the first time that prioritization rules are being evaluated for the interaction, and the priority was not set up by any rules.
24. Get last error that was occured in GRE call and assign it to `vLastError` variable.
25. A check is done to see if the error code is related to the ESP server communication.
26. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to step 11 to retry the connection to the ESP server.
27. The last Interaction Server-related error is extracted from a variable.
28. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
30. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable
31. The interaction is placed in the `iwd_bp_comp.Main.iWD_Rejected` queue.
32. Exit InvokeGRE workflow.
33. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Prioritization_Error`
    - `vInLastErrorString`—Error description: '`Priority is not set up by rules`'.
34. The interaction is placed in the `iwd_bp_comp.Main.iWD_Queued` queue.
35. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
36. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
37. Exit InvokeGRE workflow.

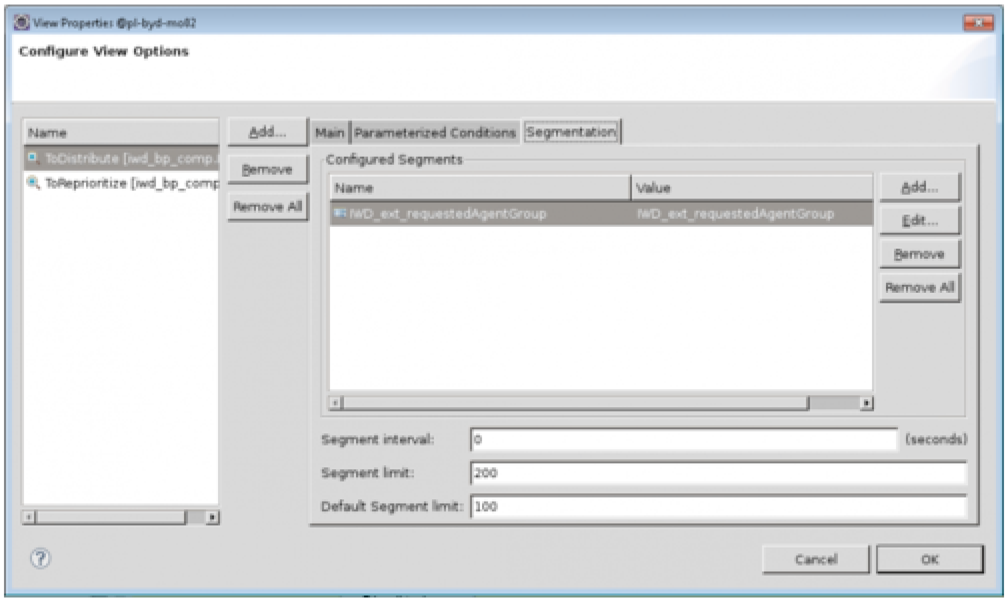
Distribution Strategy

This strategy routes interactions to a requested Agent, requested Agent Group, requested Skill, or to the default iWD Agent Group. This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are not subject for immediate reprioritization (interactions that do not have the property `IWD_reprioritizeDateTime` set, or that have this property set to a time stamp that is in the future).
    - Interactions are taken in order of priority (highest priority first)

A Segmentation feature ensures that all agents can be kept busy by distributing tasks in each segment separately. As a result, even in a Distribution strategy that is populated by high-priority tasks assigned to small groups of agents, the strategy will not become so saturated that distribution of tasks to other agents is blocked. Segmentation settings are found in the **ToDistribute** view of the Distribution routing strategy. The Distribution strategy can make a call to the segmentation setting and add an `IWD_Segment` attribute to the interaction data.

Composer Configuration - Segmentation View


View Properties: @pl-byd-mo82

Configure View Options

Name
ToDistribute [iwd_bp_comp.
ToReprioritize [iwd_bp_comp

Configured Segments

Name	Value
IWD_ext_requestedAgentGroup	IWD_ext_requestedAgentGroup

Segment interval: 0 (seconds)

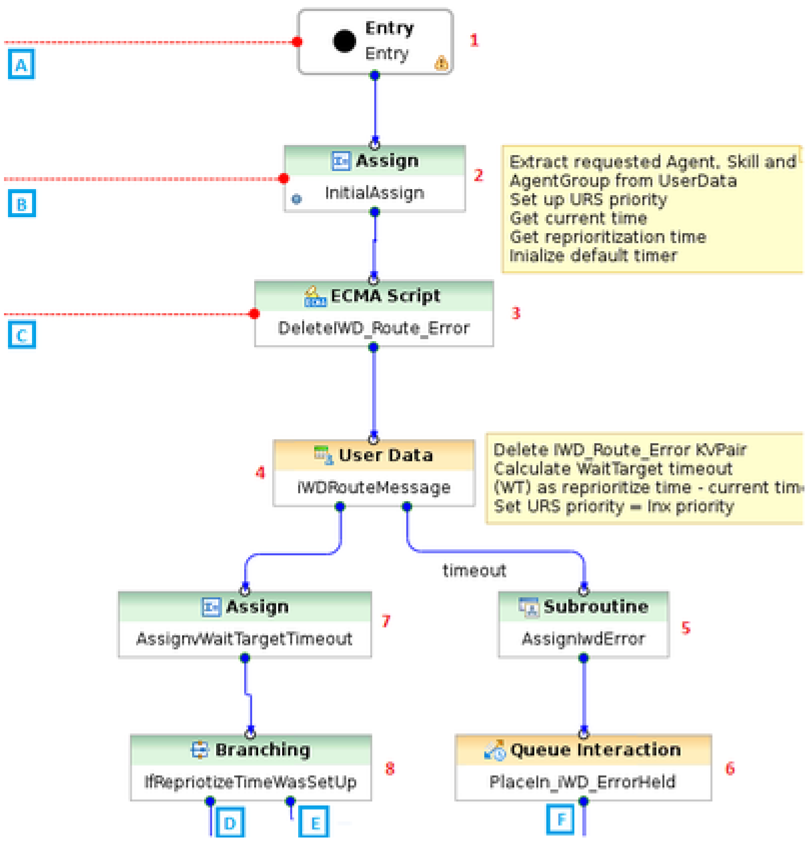
Segment limit: 200

Default Segment limit: 100

Flow Summary

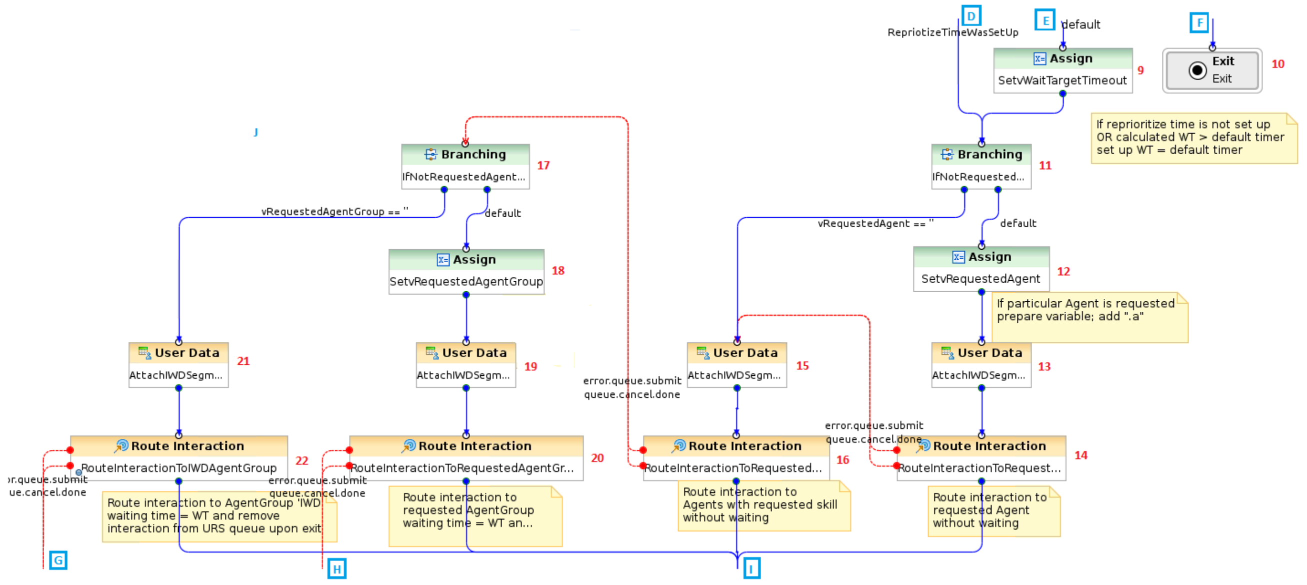
Part 1

Click to enlarge.



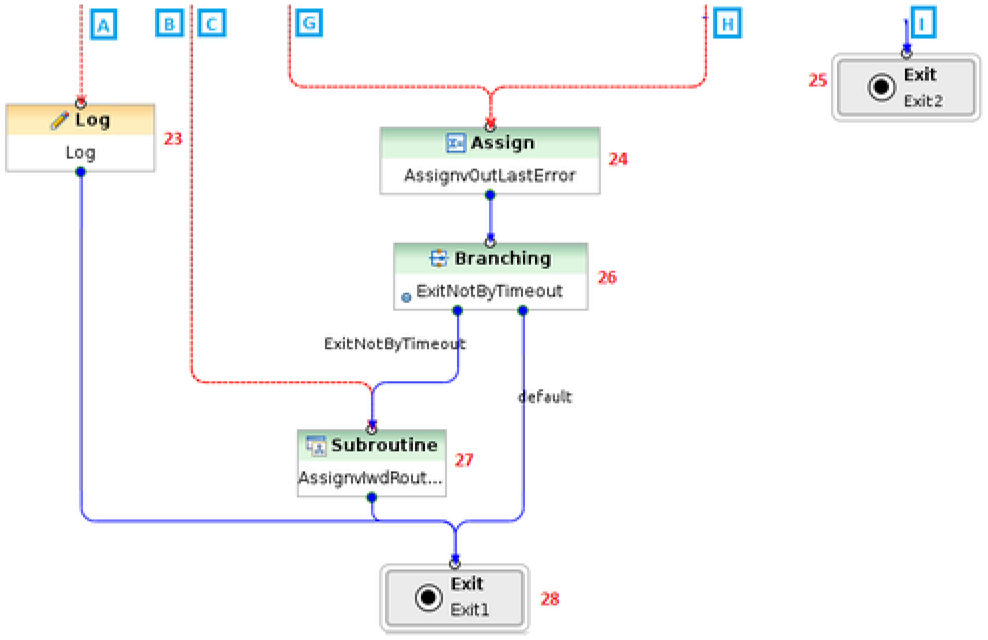
Part 2

Click to enlarge.



Part 3

Click to enlarge.



Flow Detail

1. Entry to Distribution workflow.
2. Variables are initialized:
   - vRequestedAgentGroup—Read from task attribute IWD_ext_requestedAgentGroup
   - vRequestedAgentGroup—Read from task attribute IWD_ext_requestedAgent
   - vRequestedSkill—Read from task attribute IWD_ext_requestedSkill
   - vCurrentTint—Current time in seconds
   - vReprioritizeDint—Read from task attribute IWD_businessValue
   - vDefaultTargetTimeout—Default target timeout set to 3600 seconds
   - vInxPriority—Read from task attribute Priority
3. Delete IWD_Route_Error from attached data. Calculate WaitTarget timeout based on vReprioritizeDTInt and vCurrentDTInt. Sets URS priority.
4. Set information about clear IWD_Route_Error attribute.
5. Invoke AssignLastError subroutine with attributes:
   - vInLastErrorkey—IWD_Error
   - vInLastErrorString—Error description: 'Update IWD_Route_Error timeout'
6. The interaction is placed in the iwd_bp_comp.Main.iWD_ErrorHeld queue.

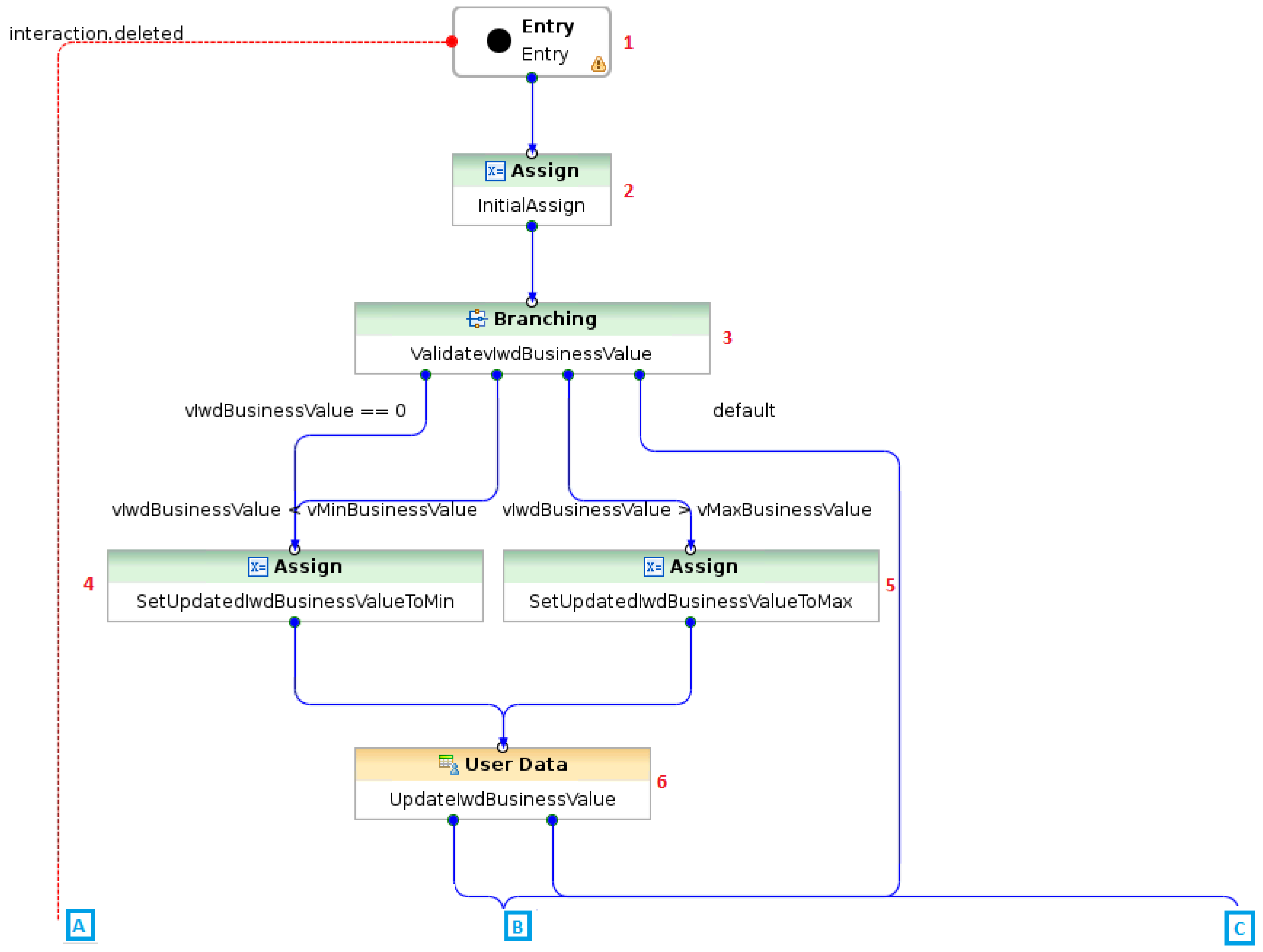
7. Calculate `vWaitTargetTimeout`.
8. Check if calculated `vWaitTargetTimeout` is in range `(0, vDefaultTargetTimeout>`.
9. Set `vWaitTargetTimeout` to `vDefaultTargetTimeout`.
10. Exit Distribution workflow.
11. Check if particular Agent is requested.
12. Assign `vRequestedAgent` + '`.a`' to `vRequestedAgent` variable.
13. Set `vIWDSegment` to '`_requested_agent`'.
14. Route interaction to requested `vRequestedAgent` without waiting.
15. Set `vIWDSegment` to '`_requested_skill`'.
16. Route interaction to requested `vRequestedAgent` with requested skill without waiting.
17. Check if particular `AgentGroup` is requested.
18. Assign `vRequestedAgentGroup` + '.qa' to `vRequestedAgentGroup` variable.
19. Set `vIWDSegment` to '`_requested_agent_group`'.
20. Route interaction to requested `vRequestedAgentGroup` with `vWaitTargetTimeout`.
21. Set `vIWDSegment` to '`default`'.
22. Route interaction to IWD Agent Group with `vWaitTargetTimeout`.
23. Log message in case if interaction was from some reasons deleted.
24. Assign last route interaction error to `vLastError`.
25. Exit Distribution workflow.
26. Check if route interaction finished with an error.
27. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Route_Error`
    - `vInLastErrorString`—Error description that occurred in route interaction
28. Exit Distribution workflow.

CheckBusinessValueandPriority Subroutine

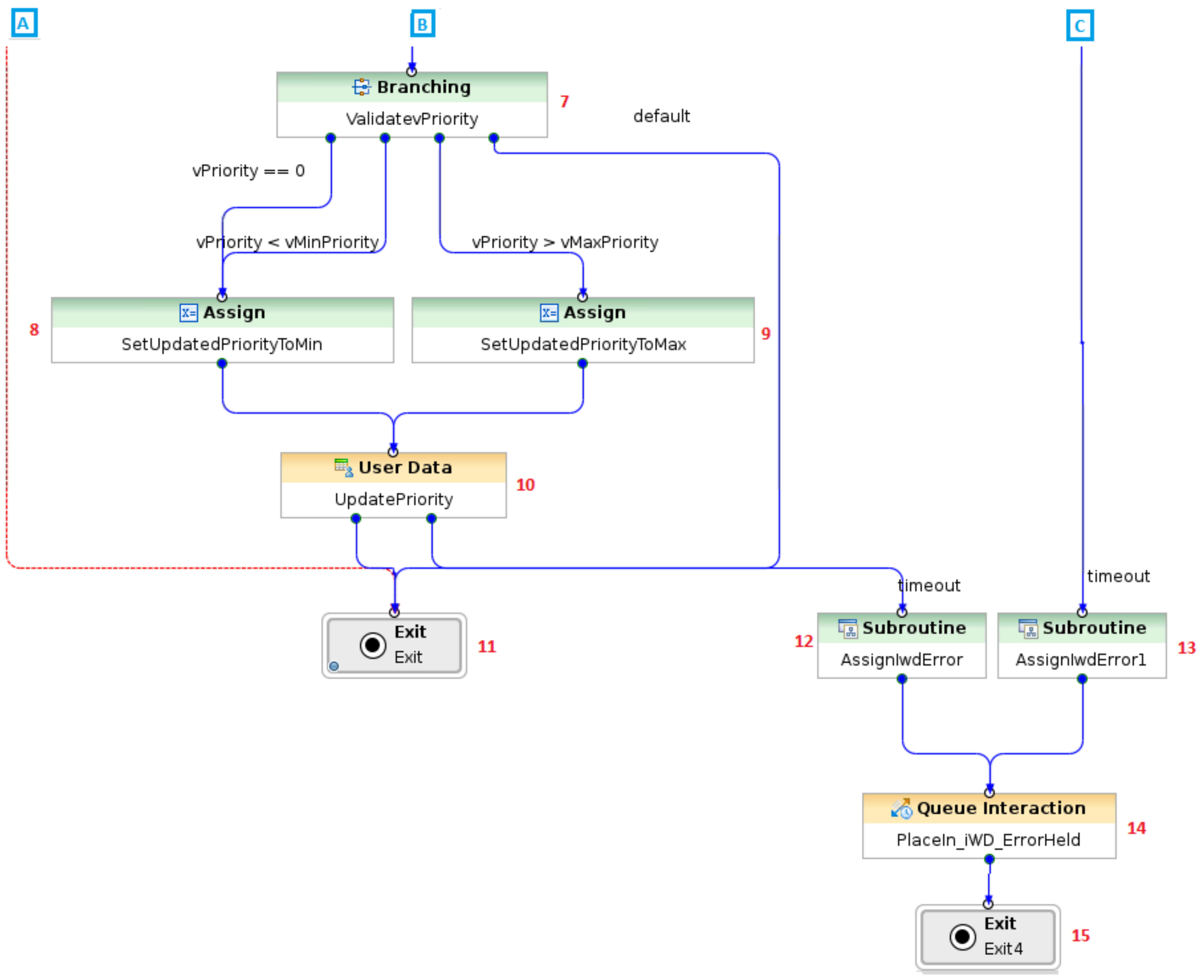
The purpose of this workflow is to verify if `Priority` and `IWD_businessValue` have correct values.

Flow Summary

Part 1


interaction.deleted
Entry
Entry
1
Assign
InitialAssign
2
Branching
ValidatevIwdBusinessValue
3
vIwdBusinessValue == 0
default
vIwdBusinessValue < vMinBusinessValue
vIwdBusinessValue > vMaxBusinessValue
4
Assign
SetUpdatedIwdBusinessValueToMin
Assign
SetUpdatedIwdBusinessValueToMax
5
User Data
UpdateIwdBusinessValue
6
A
B
C

Part 1



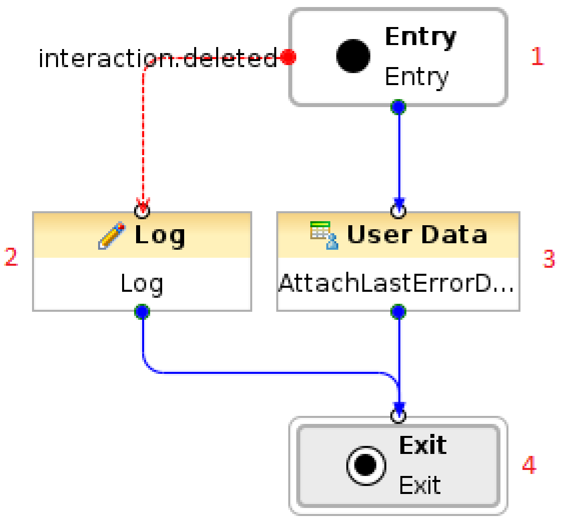
Flow Detail

1. Entry to CheckBusinessValueAndPriority workflow.
2. Variables are initialized:
   - `vIwdBusinessValue`—Read from task attribute `IWD_businessValue`
   - `vIwdPriority`—Read from task attribute `Priority`
3. Validate if `vIwdBusinessValue` is valid.
4. Set `vIwdBusinessValue` to `vMinBusinessValue`.
5. Set `vIwdBusinessValue` to `vMaxBusinessValue`.
6. Update `IWD_businessValue` to `vIwdBusinessValue`.

7. Validate if `vIwdPriority` is valid.
8. Set `vIwdPriority` to `vMinPriority`.
9. Set `vIwdPriority` to `vMaxPriority`.
10. Update Priority to `vIwdPriority`.
11. Exit `CheckBusinessValueAndPriority` workflow.
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString` - Error description: '`Update Priority timeout`'
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString`— Error description: '`Update iWD_businessValue timeout`'
14. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
15. Exit `CheckBusinessValueAndPriority` workflow.

AssignLastError Subroutine

Flow Summary



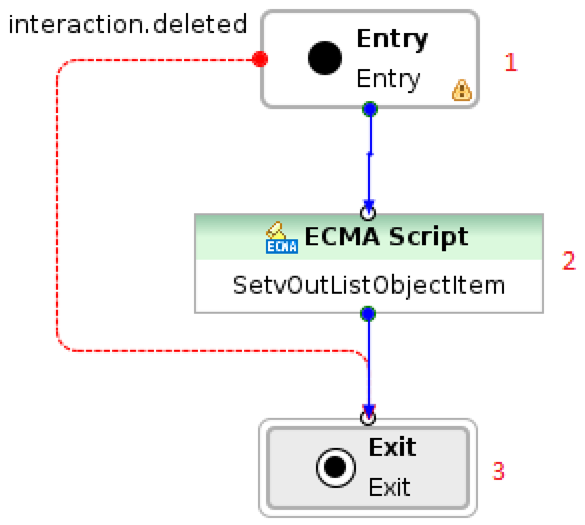
Flow Detail

1. Entry to AssignLastError workflow.
2. The last error is attached to user data as a key-value pair with the key `vInLastErrorkey` and value `vInLastErrorString`.

   `vInLastErrorkey` and `vInLastErrorString` are workflow attributes that need to be set before calling this workflow.
3. Log message if interaction was from some reasons deleted.
4. Exit AssignLastError workflow.

FindListObjectItem Subroutine

Flow Summary



Flow Detail

1. Entry to FindListObjectItem workflow.
2. Search `vKeyToFindInListObject` in `vInListName`.
   - `vInItemName`—Section in `vInListName`
   - `vInListName`—List object where `vKeyToFindInListObject` should be searched

- `vKeyToFindInListObject`—Option in `vInItemName` that should be found

3. When `vKeyToFindInListObject` is found in `vInListName`, then the value assigned to this option will be assigned to `vOutListObjectItem`.
4. Exit FindListObjectItem workflow.

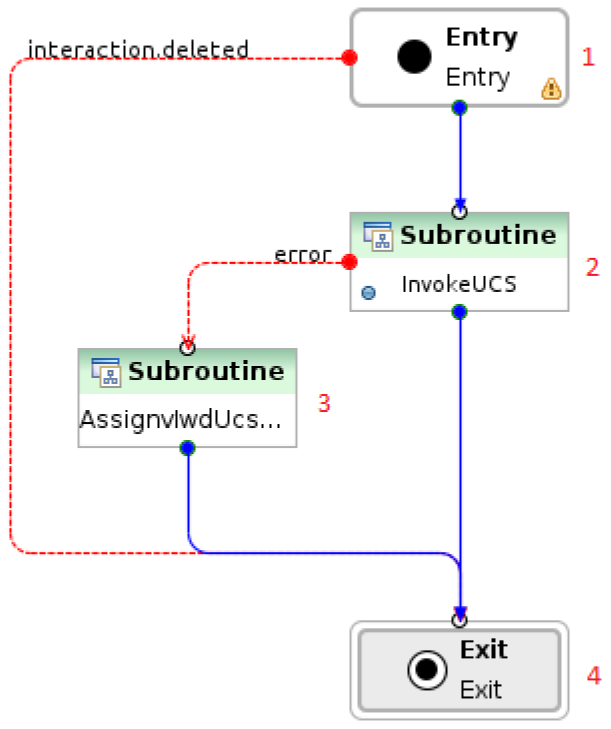
MarkInteractionAsDone Strategy

The purpose of this strategy is to update the Universal Contact Server (UCS) database to mark the interaction as done. This equates to setting the value in the Status column of the Interactions table to 3. UCS clients, such as Interaction Workspace, will then display the status of this interaction as done when the user looks at interactions they have previously processed.

Interactions have to satisfy the following conditions:

- The value of the attached data key `IWD_isContactServer` is `1`
- The value of the attached data key `IWD_isDone` is either null or `0` (zero)

Flow Summary



Flow Detail

1. Entry to MarkInteractionAsDone workflow.
2. The InvokeUCS subroutine is invoked to complete interaction in the UCS database.
3. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine
4. Exit MarkInteractionAsDone workflow.

Removal Strategy

The purpose of this strategy is to delete expired interactions from the Interaction Server database.

A key-value pair in user data with the key `IWD_expirationDateTime` contains information about when an interaction has to be deleted.

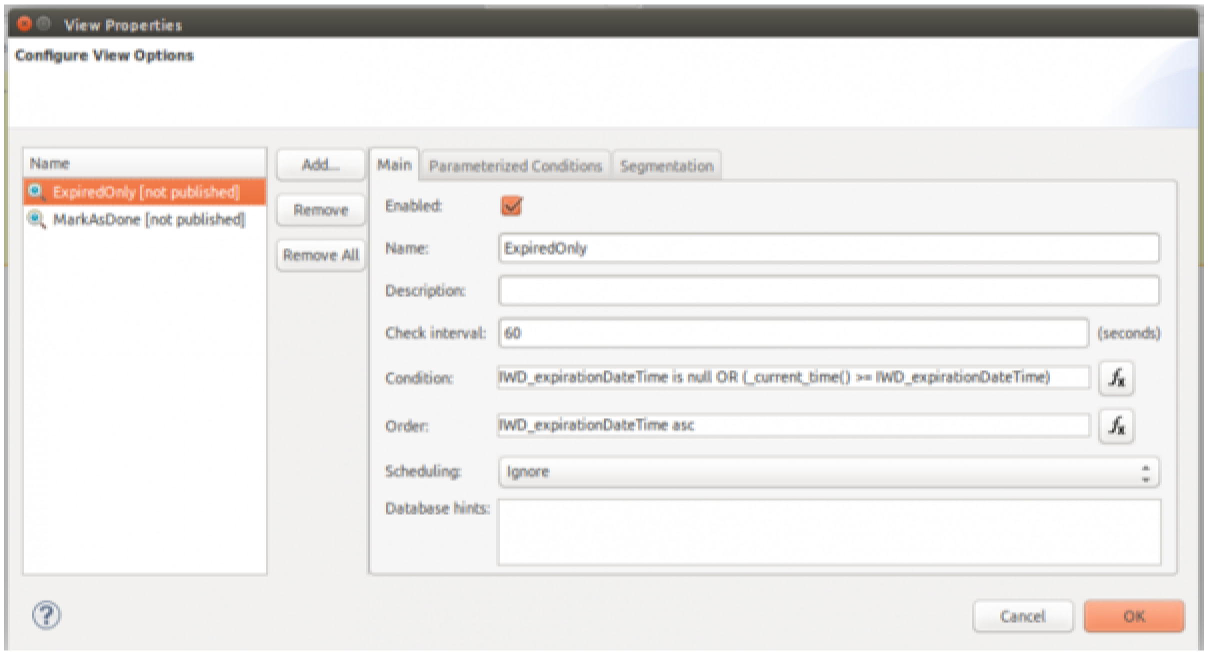
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Rejected`

Interactions have to satisfy the following conditions:

- Interactions must either have the property `IWD_expirationDateTime` not set, or this property must have a time stamp which is in the past.
- Interactions are taken in the order they were submitted.

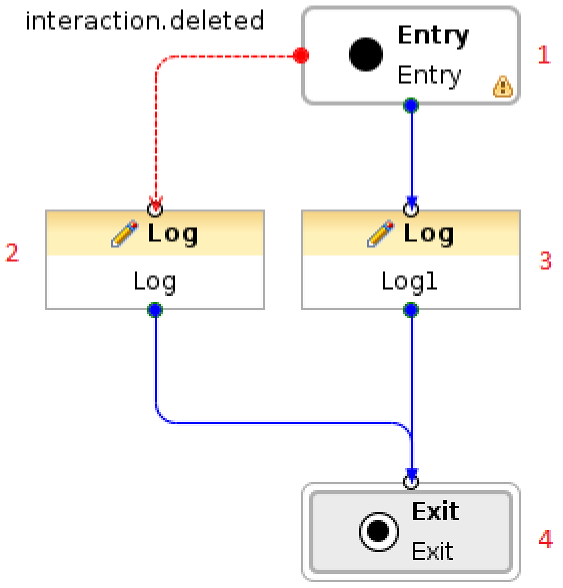
Composer Configuration



Name
ExpiredOnly [not published]
MarkAsDone [not published]

Enabled:	✓
Name:	ExpiredOnly
Description:	
Check interval:	60 (seconds)
Condition:	IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)
Order:	IWD_expirationDateTime asc
Scheduling:	Ignore
Database hints:	

Flow Summary



Flow Detail

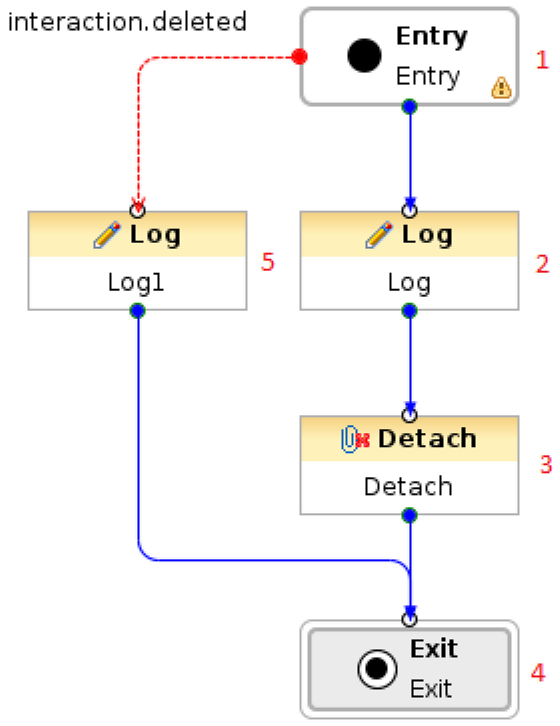
1. Entry to Removal workflow.
2. Log message in case if interaction was from some reasons deleted.
3. Log message: `Task will be terminated on exit`
4. Exit Removal workflow.

Finish Strategy

This workflow detaches interactions from the session. This workflow processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Errorheld`

Flow Summary



Flow Detail

1. Entry to Finish workflow.
2. Log message :'Task processing completed'.
3. Detach interaction. After this operation, the interaction will not be processed any more.
4. Log message in case if interaction was for some reason deleted.
5. Exit Finish workflow.

IWDBP Strategies & Subroutines prior to 9.0.004

Classification Strategy

The purpose of this strategy is to invoke corresponding classification rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

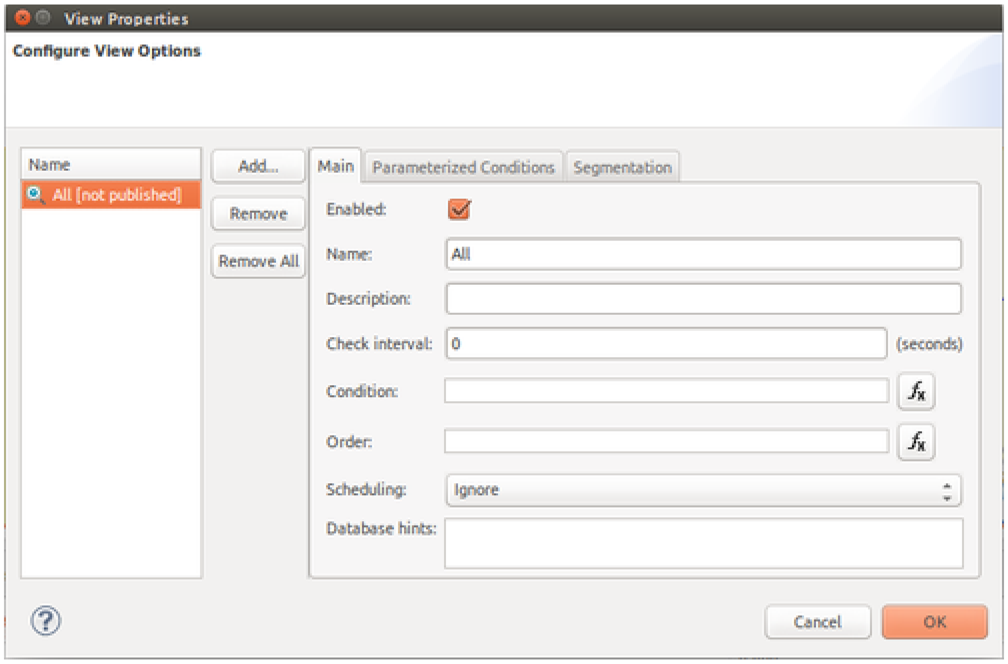
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_New`

Interactions have to satisfy the following conditions:

- There are no conditions here.
- Interactions are taken in order they were submitted.

Composer Configuration

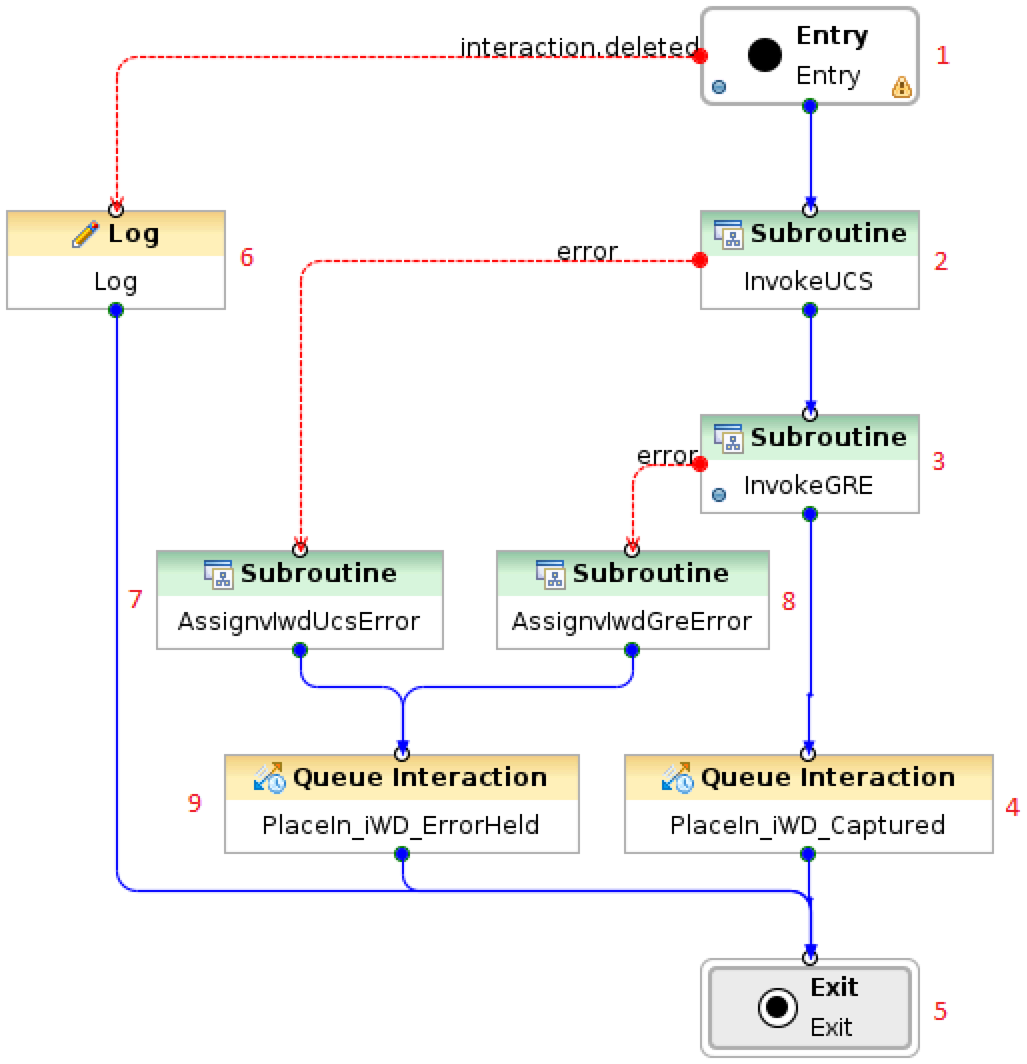


All [not published]

Enabled: ✓
Name: All
Description:
Check interval: 0 (seconds)
Condition:
Order:
Scheduling: Ignore
Database hints:

The Composer configuration for this strategy block shows that all interactions are distributed to the `iwd_bp_comp.Main.iWD_New` queue without conditions.

Flow Summary



Flow Detail

1. Entry to Classification workflow.
2. The InvokeUCS subroutine is invoked to create new interaction in the UCS database.
3. The InvokeGRE subroutine is invoked.
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
5. Exit Classification workflow.
6. Log message in case if interaction was from some reasons deleted.

7. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`.
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine.
8. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error description that occured in InvokeGRE subroutine.
9. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

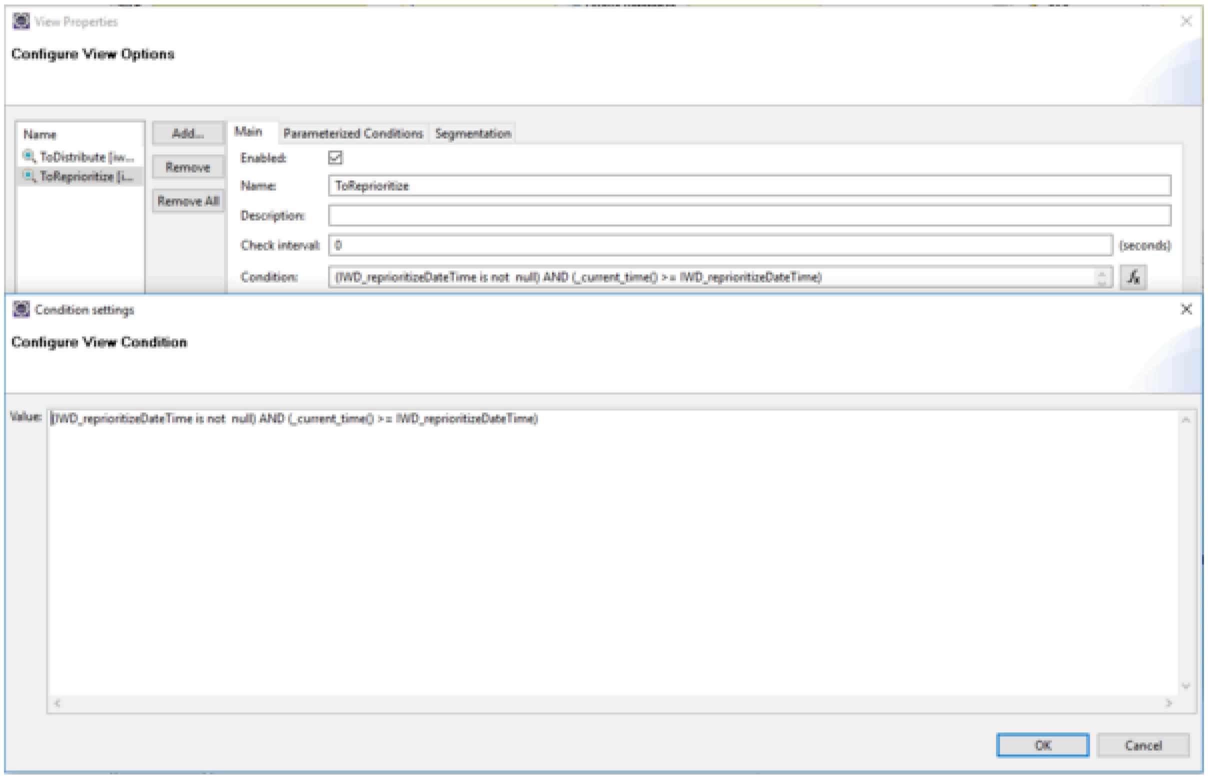
Prioritization Strategy

The purpose of this strategy is to invoke the corresponding prioritization rules, analyze the result of the rules application and place the interaction into the appropriate queue, depending on the result.

This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Captured`—Interactions have to satisfy the following conditions:
    - Active interactions only (interactions which do not have the property `IWD_activationDateTime` set, or this property has a time stamp which is in the past.
    - Interactions are taken in the order they were submitted.
- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are subject for immediate reprioritization (interactions that have the property `IWD_reprioritizeDateTime` set to a time stamp which is in the past).
    - Interactions are taken in order of `IWD_reprioritizationDateTime` (oldest first).

Composer Configuration


View Properties

Configure View Options

Name: ToDistribute [iw..., ToReprioritize [i...

Enabled: ✓

Name: ToReprioritize

Description:

Check interval: 0 (seconds)

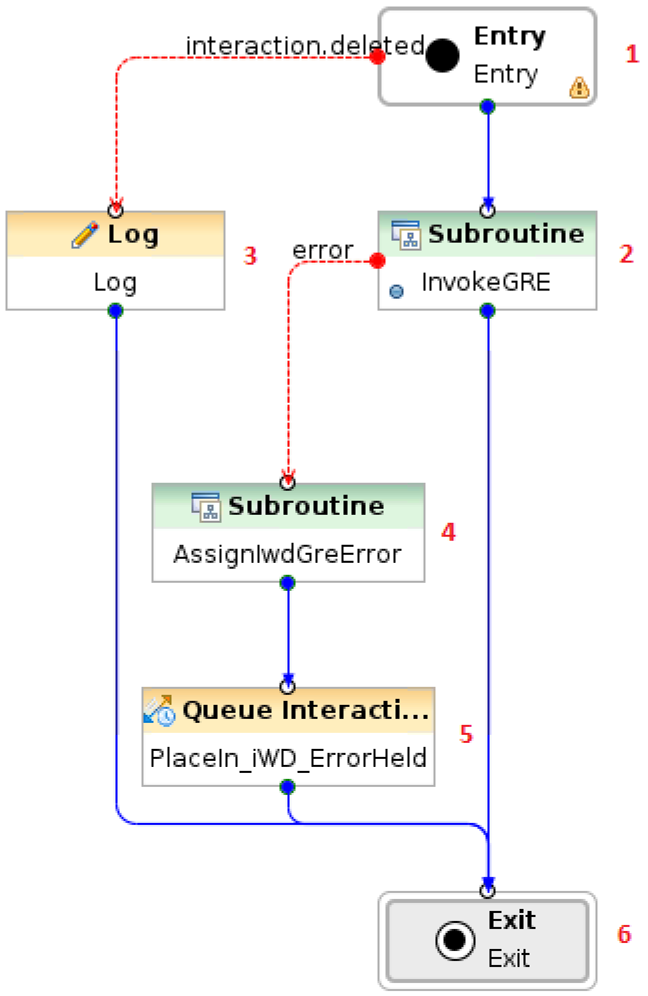
Condition: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Condition settings

Configure View Condition

Value: (IWD_reprioritizeDateTime is not null) AND (_current_time() >= IWD_reprioritizeDateTime)

Flow Summary



Flow Detail

1. Entry to Prioritization workflow.
2. The `InvokeGRE` subroutine is invoked.
3. Log message in case if interaction was from some reasons deleted.
4. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error description that occurred in `InvokeGRE` subroutine.
5. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.

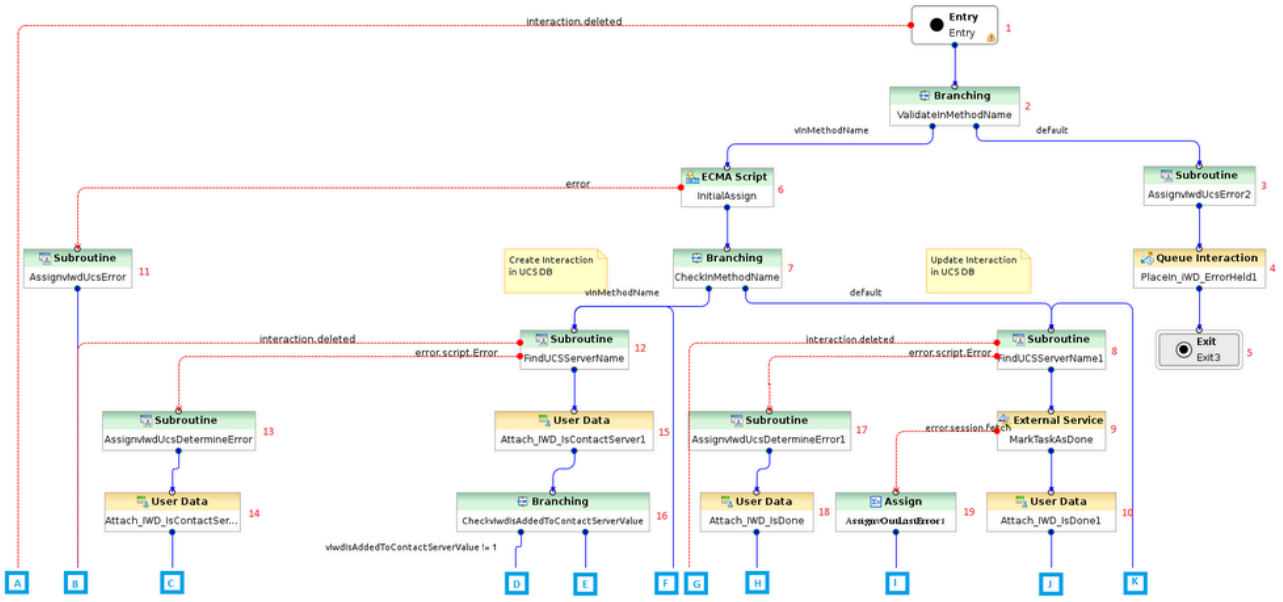
6. Exit Prioritization workflow.

Invoke UCS Strategy

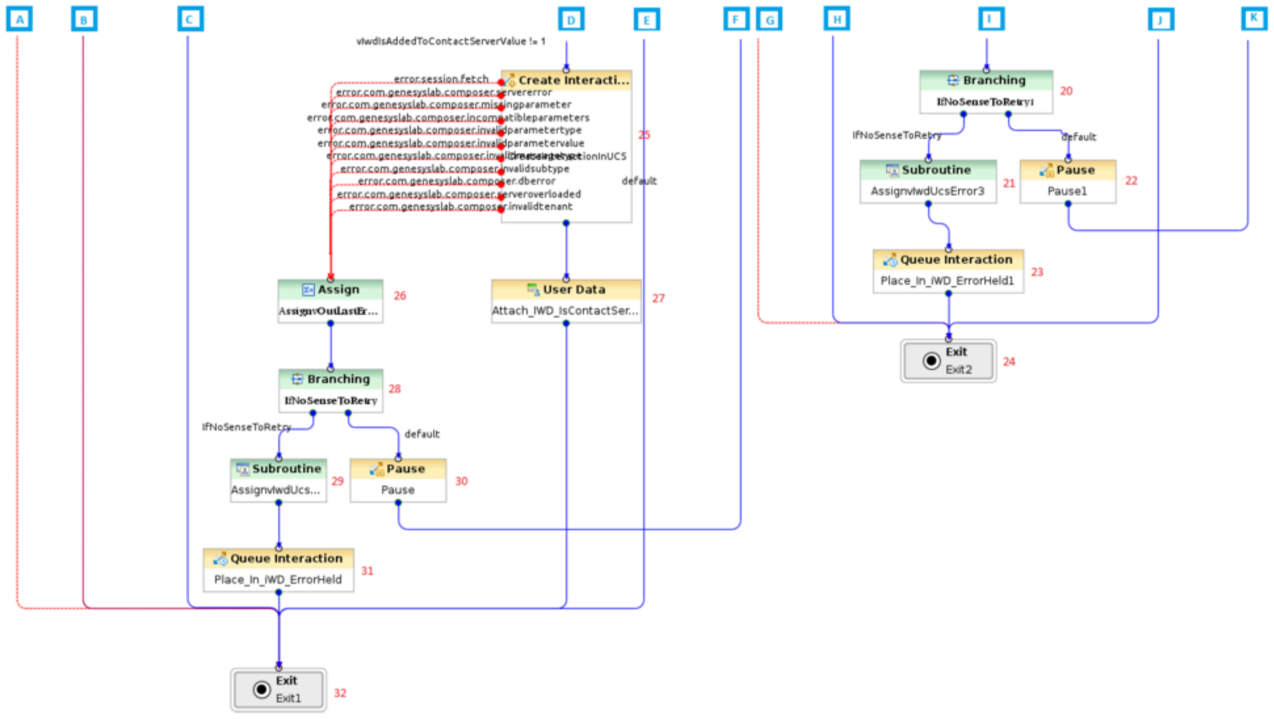
The purpose of this workflow is to create an interaction in the UCS database if UCS is configured.

Flow Summary

Part 1



Part 2


A
B
C
D
E
F
G
H
I
J
K
vIwdIsAddedToContactServerValue != 1
Create Interacti...
error.session.fetch
error.com.genesyslab.composer.servererror
error.com.genesyslab.composer.missingparameter
error.com.genesyslab.composer.incompatibleparameters
error.com.genesyslab.composer.invalidparametertype
error.com.genesyslab.composer.invalidparametervalue
error.com.genesyslab.composer.invalidsubtype
error.com.genesyslab.composer.dberror
error.com.genesyslab.composer.serveroverloaded
error.com.genesyslab.composer.invalidtenant
default
25
Assign
AssignvOutLastEr...
26
User Data
Attach_IWD_IsContactSer...
27
Branching
IfNoSenseToRetry
28
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcs...
29
Pause
Pause
30
Queue Interaction
Place_In_iWD_ErrorHeld
31
Exit
Exit1
32
Branching
IfNoSenseToRetry1
20
IfNoSenseToRetry
default
Subroutine
AssignvIwdUcsError3
21
Pause
Pause1
22
Queue Interaction
Place_In_iWD_ErrorHeld1
23
Exit
Exit2
24

Flow Detail

1. Entry InvokeUCS strategy.
2. Check if `in_method_name = 'Create' | in_method_name = 'OMInteractions'`.
3. Invoke AssignLastError subroutine with attributes:
   - `vInLastErrorkey`—`IWD_UCS_Error`
   - `vInLastErrorString`—Error informs that: `vInMethodName + ' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeUCS workflow.
6. Variables are initialized:
   - `vExternalId`—Read from task attribute `ExternalId`
   - `vMediaType`—Read from task attribute
   - `vSubmittedBy`—Read from task attribute `attr_itx_submitted_by`
   - `vType`—Read from task attribute `'InteractionType'`
   - `vSubType`—Read from task attribute `'InteractionSubtype'`
   - `vIwdIsAddedToContactServerValue`—Read from task attribute `'IWD_isAddedToContactServer'`

7. Check if `in_method_name = 'Create'`.
8. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
   - `vInItemName—ContactServerList`
   - `vInListName—Iwd_Esp_List`
9. The strategy calls a method on the Universal Contact Server to set the status of the interaction to 3, indicating that the interaction is done.
10. The value of the user data key `IWD_isDone` is set to 1.
11. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`'
    - `vInLastErrorString`—Error description that occurred when variables were initialized
12. The FindListObjectItem subroutine is invoked to determine the name of the UCS Application. The subroutine uses the List Object list GREServerList:
    - `vInItemName—ContactServerList`
    - `vInListName—Iwd_Esp_List`
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`'
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
14. The value of the user data key `IWD_isContactServer` is set to `0`.
15. The value of the user data key `IWD_isContactServer` is set to 1.
16. Check if `IWD_isContactServer` is set to 1.
17. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in FindListObjectItem subroutine.
18. The value of the user data key `IWD_isDone` is set to `0`.
19. An error is extracted from user data and assigned in `vLastError` variable.
20. If it makes sense to retry updating the interaction record in UCS.
21. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry update interaction in UCS.
22. A delay is introduced into the processing. Flow returns to step 8.
23. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
24. Exit InvokeUCS workflow.
25. A new interaction is created in the UCS database, for this iWD task.
26. An error is extracted from user data and assigned in `vLastError` variable.

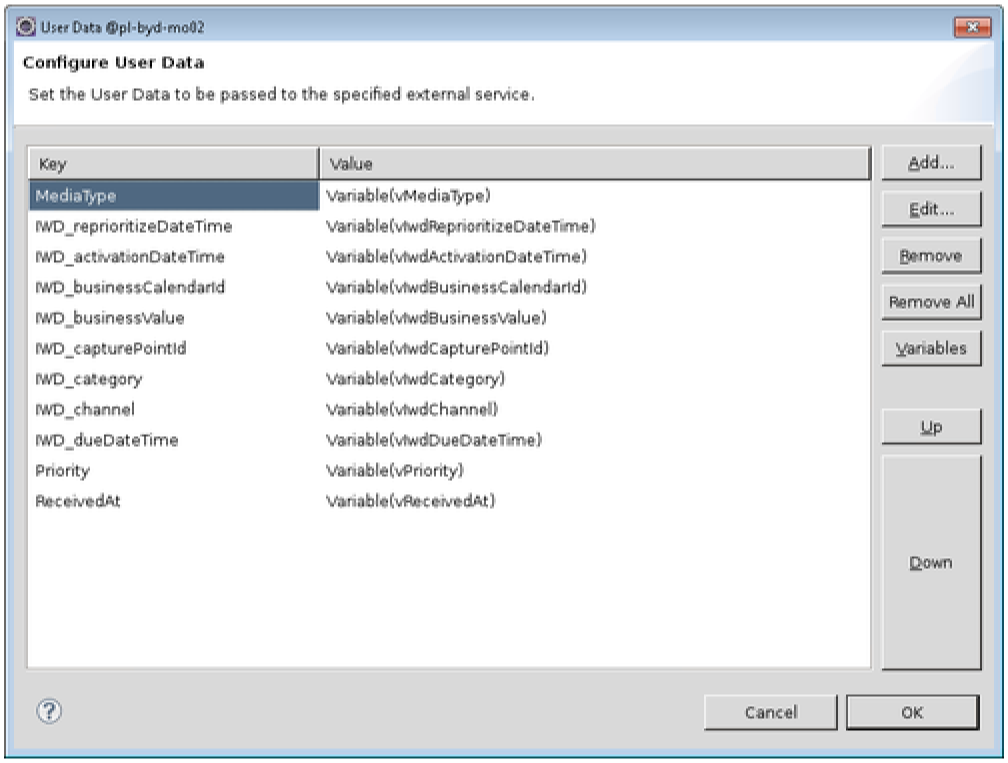
27. The user data key `IWD_isAddedToContactServer` is updated to 1 to indicate that the task was successfully added to the interaction history in UCS. The result returned from the ESP call to UCS (from See A new interaction is created in the UCS database, for this iWD task. If that function is successful is written to the variable `IWD_UCS_Result`.
28. If it makes sense to retry creating the interaction record in UCS.
29. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Information that it does not make sense to retry create interaction in UCS
30. A delay is introduced into the processing. Flow returns to step 12.
31. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
32. Exit InvokeUCS workflow.

InvokeGRE Strategy

Important

For Composer/ORS versions prior 8.1.400.48—If custom task attributes will be used in the Standard Rules Template, you must add them in the External Service block called `InvokeGRE` in the InvokeGRE workflow. All user-defined attributes need to be added in the User Data attribute, otherwise they will not be attached to the task and so will not be sent in the ESP request to the external ESP service.

Composer configuration


User Data @pl-byd-mo02

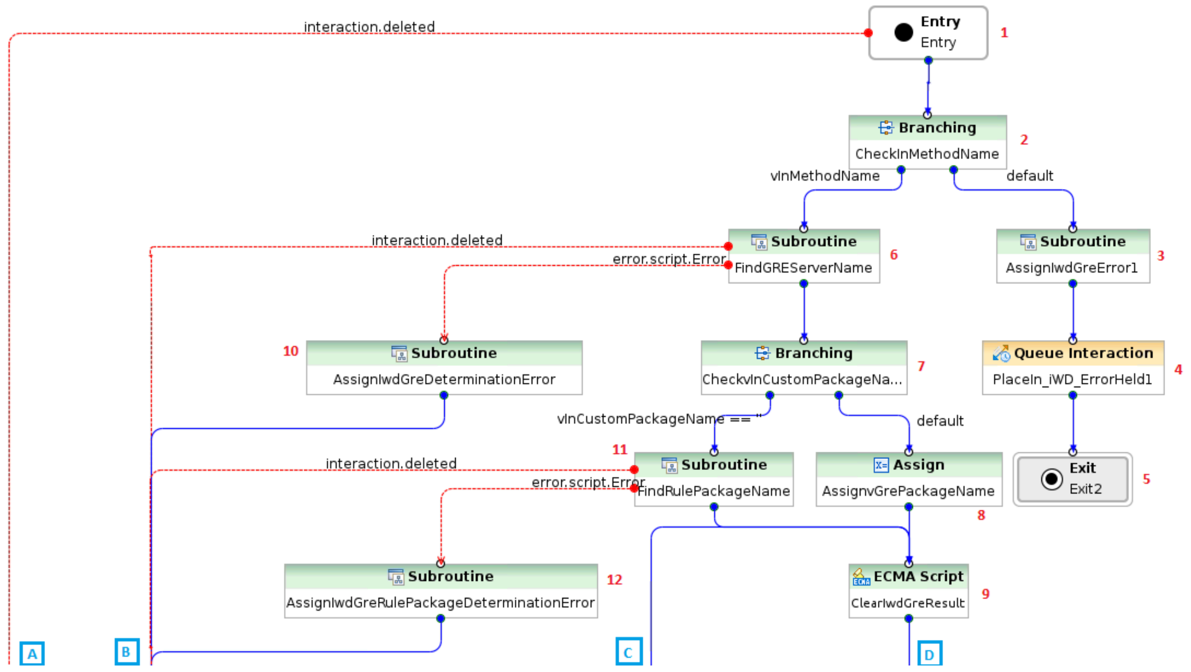
Configure User Data

Set the User Data to be passed to the specified external service.

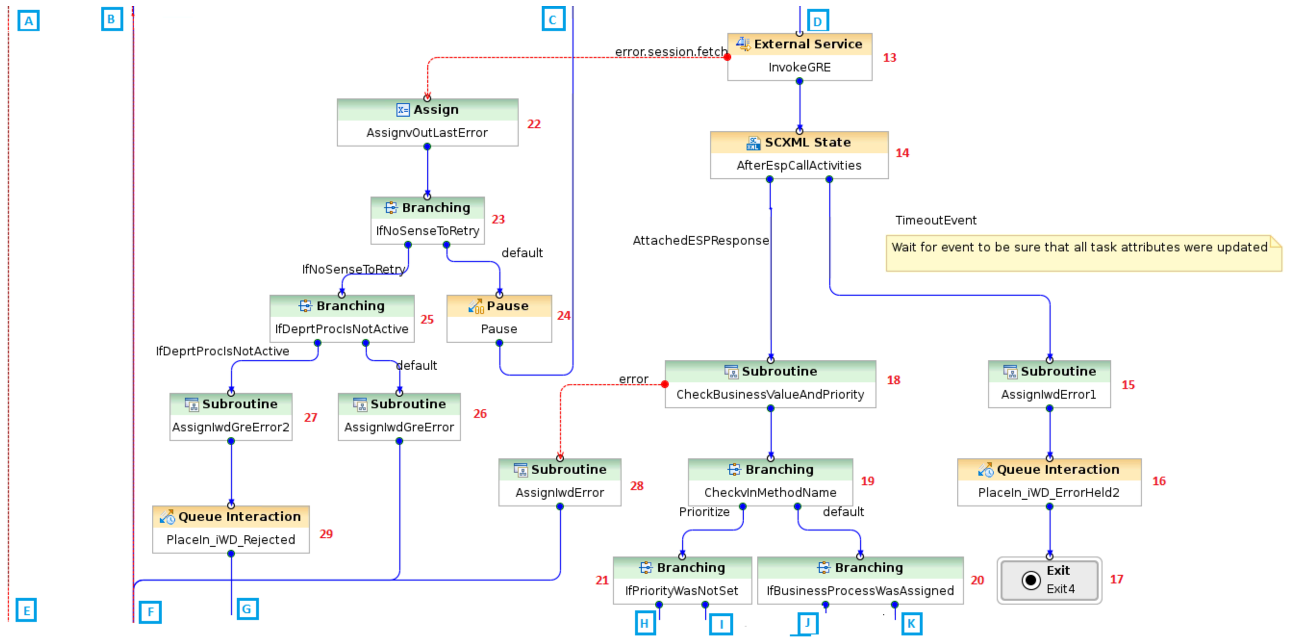
Key	Value
MediaType	Variable(vMediaType)
IWD_reprioritizeDateTime	Variable(vIwdReprioritizeDateTime)
IWD_activationDateTime	Variable(vIwdActivationDateTime)
IWD_businessCalendarId	Variable(vIwdBusinessCalendarId)
IWD_businessValue	Variable(vIwdBusinessValue)
IWD_capturePointId	Variable(vIwdCapturePointId)
IWD_category	Variable(vIwdCategory)
IWD_channel	Variable(vIwdChannel)
IWD_dueDateTime	Variable(vIwdDueDateTime)
Priority	Variable(vPriority)
ReceivedAt	Variable(vReceivedAt)

Flow Summary

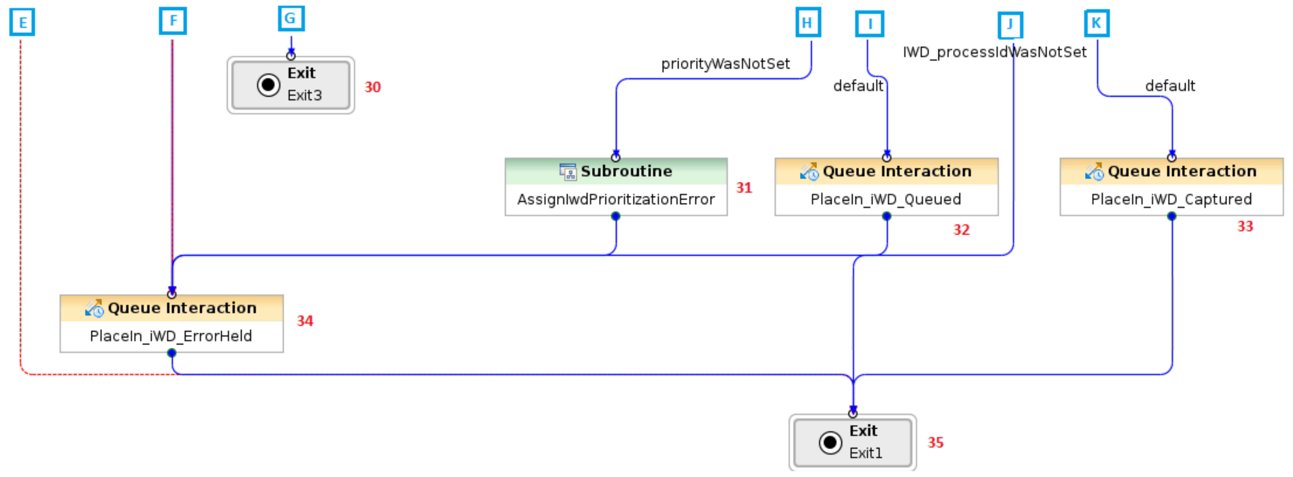
Part 1



Part 2



Part 3



Flow Detail

1. Entry to InvokeGRE strategy.
2. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
3. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: `vInMethodName` + `' is not valid'`
4. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
5. Exit InvokeGRE workflow.
6. The **FindListObjectItem** subroutine is invoked to determine the name of the Genesys Rules Engine Application. The subroutine uses the List Object list **GREServerList**:
    - `vInItemName`—`GREServerList`
    - `vInListName`—`Iwd_Esp_List`
7. Check if `vInCustomPackageName` was published to this subroutine. If it is set then `vInCustomPackageName` will be run. Otherwise package name needs to be found in `Iwd_Package_List`.
8. Assign `vInCustomPackageName` to `vGrePackageName`.
9. Delete `IWD_GRE_Result`, `IWD_Error`, `RulePhase` before Invoke GRE.
10. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
11. The `FindListObjectItem` subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules:
    - `vInItemName`—`RulePackageList`

- `vInListName`—`Iwd_Package_List`

12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Rule_Package_Determination_Error`
    - `vInLastErrorString`—Error description that occurred in `FindListObjectItem` subroutine.
13. An ESP request is sent to the Genesys Rules Engine to evaluate the classification rules.

> **Important**
> All user data that needs to be added to ESP request must be added in User Data attributes.

14. Parse ESP result and attach to the interaction all attributes modified by the GRE.
15. Invoke **AssignLastError** subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—Error informs that: '`Attach GreResult timeout`'
16. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
17. Exit InvokeGRE workflow.
18. CheckBusinessValueAndPriority subroutine is called to verify if `IWD_businessValue` and `Priority` have correct values.
19. Check if `in_method_name` is set to `SetBusinessContext` or `Prioritize`.
20. Check if `IWD_processId` was set by any rules or when task was created.
21. Check is made to see if this is the first time that prioritization rules are being evaluated for the interaction, and the priority was not set up by any rules.
22. Get last error that was occured in GRE call and assign it to `vLastError` variable.
23. A check is done to see if the error code is related to the ESP server communication.
24. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to step 11 to retry the connection to the ESP server.
25. The last Interaction Server-related error is extracted from a variable.
26. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
27. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable.
28. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_GRE_Error`
    - `vInLastErrorString`—The last Interaction Server-related error is extracted from a variable

29. The interaction is placed in the `iwd_bp_comp.Main.iWD_Rejected` queue.
30. Exit InvokeGRE workflow.
31. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Prioritization_Error`
    - `vInLastErrorString`—Error description: '`Priority is not set up by rules`'.
32. The interaction is placed in the `iwd_bp_comp.Main.iWD_Queued` queue.
33. The interaction is placed in the `iwd_bp_comp.Main.iWD_Captured` queue.
34. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
35. Exit InvokeGRE workflow.

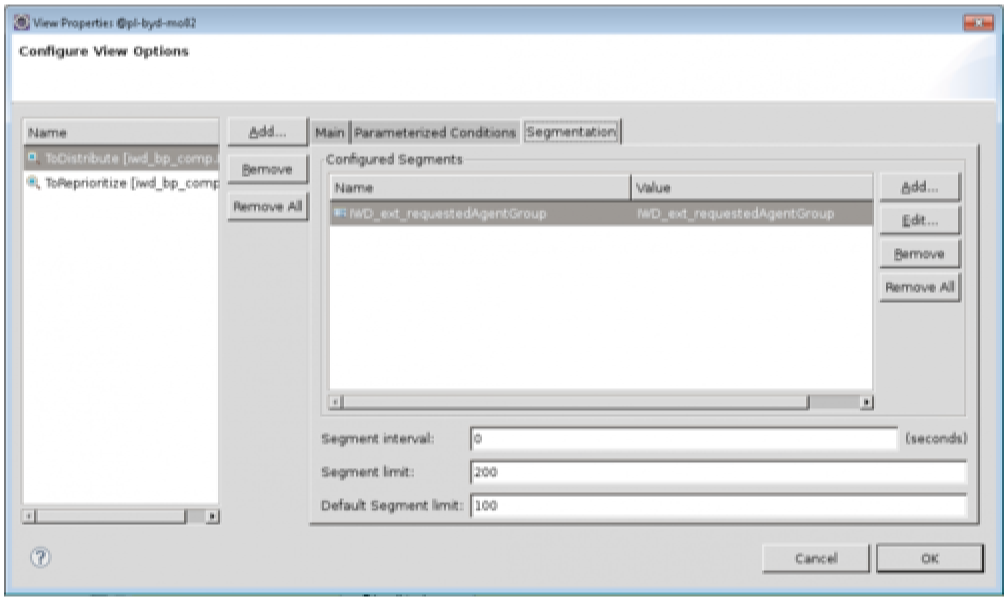
Distribution Strategy

This strategy routes interactions to a requested Agent, requested Agent Group, requested Skill, or to the default iWD Agent Group. This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Queued`—Interactions have to satisfy the following conditions:
    - Interactions that are not subject for immediate reprioritization (interactions that do not have the property `IWD_reprioritizeDateTime` set, or that have this property set to a time stamp that is in the future).
    - Interactions are taken in order of priority (highest priority first)

A Segmentation feature ensures that all agents can be kept busy by distributing tasks in each segment separately. As a result, even in a Distribution strategy that is populated by high-priority tasks assigned to small groups of agents, the strategy will not become so saturated that distribution of tasks to other agents is blocked. Segmentation settings are found in the **ToDistribute** view of the Distribution routing strategy. The Distribution strategy can make a call to the segmentation setting and add an `IWD_Segment` attribute to the interaction data.

Composer Configuration - Segmentation View



Name
ToDistribute [iwd_bp_comp.
ToReprioritize [iwd_bp_comp

Configured Segments

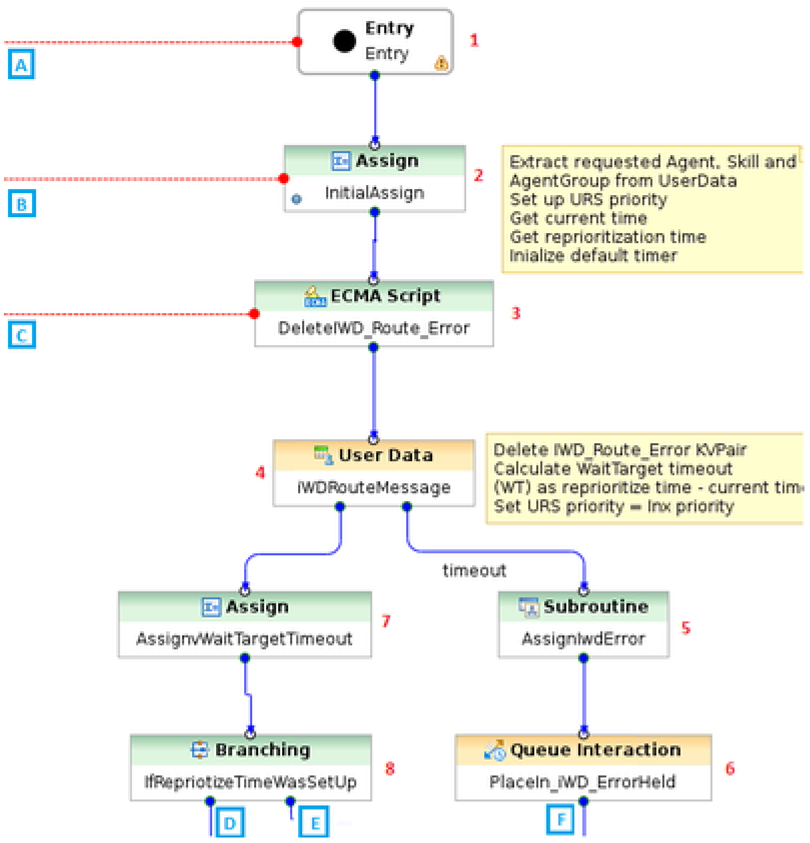
Name	Value
IWD_ext_requestedAgentGroup	IWD_ext_requestedAgentGroup

Segment interval: 0 (seconds)
Segment limit: 200
Default Segment limit: 100

Flow Summary

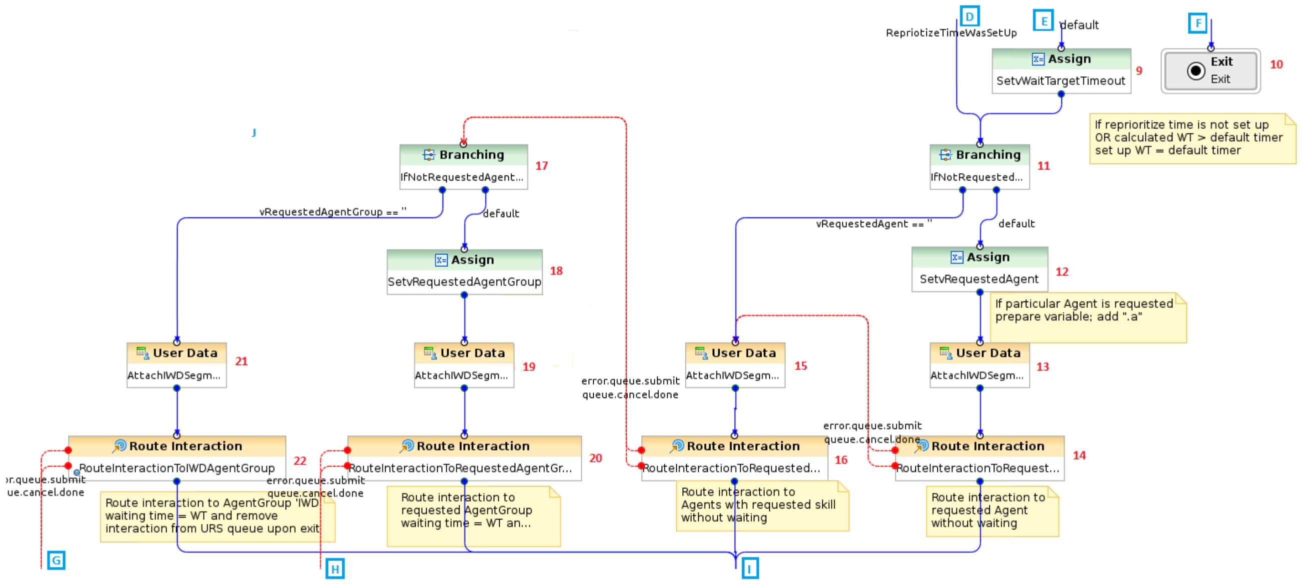
Part 1

Click to enlarge.



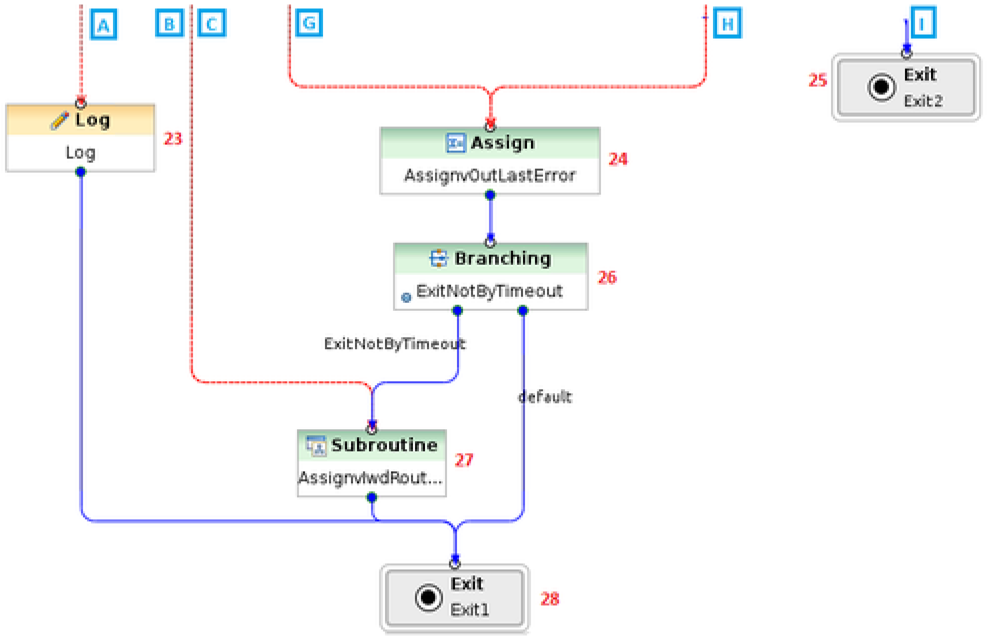
Part 2

Click to enlarge.



Part 3

Click to enlarge.



Flow Detail

1. Entry to Distribution workflow.
2. Variables are initialized:
   - vRequestedAgentGroup—Read from task attribute IWD_ext_requestedAgentGroup
   - vRequestedAgentGroup—Read from task attribute IWD_ext_requestedAgent
   - vRequestedSkill—Read from task attribute IWD_ext_requestedSkill
   - vCurrentTint—Current time in seconds
   - vReprioritizeDint—Read from task attribute IWD_businessValue
   - vDefaultTargetTimeout—Default target timeout set to 3600 seconds
   - vInxPriority—Read from task attribute Priority
3. Delete IWD_Route_Error from attached data. Calculate WaitTarget timeout based on vReprioritizeDTInt and vCurrentDTInt. Sets URS priority.
4. Set information about clear IWD_Route_Error attribute.
5. Invoke AssignLastError subroutine with attributes:
   - vInLastErrorkey—IWD_Error
   - vInLastErrorString—Error description: 'Update IWD_Route_Error timeout'
6. The interaction is placed in the iwd_bp_comp.Main.iWD_ErrorHeld queue.

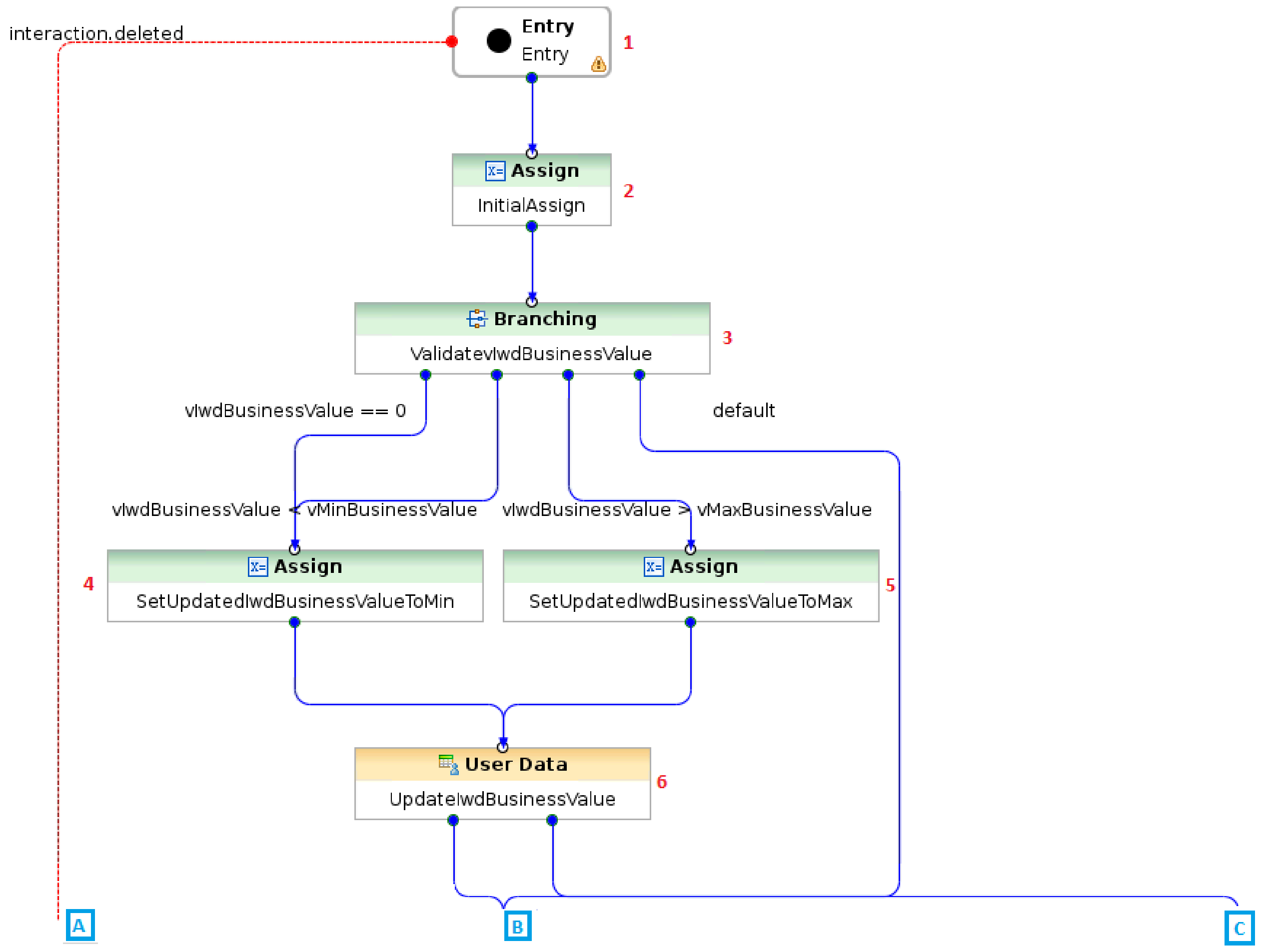
7. Calculate `vWaitTargetTimeout`.
8. Check if calculated `vWaitTargetTimeout` is in range `(0, vDefaultTargetTimeout>`.
9. Set `vWaitTargetTimeout` to `vDefaultTargetTimeout`.
10. Exit Distribution workflow.
11. Check if particular Agent is requested.
12. Assign `vRequestedAgent` + '`.a`' to `vRequestedAgent` variable.
13. Set `vIWDSegment` to '`_requested_agent`'.
14. Route interaction to requested `vRequestedAgent` without waiting.
15. Set `vIWDSegment` to '`_requested_skill`'.
16. Route interaction to requested `vRequestedAgent` with requested skill without waiting.
17. Check if particular `AgentGroup` is requested.
18. Assign `vRequestedAgentGroup` + '.qa' to `vRequestedAgentGroup` variable.
19. Set `vIWDSegment` to '`_requested_agent_group`'.
20. Route interaction to requested `vRequestedAgentGroup` with `vWaitTargetTimeout`.
21. Set `vIWDSegment` to '`default`'.
22. Route interaction to IWD Agent Group with `vWaitTargetTimeout`.
23. Log message in case if interaction was from some reasons deleted.
24. Assign last route interaction error to `vLastError`.
25. Exit Distribution workflow.
26. Check if route interaction finished with an error.
27. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey`—`IWD_Route_Error`
    - `vInLastErrorString`—Error description that occurred in route interaction
28. Exit Distribution workflow.

CheckBusinessValueandPriority Subroutine

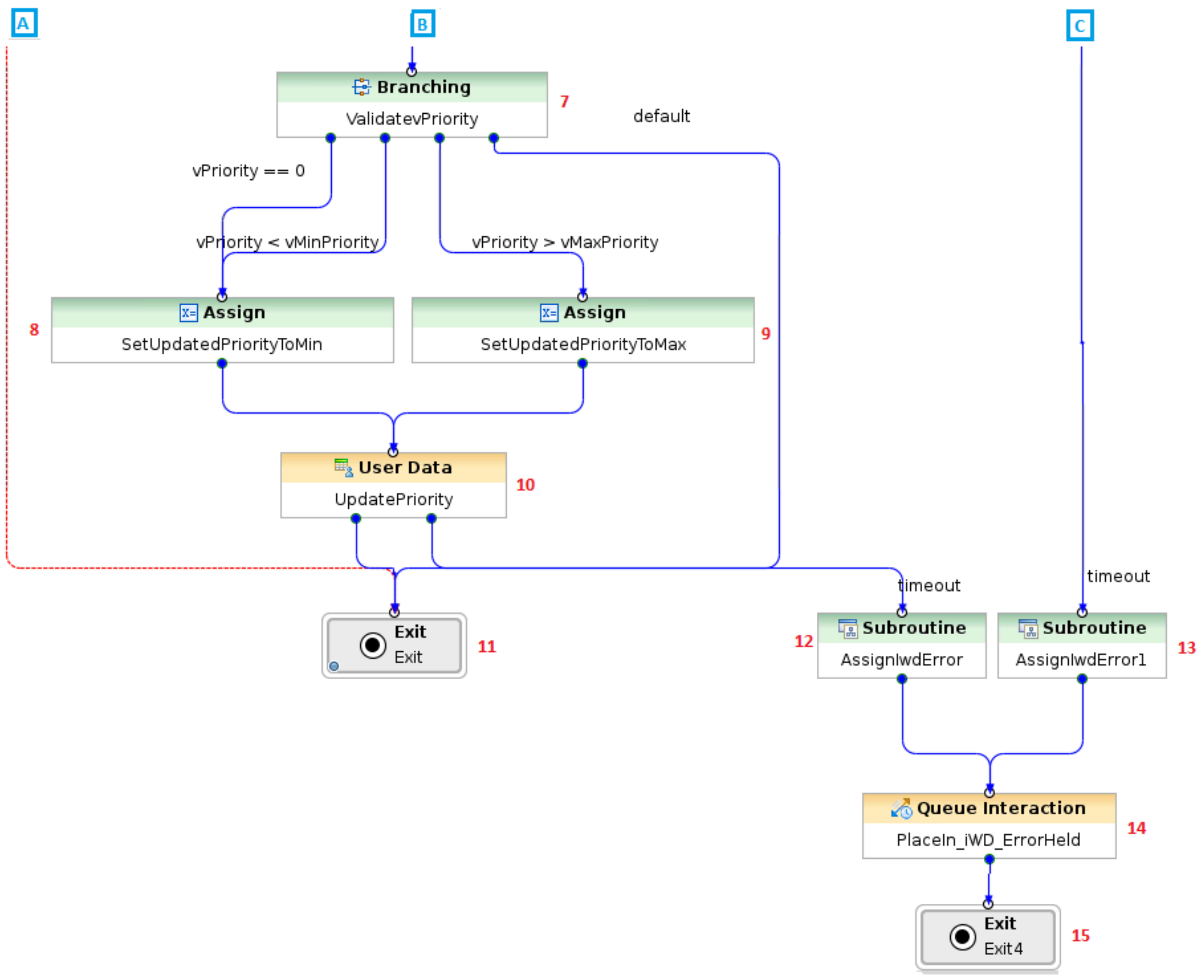
The purpose of this workflow is to verify if `Priority` and `IWD_businessValue` have correct values.

Flow Summary

Part 1


interaction.deleted
Entry
Entry
1
Assign
InitialAssign
2
Branching
ValidatevIwdBusinessValue
3
vIwdBusinessValue == 0
default
vIwdBusinessValue < vMinBusinessValue
vIwdBusinessValue > vMaxBusinessValue
4
Assign
SetUpdatedIwdBusinessValueToMin
Assign
SetUpdatedIwdBusinessValueToMax
5
User Data
UpdateIwdBusinessValue
6
A
B
C

Part 1



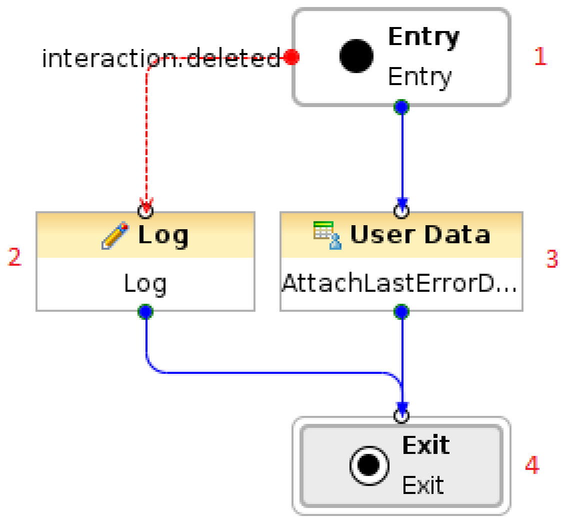
Flow Detail

1. Entry to CheckBusinessValueAndPriority workflow.
2. Variables are initialized:
   - `vIwdBusinessValue`—Read from task attribute `IWD_businessValue`
   - `vIwdPriority`—Read from task attribute `Priority`
3. Validate if `vIwdBusinessValue` is valid.
4. Set `vIwdBusinessValue` to `vMinBusinessValue`.
5. Set `vIwdBusinessValue` to `vMaxBusinessValue`.
6. Update `IWD_businessValue` to `vIwdBusinessValue`.

7. Validate if `vIwdPriority` is valid.
8. Set `vIwdPriority` to `vMinPriority`.
9. Set `vIwdPriority` to `vMaxPriority`.
10. Update Priority to `vIwdPriority`.
11. Exit `CheckBusinessValueAndPriority` workflow.
12. Invoke `AssignLastError` subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString` - Error description: '`Update Priority timeout`'
13. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey—IWD_Error`
    - `vInLastErrorString`— Error description: '`Update iWD_businessValue timeout`'
14. The interaction is placed in the `iwd_bp_comp.Main.iWD_ErrorHeld` queue.
15. Exit `CheckBusinessValueAndPriority` workflow.

AssignLastError Subroutine

Flow Summary



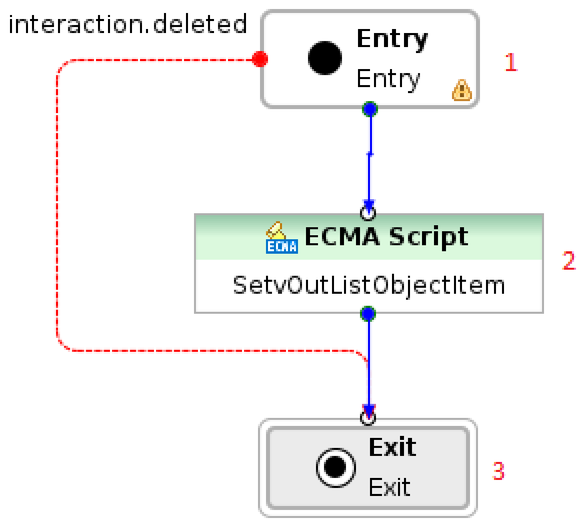
Flow Detail

1. Entry to AssignLastError workflow.
2. The last error is attached to user data as a key-value pair with the key `vInLastErrorkey` and value `vInLastErrorString`.

   `vInLastErrorkey` and `vInLastErrorString` are workflow attributes that need to be set before calling this workflow.
3. Log message if interaction was from some reasons deleted.
4. Exit AssignLastError workflow.

FindListObjectItem Subroutine

Flow Summary



Flow Detail

1. Entry to FindListObjectItem workflow.
2. Search `vKeyToFindInListObject` in `vInListName`.
   - `vInItemName`—Section in `vInListName`
   - `vInListName`—List object where `vKeyToFindInListObject` should be searched

- `vKeyToFindInListObject`—Option in `vInItemName` that should be found

3. When `vKeyToFindInListObject` is found in `vInListName`, then the value assigned to this option will be assigned to `vOutListObjectItem`.
4. Exit FindListObjectItem workflow.

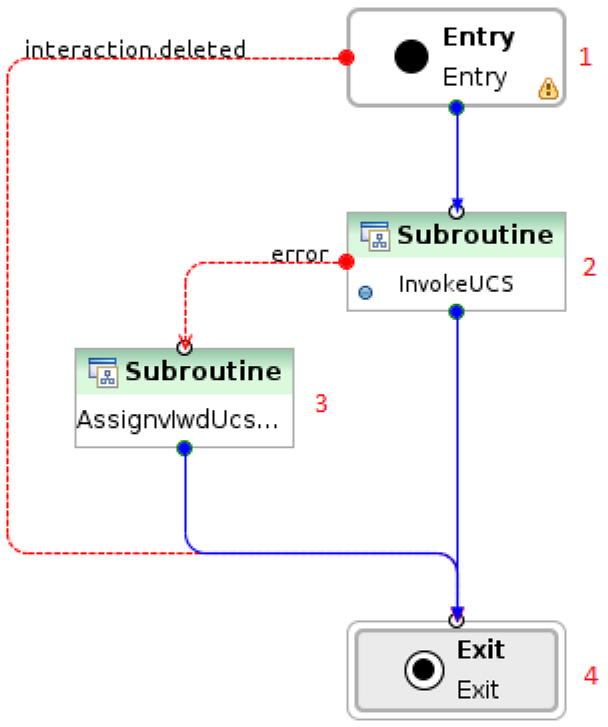
MarkInteractionAsDone Strategy

The purpose of this strategy is to update the Universal Contact Server (UCS) database to mark the interaction as done. This equates to setting the value in the Status column of the Interactions table to 3. UCS clients, such as Interaction Workspace, will then display the status of this interaction as done when the user looks at interactions they have previously processed.

Interactions have to satisfy the following conditions:

- The value of the attached data key `IWD_isContactServer` is `1`
- The value of the attached data key `IWD_isDone` is either null or `0` (zero)

Flow Summary



Flow Detail

1. Entry to MarkInteractionAsDone workflow.
2. The InvokeUCS subroutine is invoked to complete interaction in the UCS database.
3. Invoke AssignLastError subroutine with attributes:
    - `vInLastErrorkey`—`IWD_UCS_Error`
    - `vInLastErrorString`—Error description that occurred in InvokeUCS subroutine
4. Exit MarkInteractionAsDone workflow.

Removal Strategy

The purpose of this strategy is to delete expired interactions from the Interaction Server database.

A key-value pair in user data with the key `IWD_expirationDateTime` contains information about when an interaction has to be deleted.

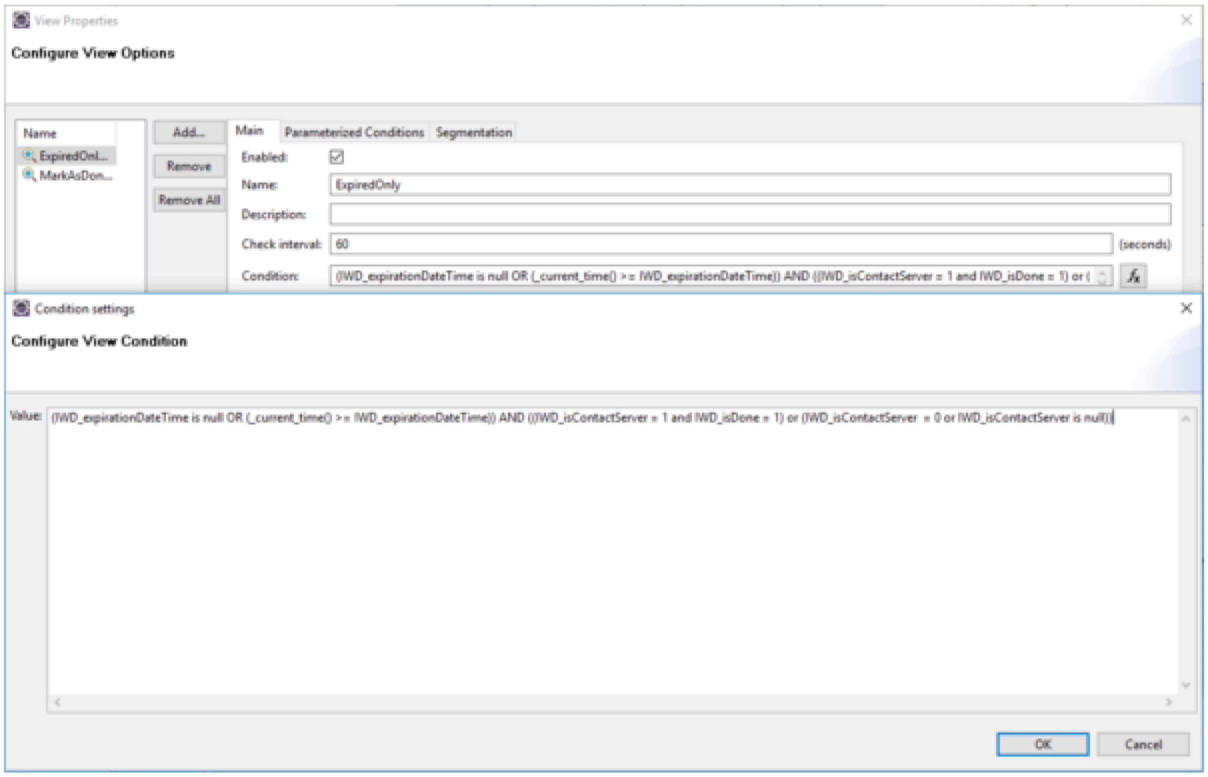
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_Rejected`

Interactions have to satisfy the following conditions:

- Interactions must either have the property `IWD_expirationDateTime` not set, or this property must have a time stamp which is in the past.
- If UCS is available, interactions must be marked as done in UCS.
- Interactions are taken in the order they were submitted.

Composer Configuration



View Properties

Configure View Options

Name: ExpiredOnl..., MarkAsDon...

Enabled: ✓

Name: ExpiredOnly

Description:

Check interval: 60 (seconds)

Condition: (IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (

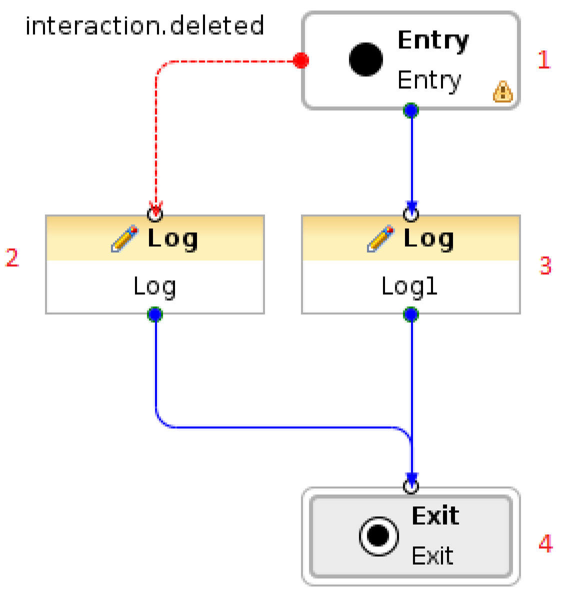
Condition settings

Configure View Condition

Value:

```
(IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (IWD_isContactServer  = 0 or IWD_isContactServer is null))
```

Flow Summary



Flow Detail

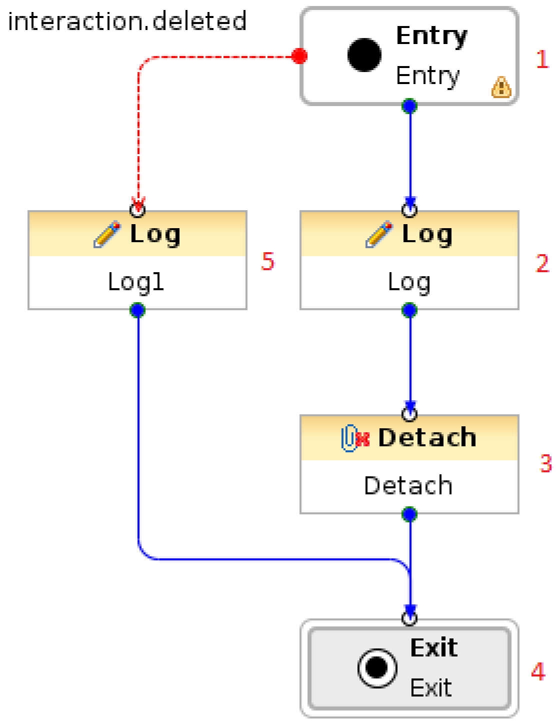
1. Entry to Removal workflow.
2. Log message in case if interaction was from some reasons deleted.
3. Log message: `Task will be terminated on exit`
4. Exit Removal workflow.

Finish Strategy

This workflow detaches interactions from the session. This workflow processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_Errorheld`

Flow Summary



Flow Detail

1. Entry to Finish workflow.
2. Log message :'Task processing completed'.
3. Detach interaction. After this operation, the interaction will not be processed any more.
4. Log message in case if interaction was for some reason deleted.
5. Exit Finish workflow.

Modifying the iWDBP

For most environments, the only modification that will need to be made to the iWD Business Process (`iwd_bp_comp`) is to the `Distribution` strategy. The recommended approach to doing this is:

1. Add a new strategy into the iWD Business Process.
2. Replace the connection from `iwd_bp_comp.Main.iWD_Queued/All` view to the `Distribution` routing strategy with a connection from `iwd_bp_comp.Main.iWD_Queued` to your own routing strategy where distribution logic is described.
3. Link your new distribution strategy to the out-of-the-box `iwd_bp_comp.Main.iWD_Completed` queue.

By modifying the business process in this way, rather than simply updating the provided Distribution strategy, you can easily import any new versions of the iWD Business Process that might be available in the future (the links will have to be re-established to your own distribution strategy).

You can also add additional interaction queues into the IWDBP business process, based on your business requirements. However, keep the following points in mind:

- The `iwd_bp_comp.Main.iWD_Queued` queue (or its equivalent defined in the Solution configuration) must be present for Data Mart to properly count interactions/tasks. You can add other queues to the business process, but only after interactions have passed through the `iwd_bp_comp.Main.iWD_Queued` queue.
- Data Mart can properly determine when to consider a task as complete, only if the final queue in the business process is one of the following:
    - `iwd_bp_comp.Main.iWD_Rejected`
    - `iwd_bp_comp.Main.iWD_Canceled`
    - `iwd_bp_comp.Main.iWD_Completed`

or their equivalents defined in the Solution configuration.

Cloning the Composer iWDBP to Create New Business Processes

To clone the IWDBP in Composer (`iwd_bp_comp`) in this way, you must have the Genesys Deployment Agent (GDA) running.

Cloning the iWD Business Process Using Configuration Manager

iWD allows you to create more than one iWD business process (or complete interaction workflows) in one tenant. For example, interactions from different media types can be handled by separate business processes.

Procedure

1. Copy the `iwd_bp_comp` delivered with the iWD Manager installation package (for example, `iwd_bp_comp -> iwd_bp_comp1`).
2. Change the project name in the `.project` file (for example, <name>iwd_bp_comp</name> `->` <name>iwd_bp_comp1</name>).
3. Change the project name in the `.composer` file (for example, `project-name="iwd_bp_comp" -> project-name="iwd_bp_comp1"`).
4. Import the `iwd_bp_comp` copy to an instance of Eclipse with Composer already installed.
5. To create configuration objects, open the copy of `iwd_bp_comp` created in step 1, expand **Interaction Processes** and select `Main.ixnprocess`.
6. Right-click and then select **Publish to Configuration Server**.
7. Select the `iwd_bp_comp` copy project, right-click and select **Generate All**.
8. Check the **Deploy Project** and **Publish Data to Configuration Server** boxes, then click **Finish**.
9. Navigate to **iWD GAX Plugin -> <*used tenant*> -> <*used solution*> -> Business Structure**.
10. Change the queue names to the newly created queues.

Additional Configuration

Interaction Queues

iWD recognizes seven interaction queues. By default they are created by the delivered iWD business process (iwd_bp_comp) and have the following names:

- `iwd_bp_comp.Main.iWD_New`
- `iwd_bp_comp.Main.iWD_Captured`
- `iwd_bp_comp.Main.iWD_Queued`
- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Rejected`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_ErrorHeld`

If there is more than one business process, customized queues must be configured for each solution in the iWD GAX Plug-in. The set of allowed queues is taken from all defined business processes. The names of the chosen queues will then be used by both iWD Manager and iWD Data Mart instead of the default ones.

Adding Custom Queue Names to Interaction Server

You must also ensure that the names of all customized queues for completed tasks are added to the list of queue names in Interaction Server in the **completed-queues** option.

Filters

Pre-defined filters on the Global Task List have explicit queue names in their conditions. When custom queues are defined, it is necessary to update filters' criteria with generic queue names instead of explicit ones. For example, the filter criterion `Queue is iwd_bp_comp.Main.iWD_Completed` should be changed to `Queue is Completed`. After such a change the filter will work correctly in all solutions with defined custom queues for completed tasks.

The following filter criteria support generic queue names:

- `Queue is '{queue}',`
- `Queue is not '{queue}'.`

When you choose one of these criteria in the **Filters** page of iWD Manager, a drop-down list appears in place of `'{queue}'`. There are seven generic queue names available on the list:

- `iwd_bp_comp.Main.iWD_New`
- `iwd_bp_comp.Main.iWD_Captured`
- `iwd_bp_comp.Main.iWD_Queued`
- `iwd_bp_comp.Main.iWD_Completed`
- `iwd_bp_comp.Main.iWD_Rejected`
- `iwd_bp_comp.Main.iWD_Canceled`
- `iwd_bp_comp.Main.iWD_ErrorHeld`

and a special value, `"(Custom...)"`. When "(Custom...)" is selected, an edit box appears that allows you to write an explicit queue name.

Integrated Capture Points

Integrated Capture Points' options must be set accordingly so that they can put new or modified interactions in the correct interaction queues. When an integrated Capture Point is connected with an iWD solution, its options are automatically synchronized with the solution. The following options are updated in Capture Points to work with a customized iWD business process:

JMS Capture Point and File Capture Point

- `inbound-transformer-parameters`
  - `CancelQueues`
  - `CompleteQueues`
  - `RestartQueues`
- `outbound-transformer-parameters`
  - `CancelQueues`
  - `CompleteQueues`
  - `ErrorHeldQueues`
  - `RejectQueues`
  - `RestartQueues`

Web Service Capture Point and Database Capture Point

- `iwd-parameters`
  - `CancelQueues`
  - `CompleteQueues`
  - `ErrorHeldQueues`
  - `RejectQueues`
  - `RestartQueues`

All Capture Points

- `default-values`
  - `Queue`

The following mapping between configured queues and Capture Points' options is maintained.

Capture Point Option	iWD Solution's Queue	Default Value
default-values/Queue	New	`iwd_bp_comp.Main.iWD_New`
RestartQueues	New	`iwd_bp_comp.Main.iWD_New`
CompleteQueues	Completed	`iwd_bp_comp.Main.iWD_Completed`

Capture Point Option	iWD Solution's Queue	Default Value
RejectQueues	Rejected	`iwd_bp_comp.Main.iWD_Rejected`
CancelQueues	Canceled	`iwd_bp_comp.Main.iWD_Canceled`
ErrorHeldQueues	Error Held	`iwd_bp_comp.Main.iWD_ErrorHeld`

The options are updated whenever a user changes any of the queues in the iWD Solution configuration in GAX. They are also modified when a user changes the assigned Solution in the Capture Point's configuration in GAX. If no Solution has been assigned to the Capture Point, the queue options can be set manually.