



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Working with the iWD Business Process in Composer

Contents

- 1 IWDBP Strategies & Subroutines prior to 9.0.004
 - 1.1 Classification Strategy
 - 1.2 Prioritization Strategy
 - 1.3 Invoke UCS Strategy
 - 1.4 InvokeGRE Strategy
 - 1.5 Distribution Strategy
 - 1.6 CheckBusinessValueandPriority Subroutine
 - 1.7 AssignLastError Subroutine
 - 1.8 FindListObjectItem Subroutine
 - 1.9 MarkInteractionAsDone Strategy
 - 1.10 Removal Strategy
 - 1.11 Finish Strategy

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.

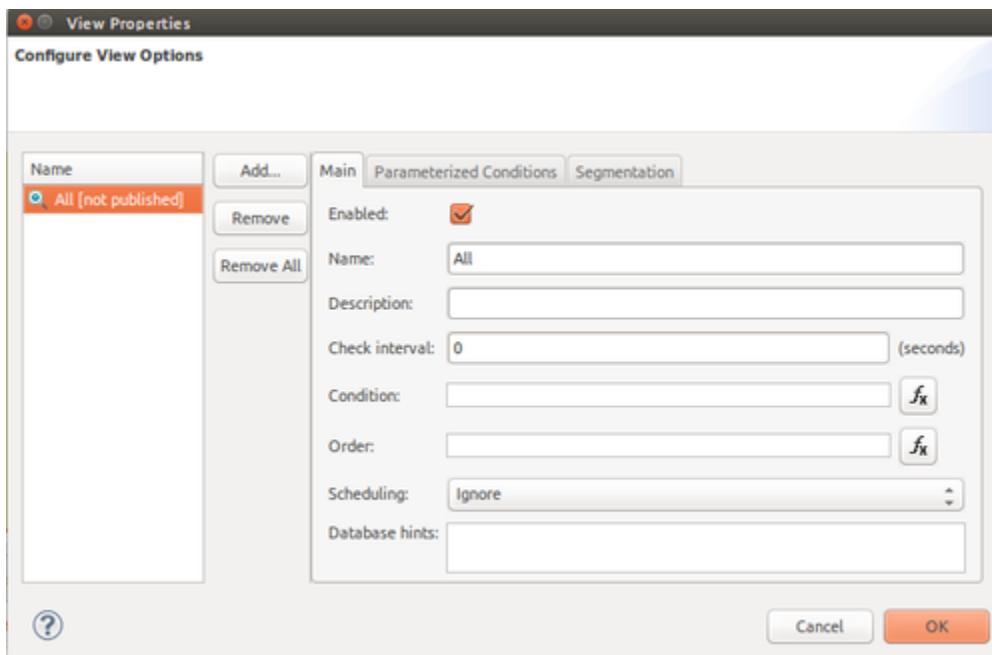
This strategy processes interactions from the following queues:

- `iwd_bp_comp.Main.iWD_New`

Interactions have to satisfy the following conditions:

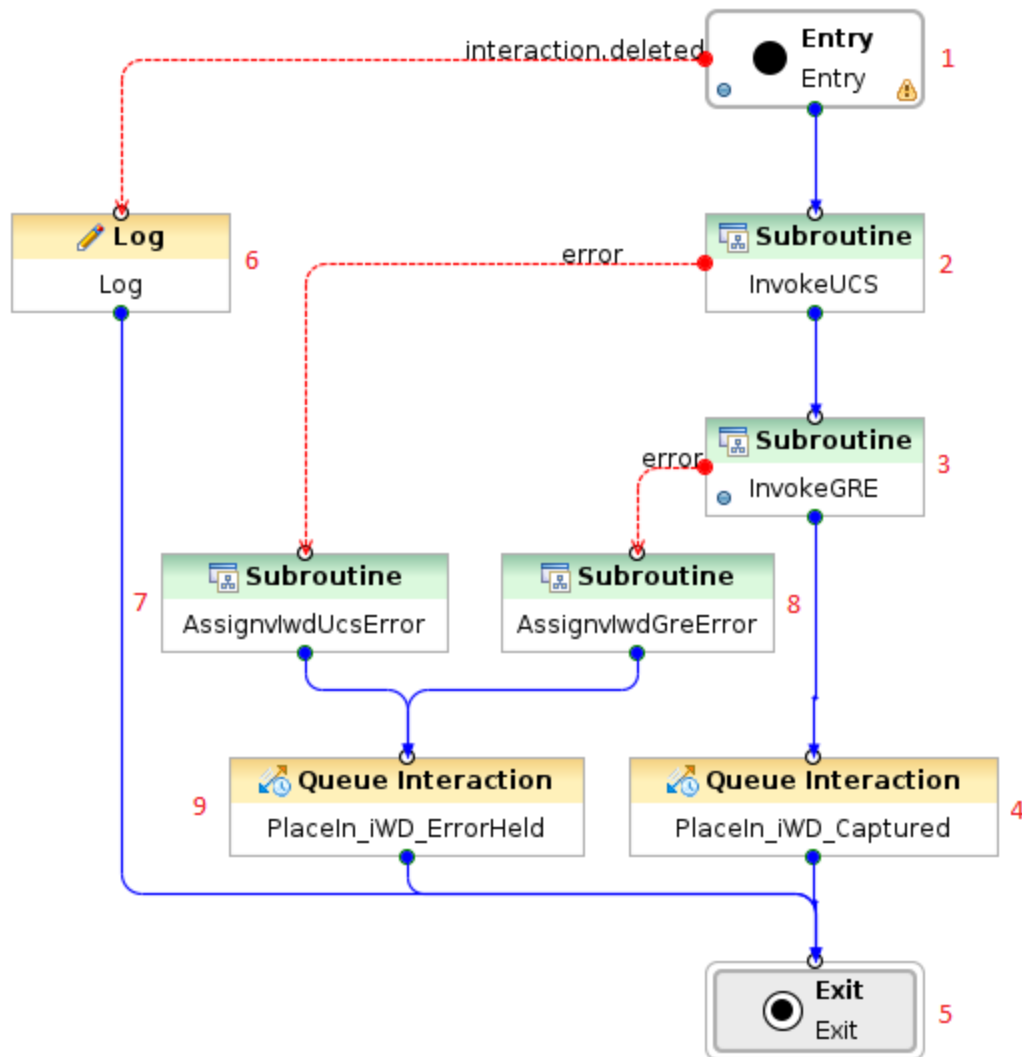
- There are no conditions here.
- Interactions are taken in order they were submitted.

Composer Configuration



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 occurred in InvokeGRE subroutine.
9. The interaction is placed in the `iwdb_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.

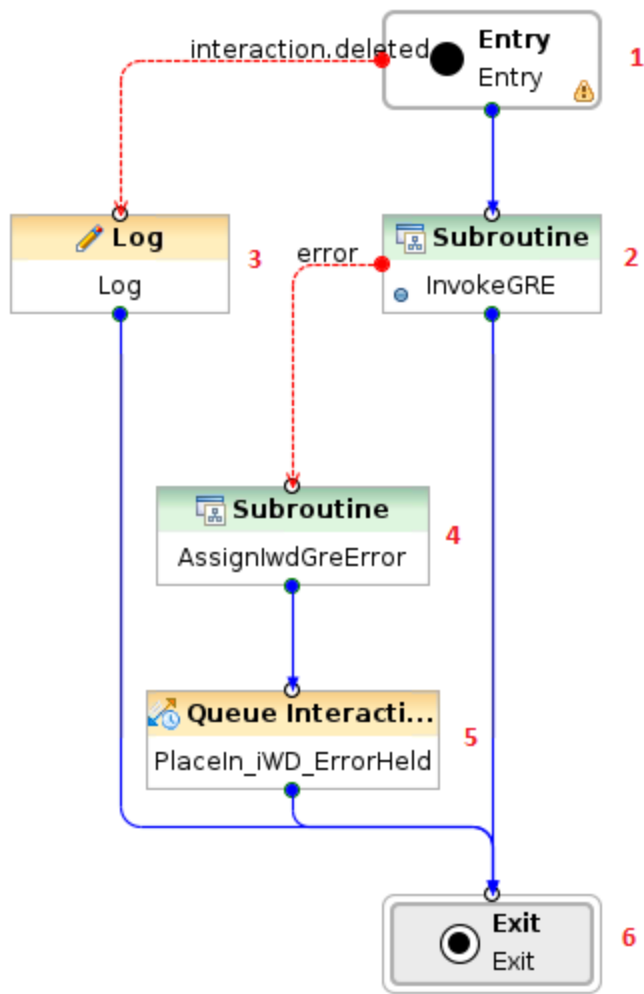
This strategy processes interactions from the following queues:

- `iwdb_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.
- `iwdb_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

The screenshot displays two overlapping windows from the Composer Configuration tool. The top window, titled 'View Properties', has a 'Configure View Options' tab. It features a list of view options on the left, including 'ToDistribute [iw...]' and 'ToReprioritize [L...]', with buttons for 'Add...', 'Remove', and 'Remove All'. The right side of this window has tabs for 'Main', 'Parameterized Conditions', and 'Segmentation'. The 'Main' tab is active, showing fields for 'Enabled' (checked), 'Name' (ToReprioritize), 'Description', 'Check interval' (0 seconds), and a 'Condition' field containing the expression: `(IWD_reprioritizeDateTime is not null) AND (_current_time()) >= IWD_reprioritizeDateTime`. The bottom window, titled 'Condition settings', has a 'Configure View Condition' tab. It shows a 'Value' field with the same condition expression: `(IWD_reprioritizeDateTime is not null) AND (_current_time()) >= IWD_reprioritizeDateTime`. Both windows have 'OK' and 'Cancel' buttons at the bottom right.

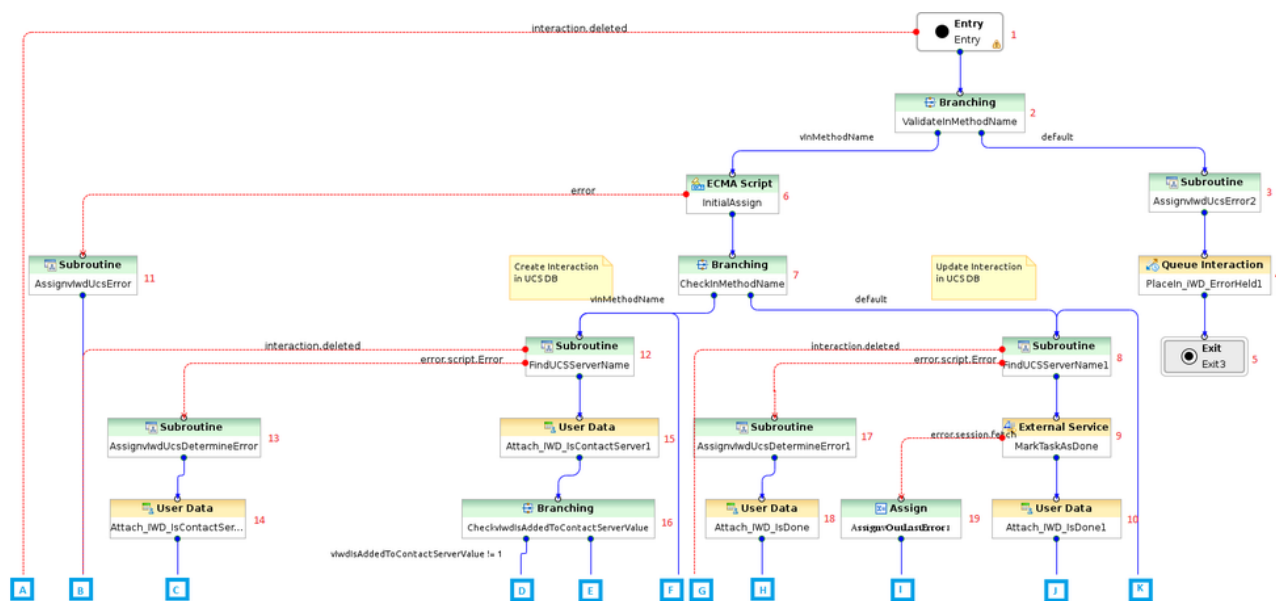
Flow Summary



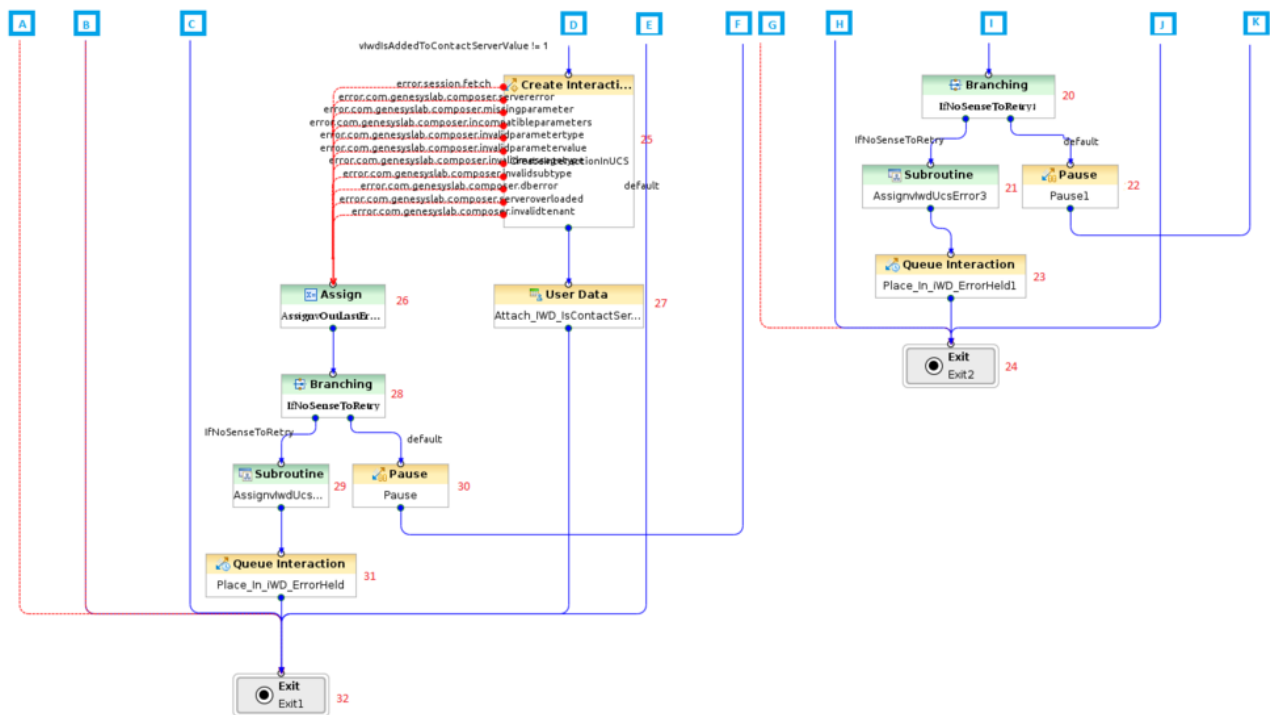
Flow Detail

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.

8



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 `FindListItem` 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 `FindListItem` 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 `FindListItem` 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 `FindListItem` 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.

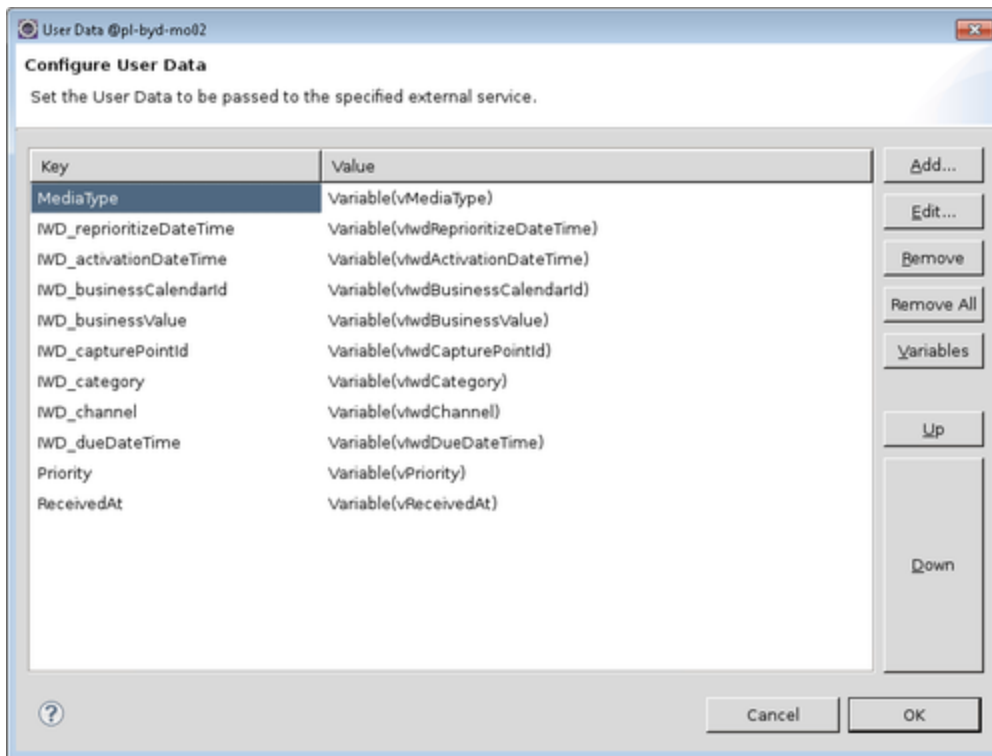
27. The user data key `IWD_isAddedToContactServer` is updated to 1 to indicate that the task was successfully added to the interaction history in UCS. The result returned from the ESP call to UCS (from See A new interaction is created in the UCS database, for this iWD task. If that function is successful 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

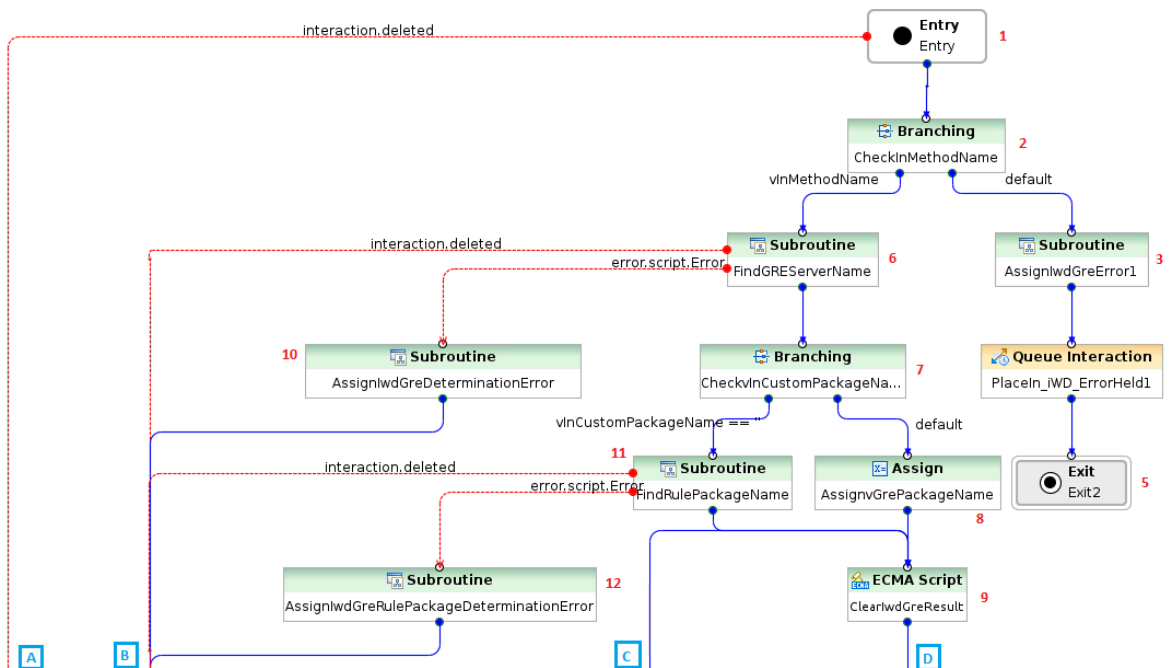
For Composer/ORS versions prior 8.1.400.48—If custom task attributes will be used in the Standard Rules Template, you must add them in the External Service block called `InvokeGRE` in the `InvokeGRE` workflow. All user-defined attributes need to be added in the User Data attribute, otherwise they will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

Composer configuration

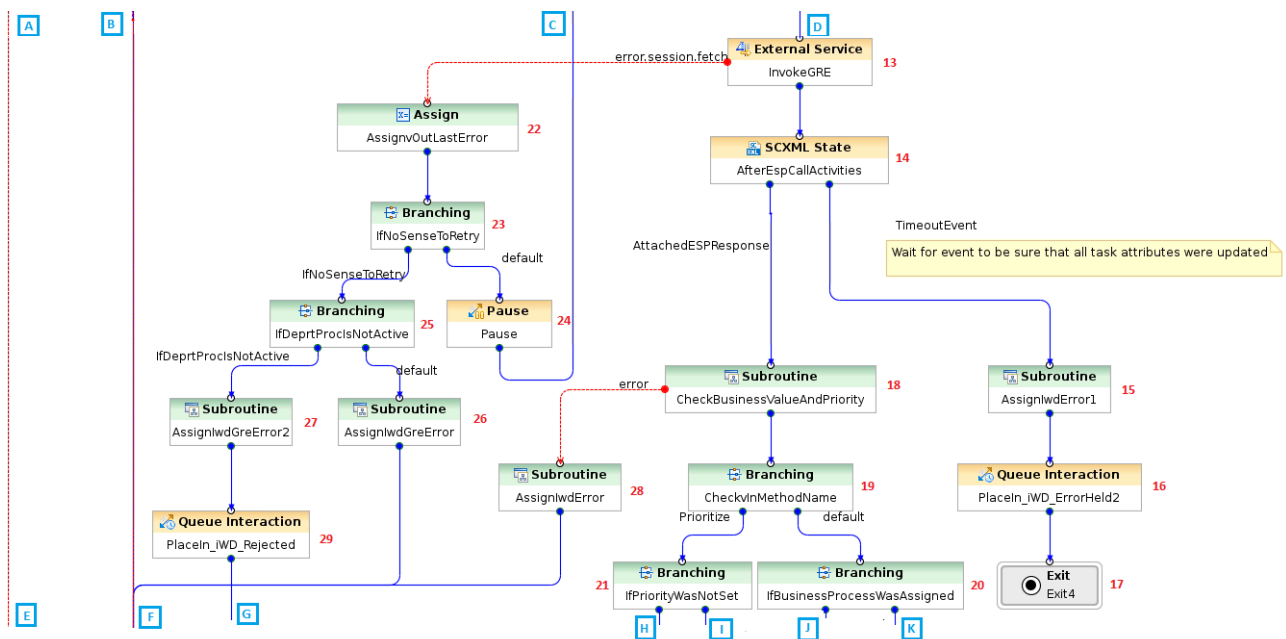


Flow Summary

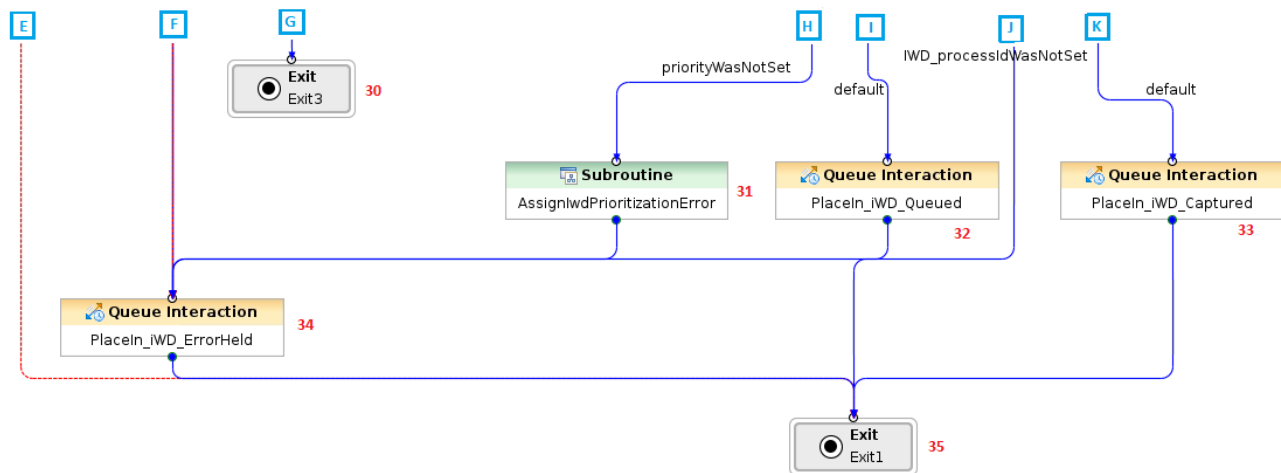
Part 1



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 **FindListItem** 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 FindListItem subroutine.`
11. The `FindListItem` 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 occurred 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.

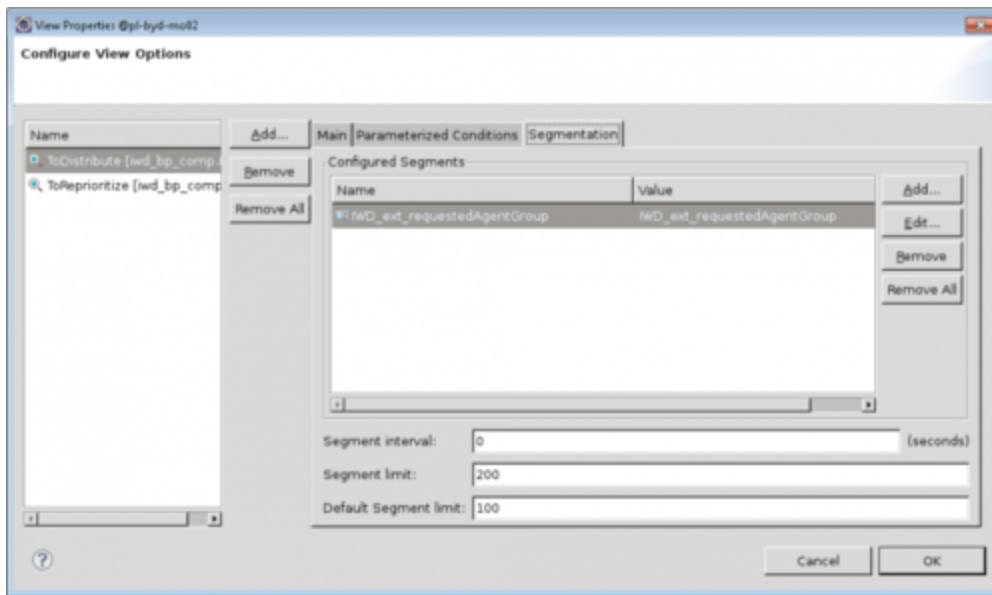
Distribution Strategy

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.

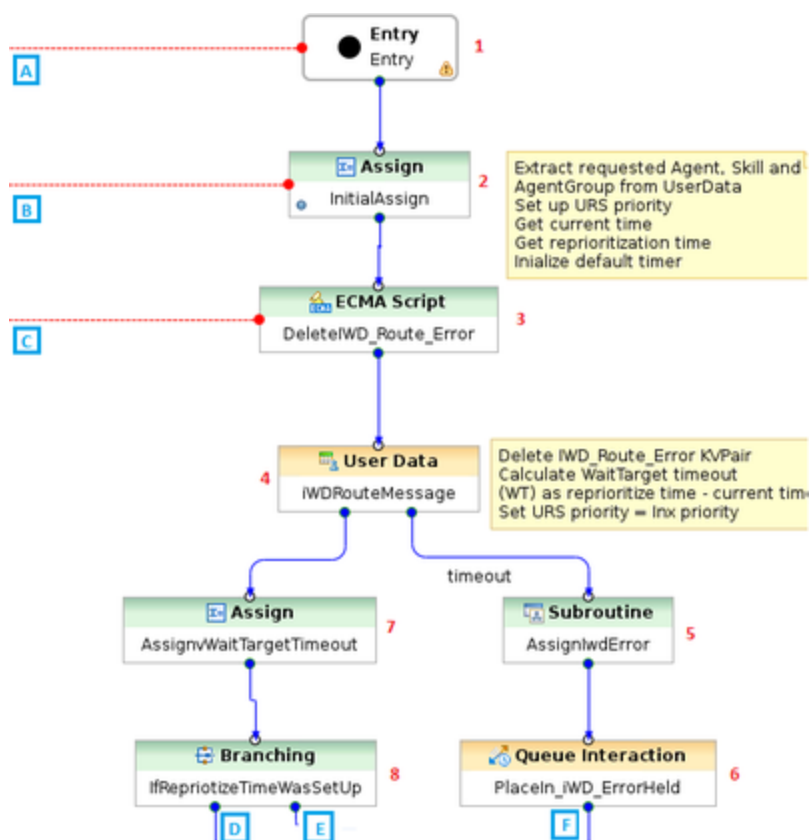
Composer Configuration - Segmentation View



Flow Summary

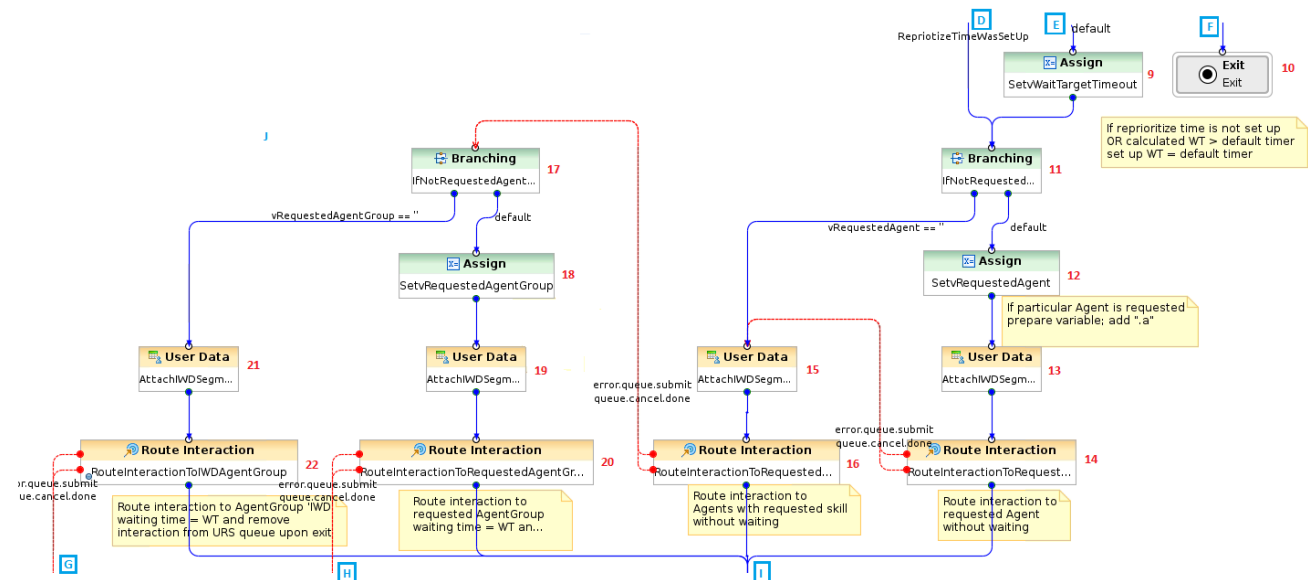
Part 1

Click to enlarge.



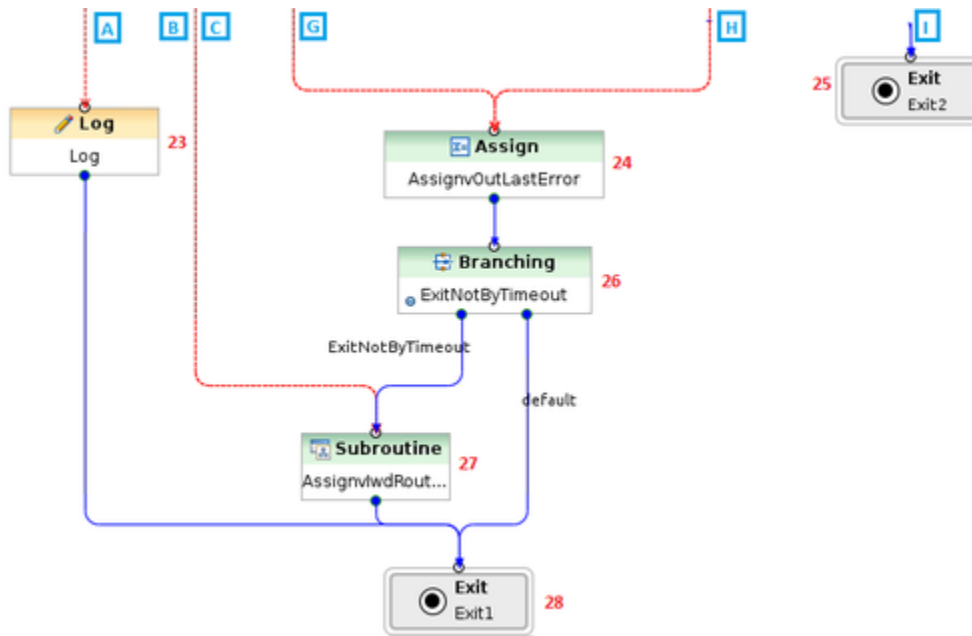
Part 2

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 vReprioritizedDTInt 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.

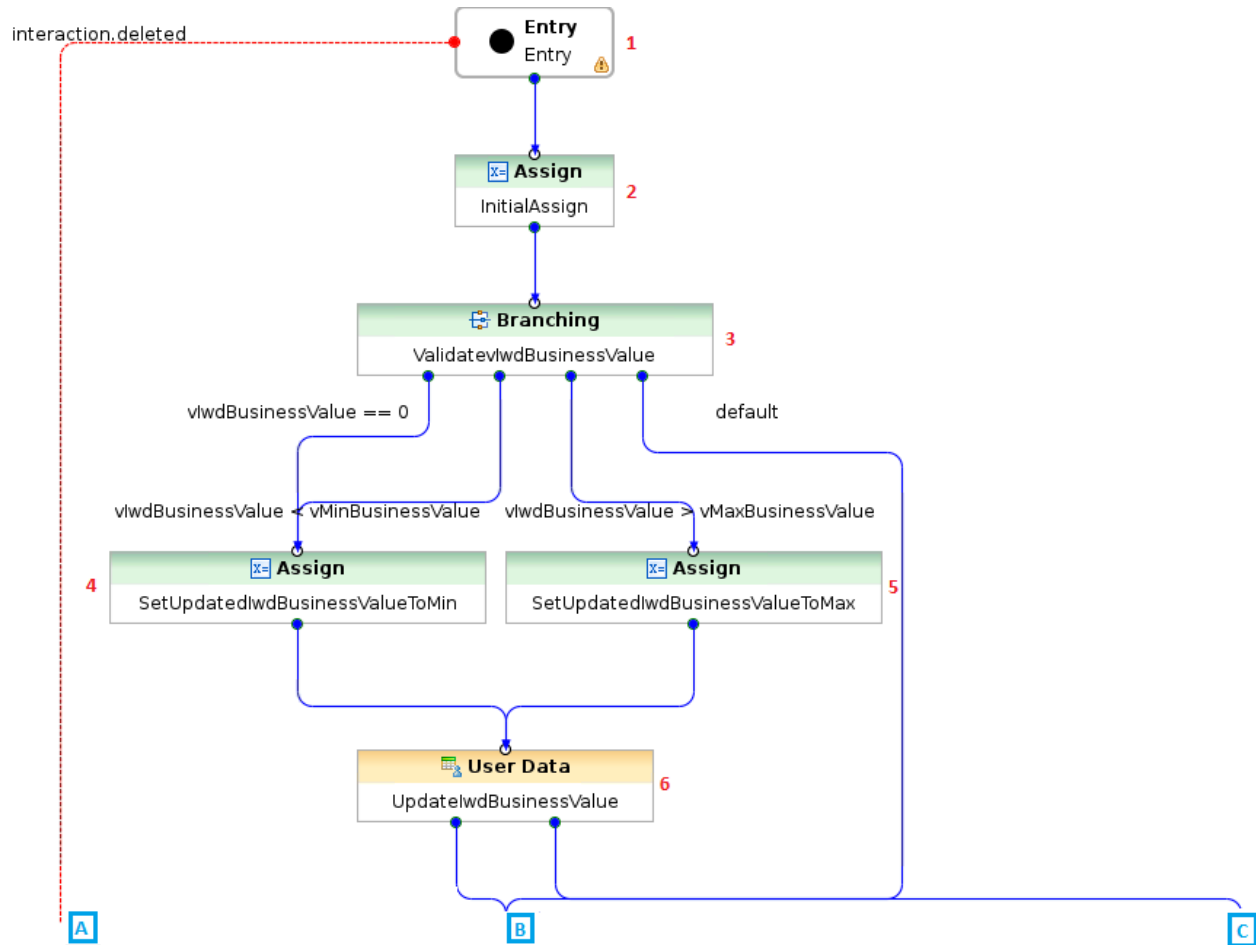
7. Calculate `vWaitTargetTimeout`.
8. Check if calculated `vWaitTargetTimeout` is in range `(0, vDefaultTargetTimeout>`.
9. Set `vWaitTargetTimeout` to `vDefaultTargetTimeout`.
10. Exit Distribution workflow.
11. Check if particular Agent is requested.
12. Assign `vRequestedAgent + '.a'` to `vRequestedAgent` 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

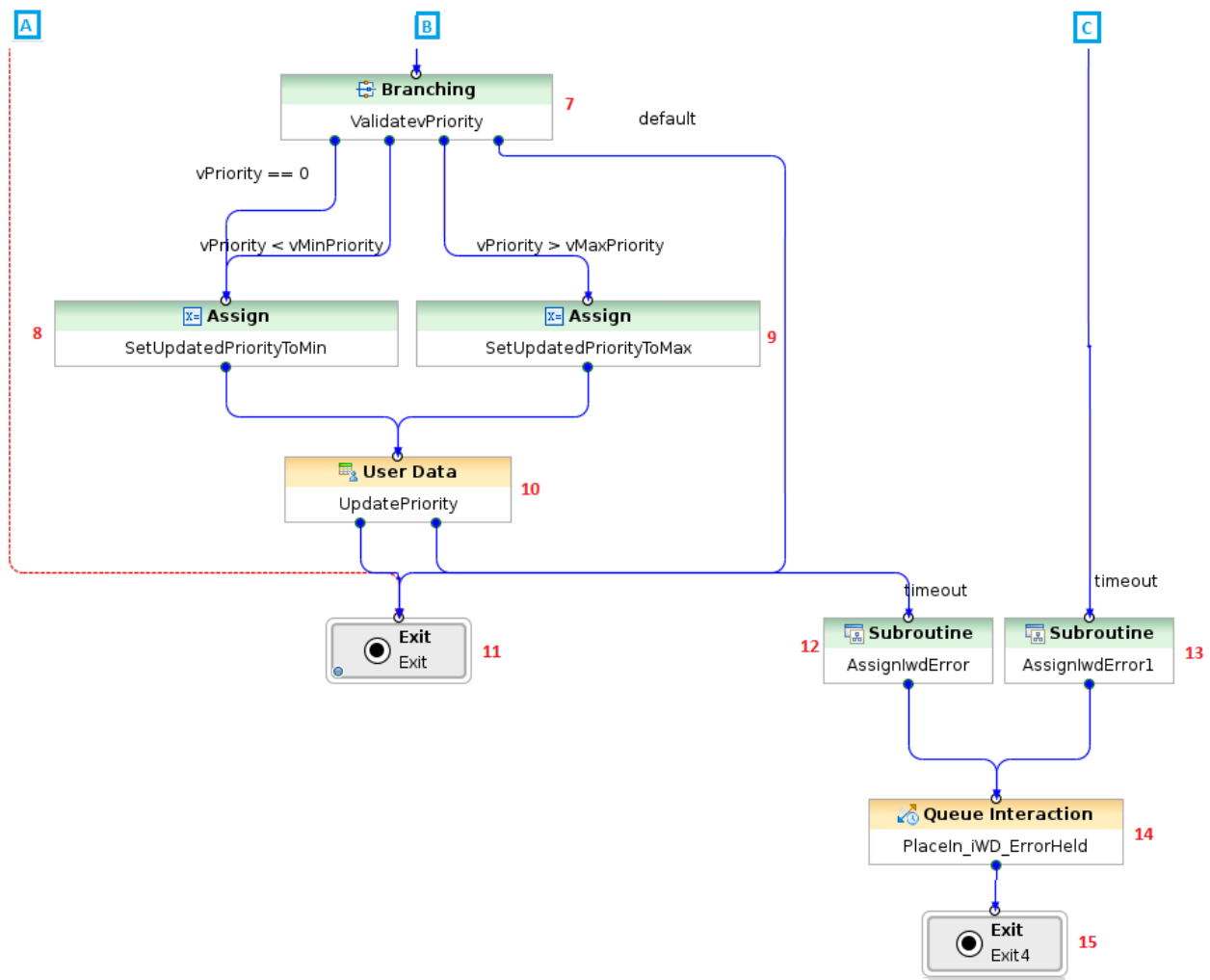
The purpose of this workflow is to verify if `Priority` and `IWD_businessValue` have correct values.

Flow Summary

Part 1



Part 1



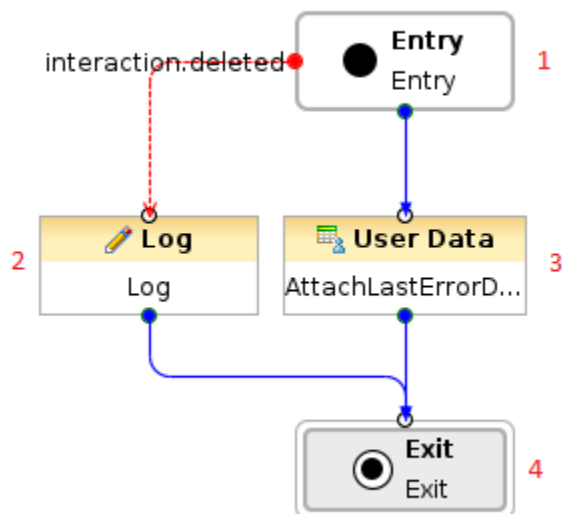
Flow Detail

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

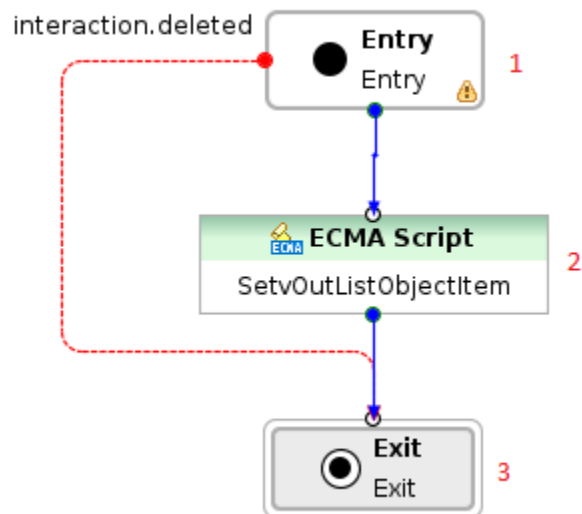
1. Entry to AssignLastError workflow.
2. The last error is attached to user data as a key-value pair with the key `vInLastErrorkey` and value `vInLastErrorString`.

`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.

FindListItem Subroutine

Flow Summary



Flow Detail

1. Entry to FindListItem 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.

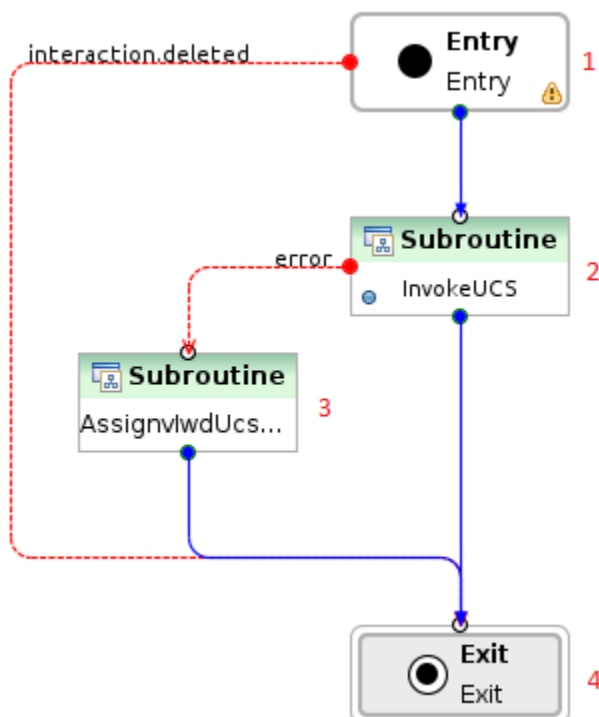
MarkInteractionAsDone Strategy

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

The screenshot displays two overlapping windows from the Composer Configuration tool. The top window, titled 'View Properties', has a 'Configure View Options' tab. It features a list of view names on the left, including 'ExpiredOnly...' and 'MarkAsDon...'. To the right of this list are buttons for 'Add...', 'Remove', and 'Remove All'. The main area of this window is divided into three tabs: 'Main', 'Parameterized Conditions', and 'Segmentation'. The 'Main' tab is active, showing fields for 'Enabled' (checked), 'Name' (set to 'ExpiredOnly'), 'Description' (empty), 'Check interval' (set to 60 seconds), and a 'Condition' field containing a complex logical expression. The bottom window, titled 'Condition settings', has a 'Configure View Condition' tab. It shows a large text area for the condition value, which contains the same logical expression as the one in the top window. At the bottom right of this window are 'OK' and 'Cancel' buttons.

View Properties - Configure View Options

View: ExpiredOnly...

View: MarkAsDon...

Buttons: Add..., Remove, Remove All

Tab: Main

Enabled: ☒

Name: ExpiredOnly

Description:

Check interval: 60 (seconds)

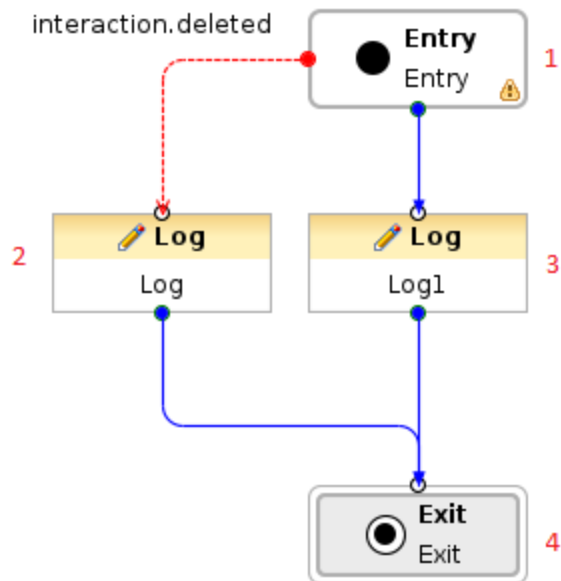
Condition: ((IWD_expirationDateTime is null OR (_current_time() >= IWD_expirationDateTime)) AND ((IWD_isContactServer = 1 and IWD_isDone = 1) or (

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))

Buttons: OK, Cancel

Flow Summary



Flow Detail

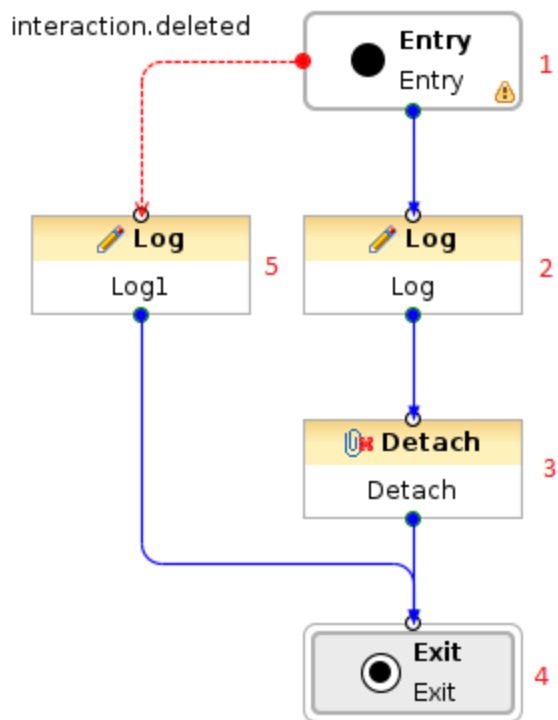
1. Entry to Removal workflow.
2. Log message in case if interaction was from some reasons deleted.
3. Log message: Task will be terminated on exit
4. Exit Removal workflow.

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.