Working with the iWD Business Process in IRD

intelligent Workload Distribution 9.0.0

12/29/2021

Table of Contents

Working with the iWD Business Process in Interaction Routing Designer (IRD)

This Document

These topics describe how to:

- Work with and adapt the Interaction Routing Designer (IRD) iWD Business Process (IWDBP) that is supplied out-of-box with intelligent Workload Distribution.
- Clone the IWDBP to create new business processes under a single Tenant

Working with and Adapting the iWDBP

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

- NEW
- ERRORHELD
- CAPTURED
- COMPLETED
- CANCELED
- REJECTED
- QUEUED

Within this Business Process, from within a routing strategy, External Service Protocol (ESP) 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 ESP 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.

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 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.

See the **Deployment Guide** for more information.

New Features by Release

New in 9.0.0

There are no new features in the iWD Business Process for Interaction Routing Designer in release 9.0.0

Prerequisites

Software Requirements

The general software prerequisites for iWD must be in place, plus:

- Interaction Routing Designer 8.1.2 or higher.

Other Information Resources

- The **Universal Routing 8.1 Deployment Guide** describes how to have the Interaction Design shortcut bar appear in IRD, if it has not appeared automatically.
- The **Universal Routing 8.1 Business Process User's Guide** provides an in-depth discussion of business processes.
- The **Universal Routing 8.1 Interaction Routing Designer Help Zip** describes how to create, save, import and export a business process, and how to load the strategies that comprise the business process.

> ### Important
> The IRD Main Window contains a Business Processes list pane that is disabled by default, but can be enabled

iWD Business Process - Contents

The iWD business process (IWDBP) contains the following strategies:

- Classification
- Prioritization
- Distribution
- MarkInteractionAsDone
- Removal
- InvokeUCS
- InvokeGRE

The iWD business process contains the following subroutines:

- FindObjectListName

> **Important**
> In release 8.5.105, the DetermineESPServerName and DetermineRulePackageName subroutines are amalgamated into and replaced by FindObjectListName

- CheckBusinessValueAndPriority

The iWD business process contains the following queues:

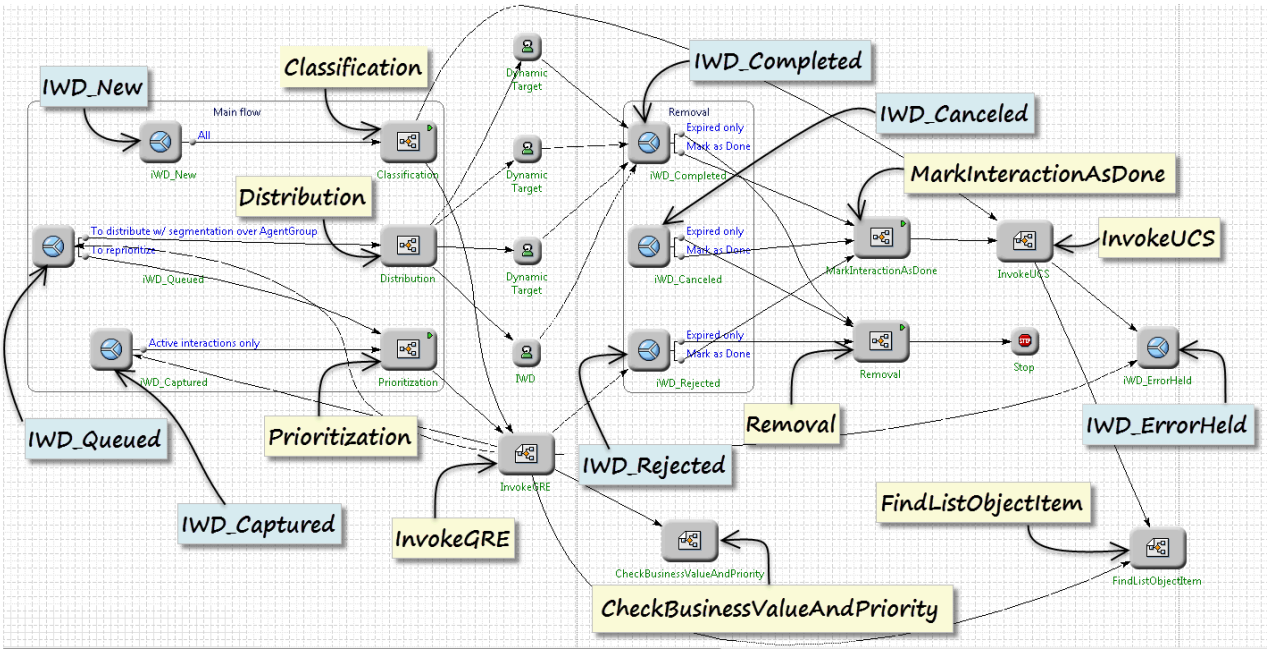
- iWD_New
- iWD_Captured
- iWD_Queued
- iWD_Canceled
- iWD_Rejected
- iWD_Completed
- iWD_ErrorHeld

The Interaction Queues that are included in the out of the box IWDBP 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.

IWDBP Main Process Diagram

9.0

The graphic below shows the entire business process as it appears in the `Interaction Design` window of Interaction Routing Designer in release 9.0. Click to expand the graphic.



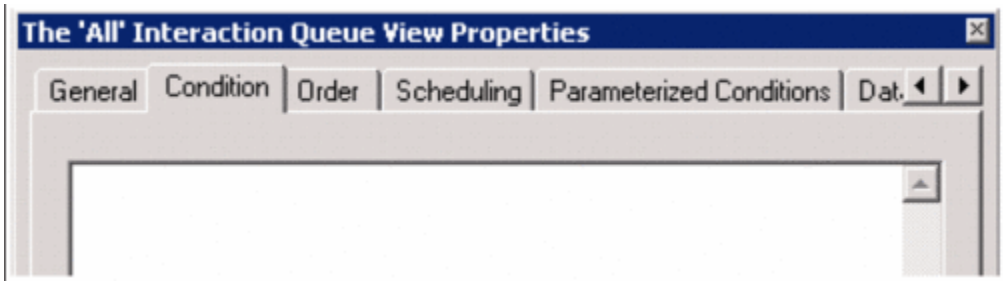
IWDBP Strategies & Subroutines

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.

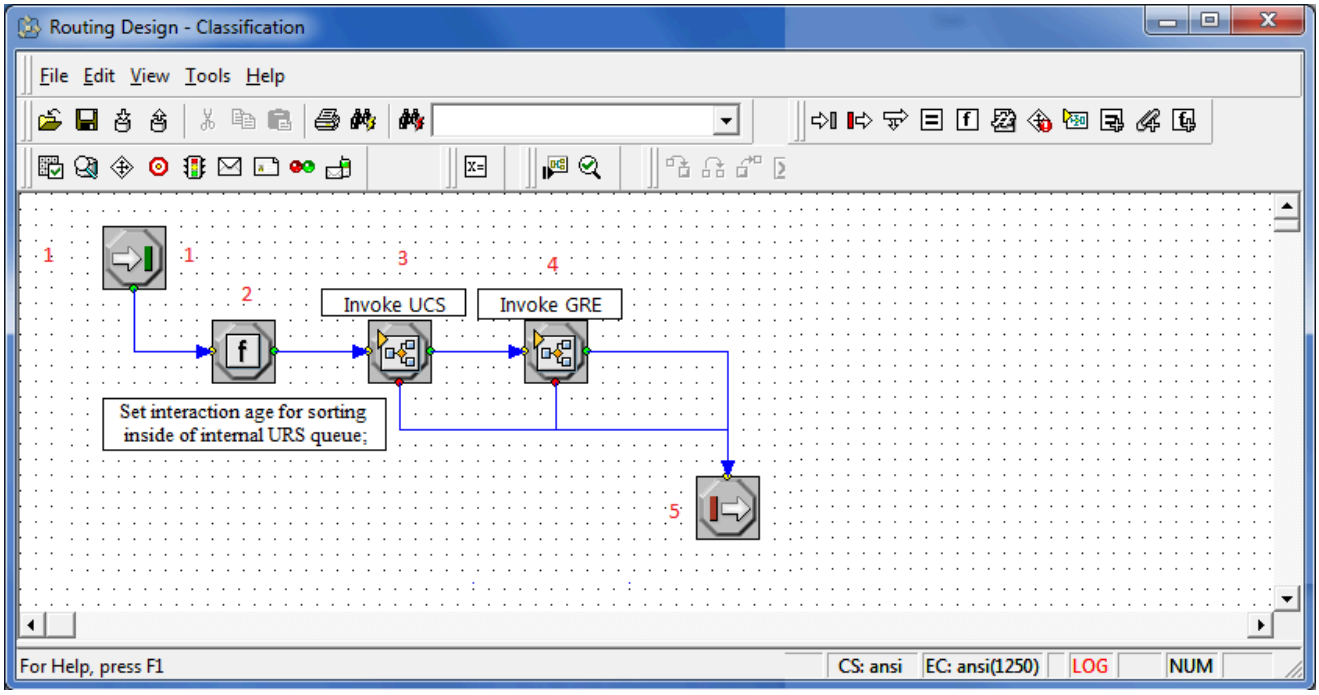
This strategy processes interactions from the following queues:

- `iWD_New`—Interactions have to satisfy the following conditions:
  - There are no conditions here.
  - Interactions are taken in order they were submitted.



Flow Summary

Click to enlarge.



Flow Detail

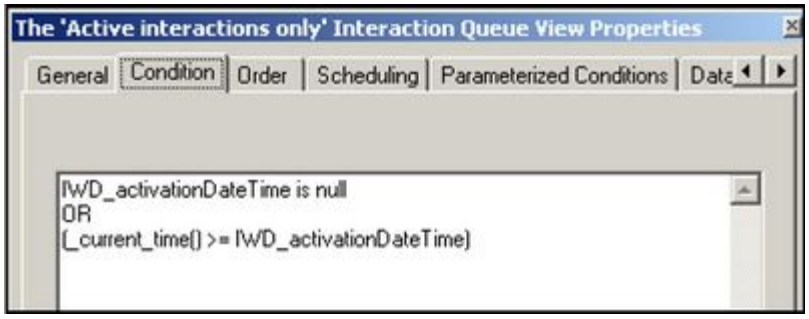
1. Entry to the Classification strategy.
2. A command is sent to URS to use the interaction age while sorting interactions in internal queues.
3. The InvokeUCS subroutine is invoked to create a new interaction in the UCS database.
4. The InvokeGRE subroutine is invoked.
5. Exit Classification strategy.

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.

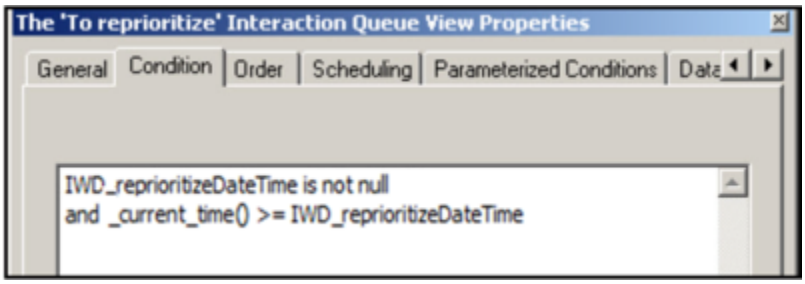
This strategy processes interactions from the following queues:

- `iWD_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.



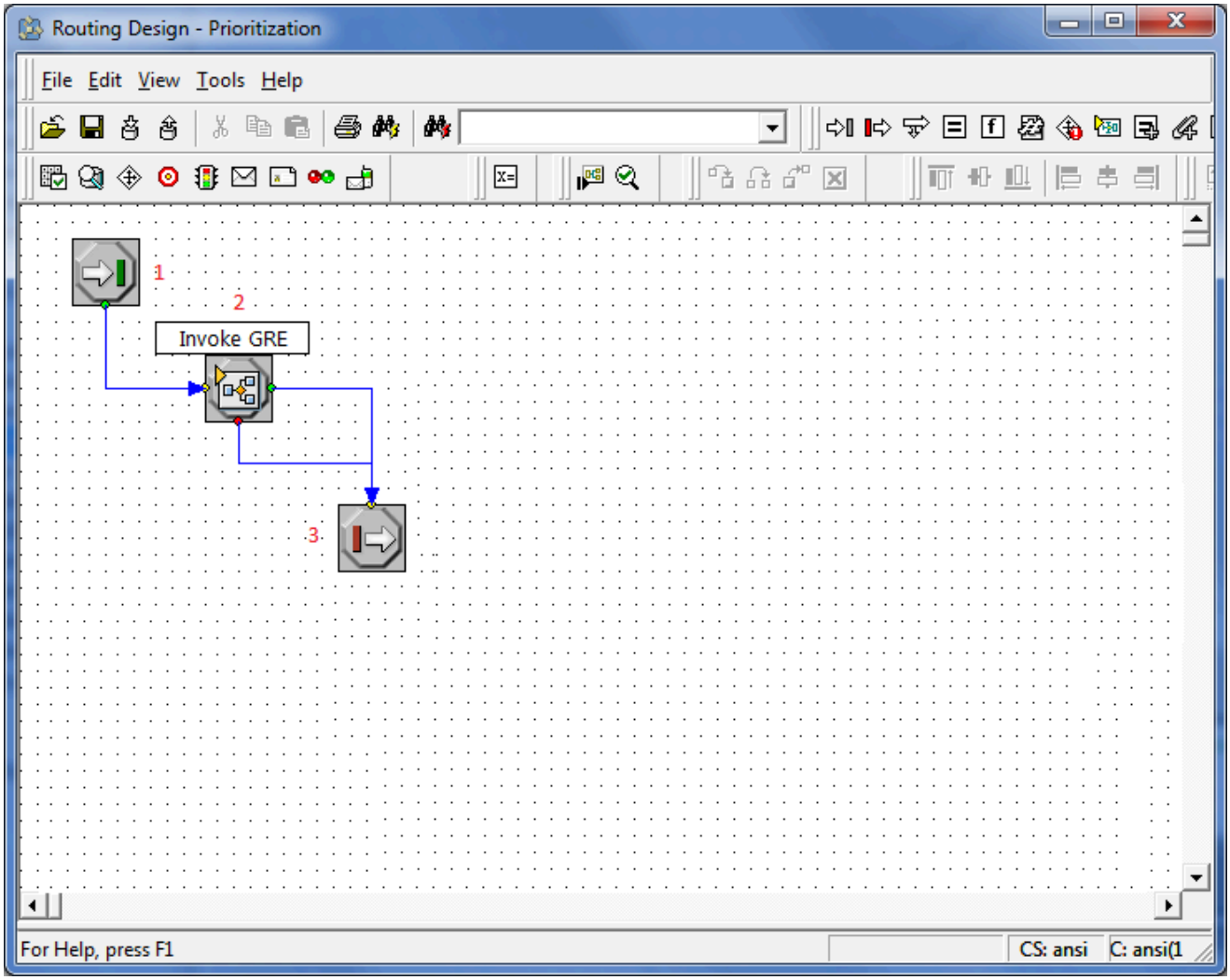
Active Interactions only

- `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).



For reprioritization

Flow Summary



Flow Detail

1. Entry to the Prioritization strategy.
2. The InvokeGRE subroutine is invoked.
3. Exit Prioritization strategy.

Distribution Strategy

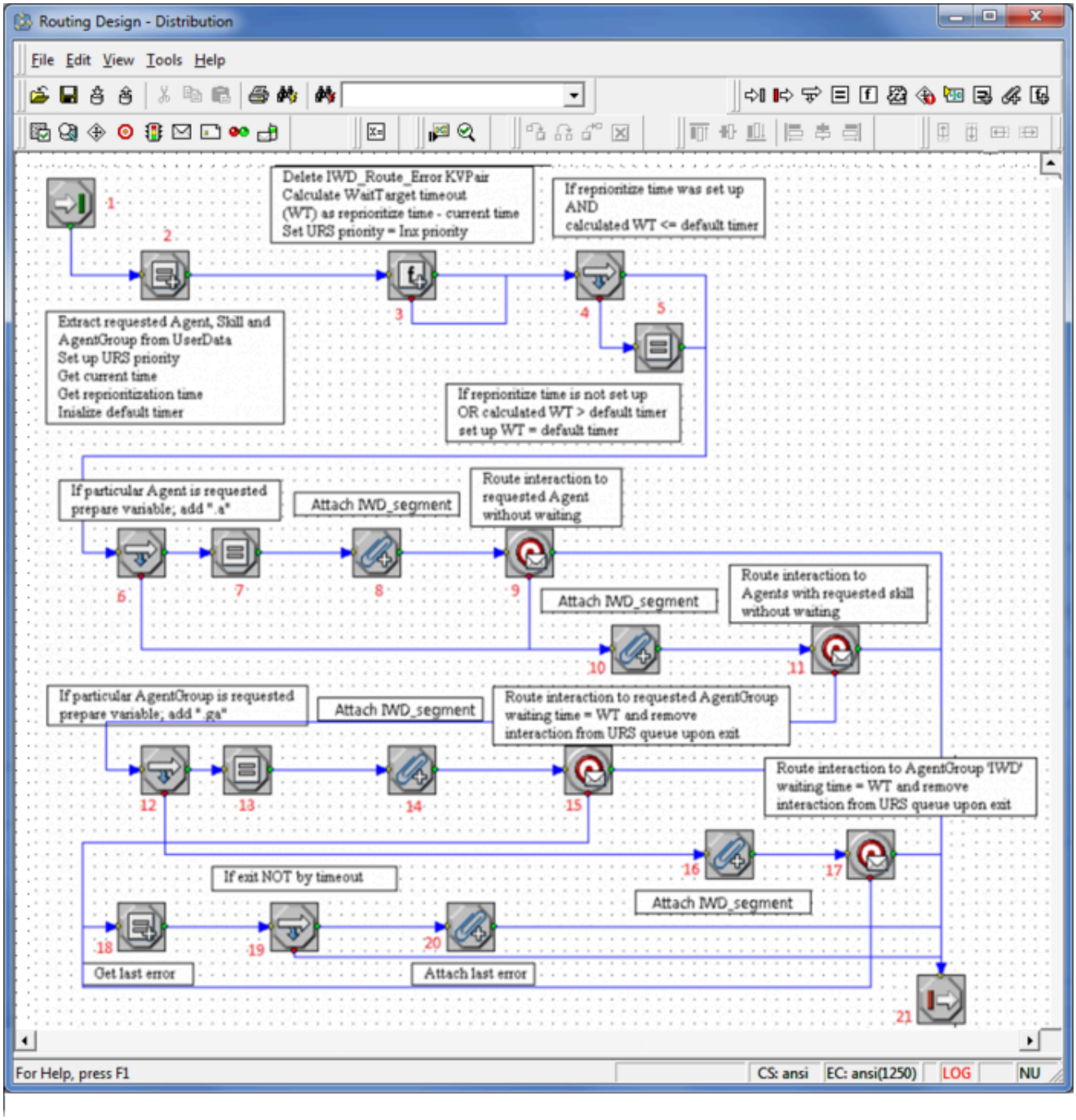
This strategy routes interactions to a requested Agent, requested Agent Group, requested Skill, or to the default iWD Agent Group, and can now process an `IWD_Segment` attribute from the segmentation

settings.

This strategy processes interactions from the following queues:

- `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)

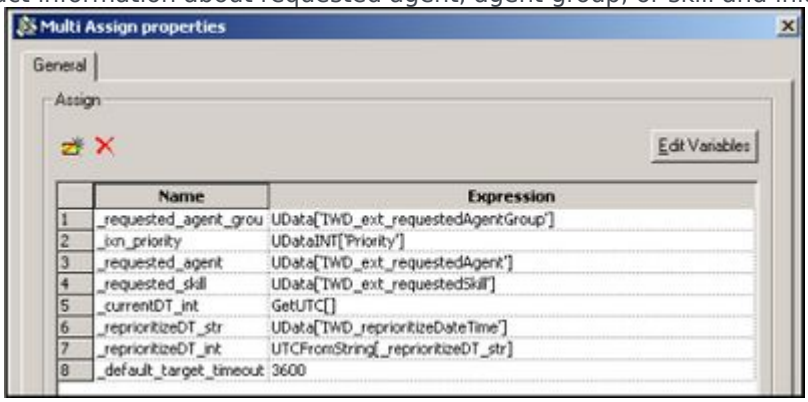
Flow Summary



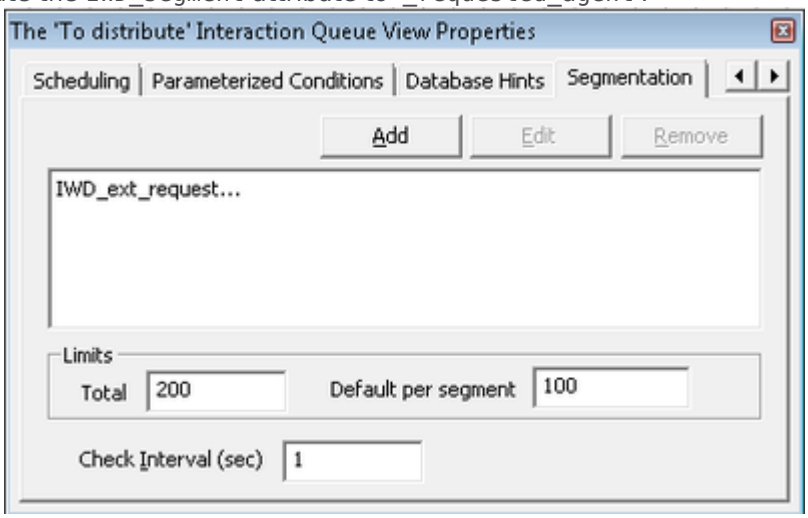
Flow Detail

1. Entry to the Distribution strategy.

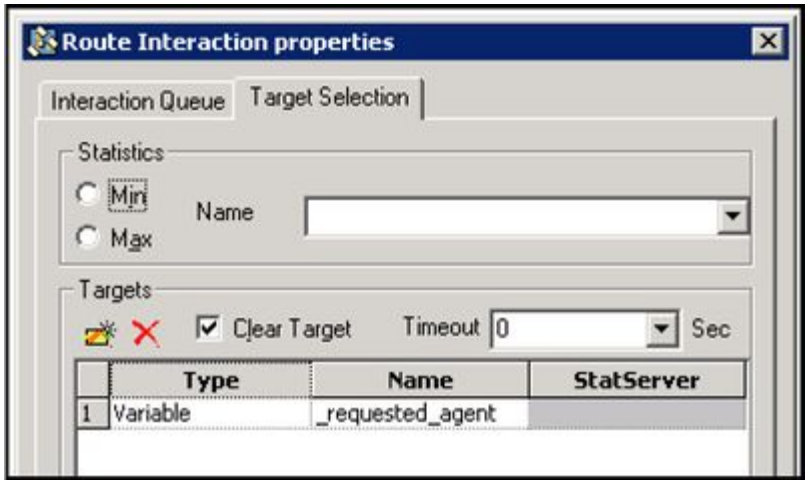
2. Extract information about requested agent, agent group, or skill and initialize internal variables.



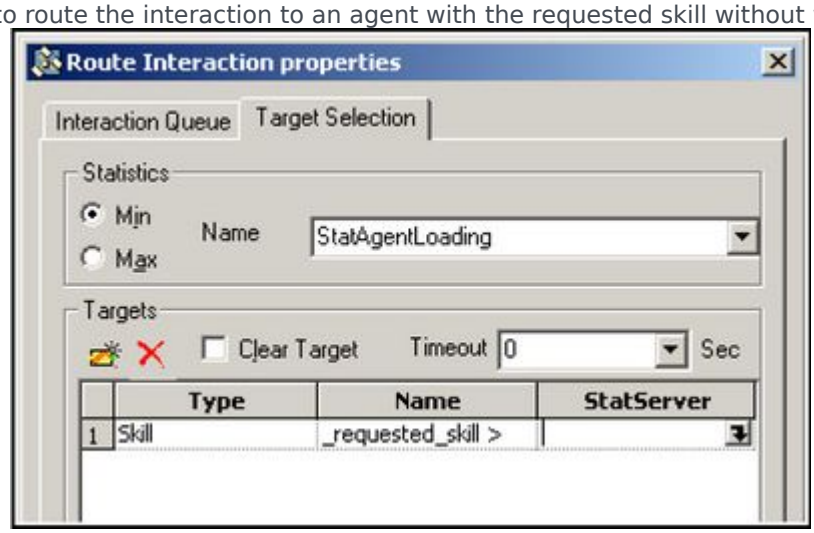
3. A calculation is done to determine the timeout—how long the interaction should wait for a target to become available.
4. If the reprioritize time was set up and the calculated timeout is less than or equal to the default timeout (`1 hour`, see Step 1), then the timeout remains as it is.
5. If the reprioritize time was not set, or the calculated timeout is greater than the default timeout, then the waiting timeout is set to the default (`1 hour`).
6. Analysis is done to determine whether an agent was requested.
7. If an agent was requested, the URS variable is prepared (`.a` is added).
8. Update the `IWD_segment` attribute to '`_requested_agent`'.



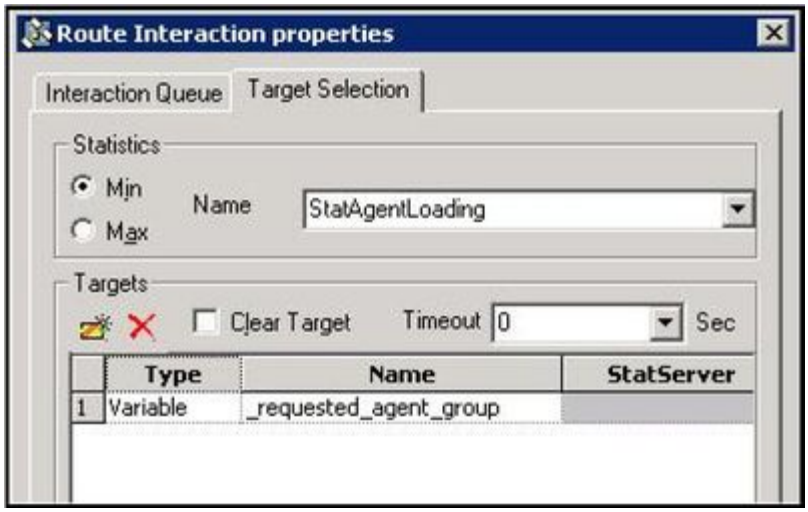
9. Try to route the interaction to the requested agent without waiting.



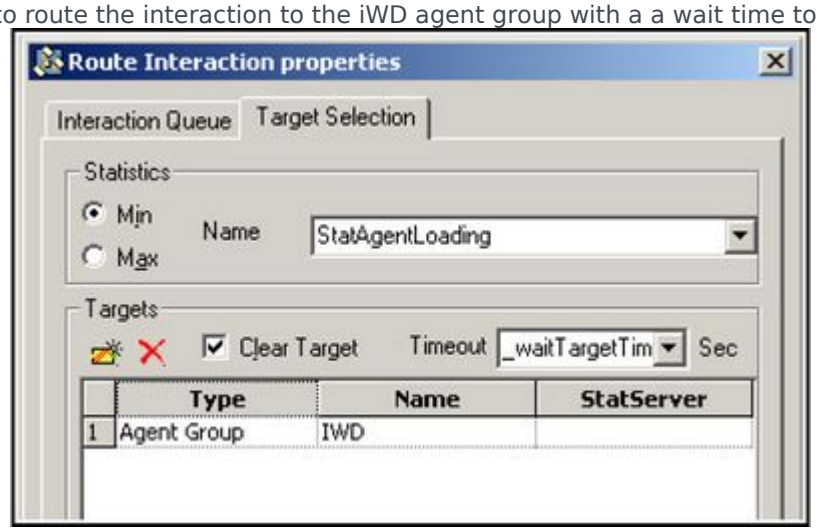
10. Update the `IWD_segment` attribute to '`_requested_skill`'.
11. Try to route the interaction to an agent with the requested skill without waiting.



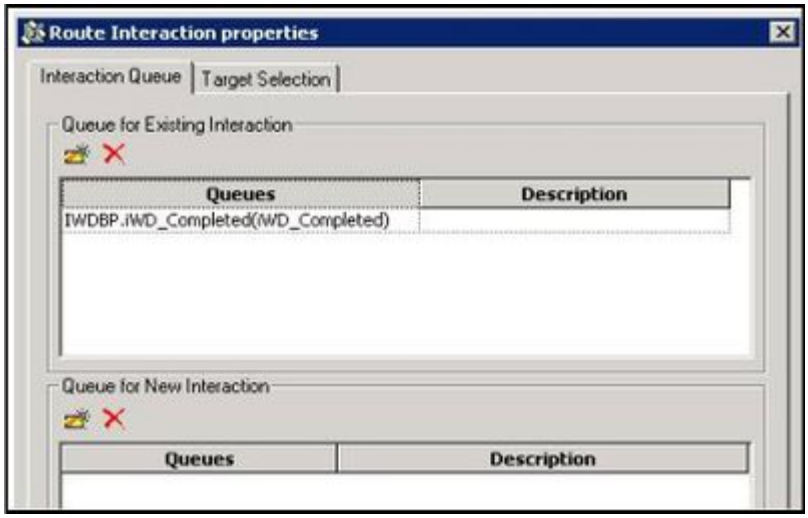
12. Analysis is done to determine whether an Agent Group was requested.
13. If an Agent Group was requested, the URS variable is prepared (`.ga` is added).
14. Update the `IWD_segment` attribute to '`_requested_agent_group`'.
15. Try to route the interaction to the requested Agent Group with wait time set to `_waitTargetTimeout`.



16. Update `IWD_segment` attribute to '`_default`'.
17. Try to route the interaction to the iWD agent group with a a wait time to `_waitTargetTimeout`.



18. Get the last error.
19. Verification is done as to why the target was not found.
20. An error code is attached in case of any error other than a timeout. If more than one target is available, URS uses the `StatAgentsLoading` statistic to select the Agent who has the minimum load (this applies to routing to Skills and routing to Groups only; routing to a requested Agent does not use statistics). For more information about this statistic, see the Universal Routing 8.1 Reference Manual.
    The Route Interaction object also has an `Interaction Queue` tab. (This applies to all three Route Interaction objects in this strategy.)



The optional `Interaction Queue` tab enables you to specify two types of queues:

- Queues for existing interactions (the queue in which the interaction should be placed after the agent is done working with it).
- Queues for new interactions (the queue in which new interactions created by the agent should be placed).

A Description (optional) appears as a hint on the agent desktop as to where to place the interaction.

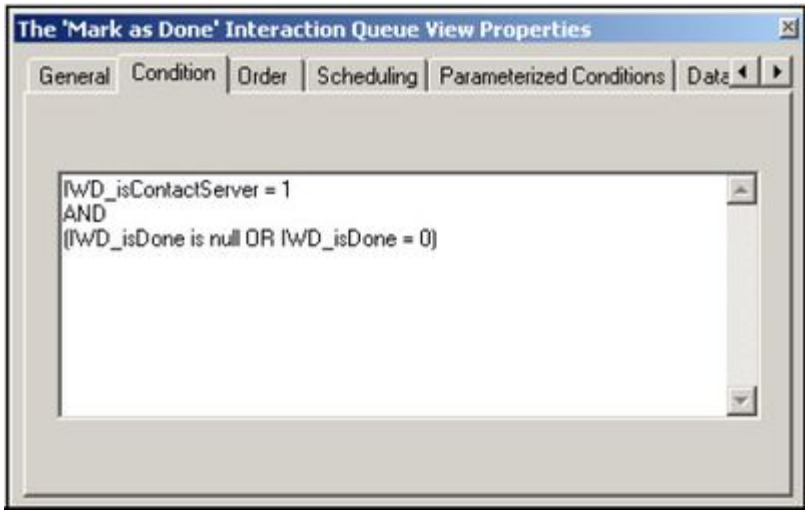
21. Exit from the Distribution strategy.

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)

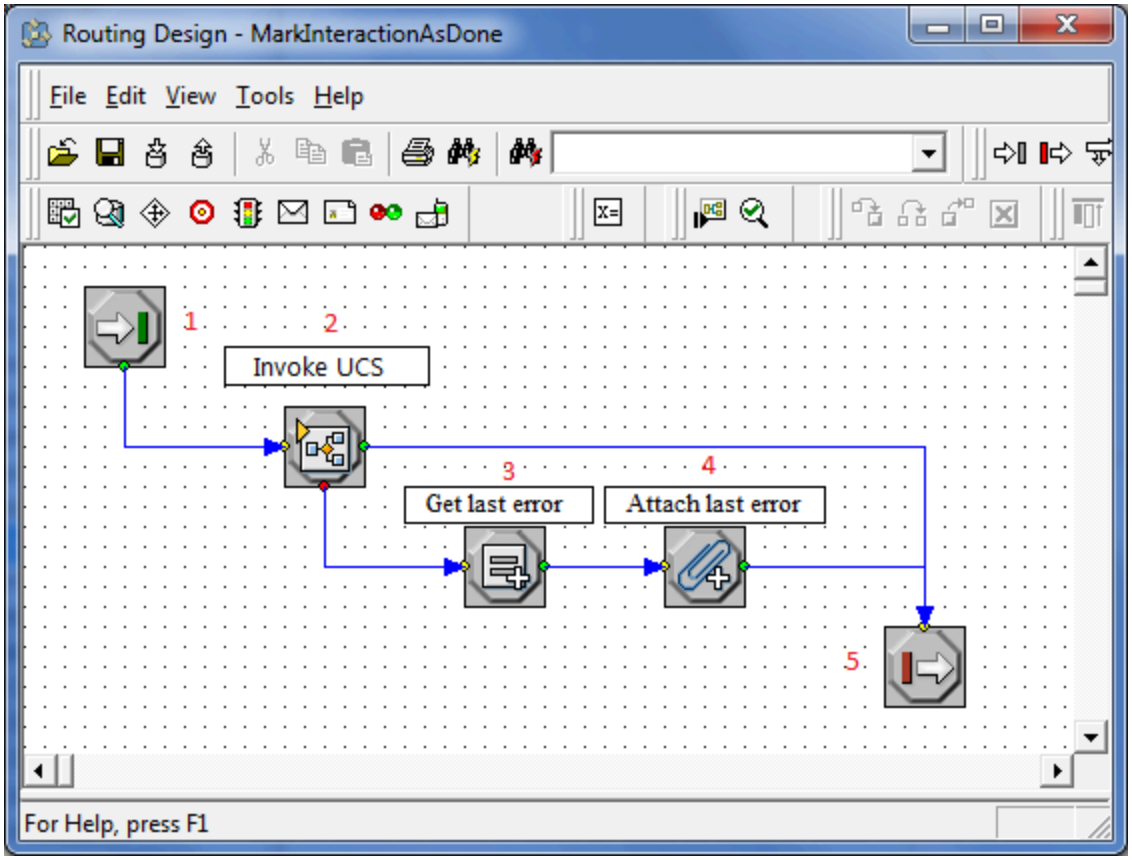


MarkInteractionAsDone View

This strategy processes interactions from the following queues:

- `iWD_Completed`
- `iWD_Canceled`
- `iWD_Rejected`

Flow Summary



Flow Detail

1. Entry to MarkInteractionAsDone strategy.
2. The InvokeUCS subroutine is invoked to complete interaction in the UCS database.
3. Reads `_last_error_str_` from InvokeUCS.
4. The last error is attached to user data as a key-value pair with the key `IWD_UCS_Error`.
5. Exit MarkInteractionAsDone strategy.

Removal Strategy

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

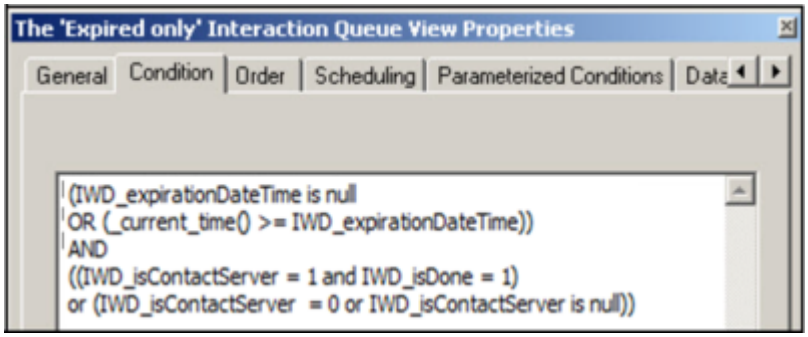
A key-value pair in user data with the key `IWD_expirationDateTime` contains information about when an interaction has to be deleted.

This strategy processes interactions from the following queues:

- `iWD_Completed`
- `iWD_Canceled`
- `iWD_Rejected`

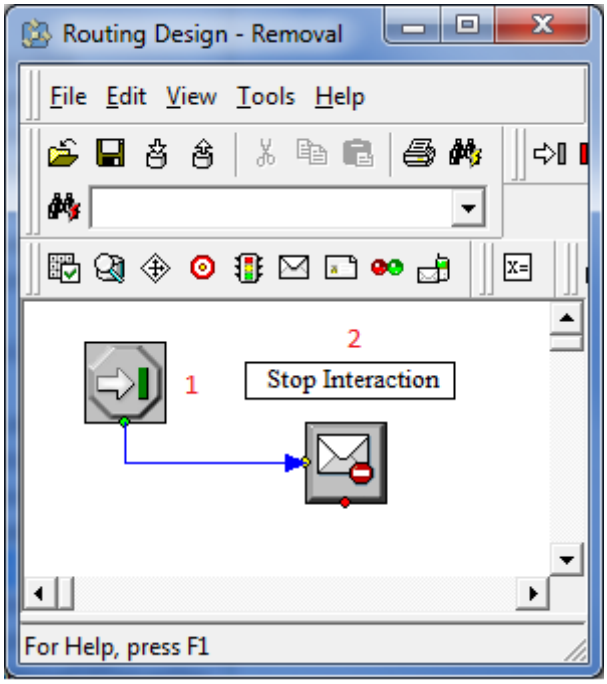
Interactions have to satisfy the following conditions:

- 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.



The 'Expired only' Interaction Queue View Properties

Flow Summary



Flow Detail

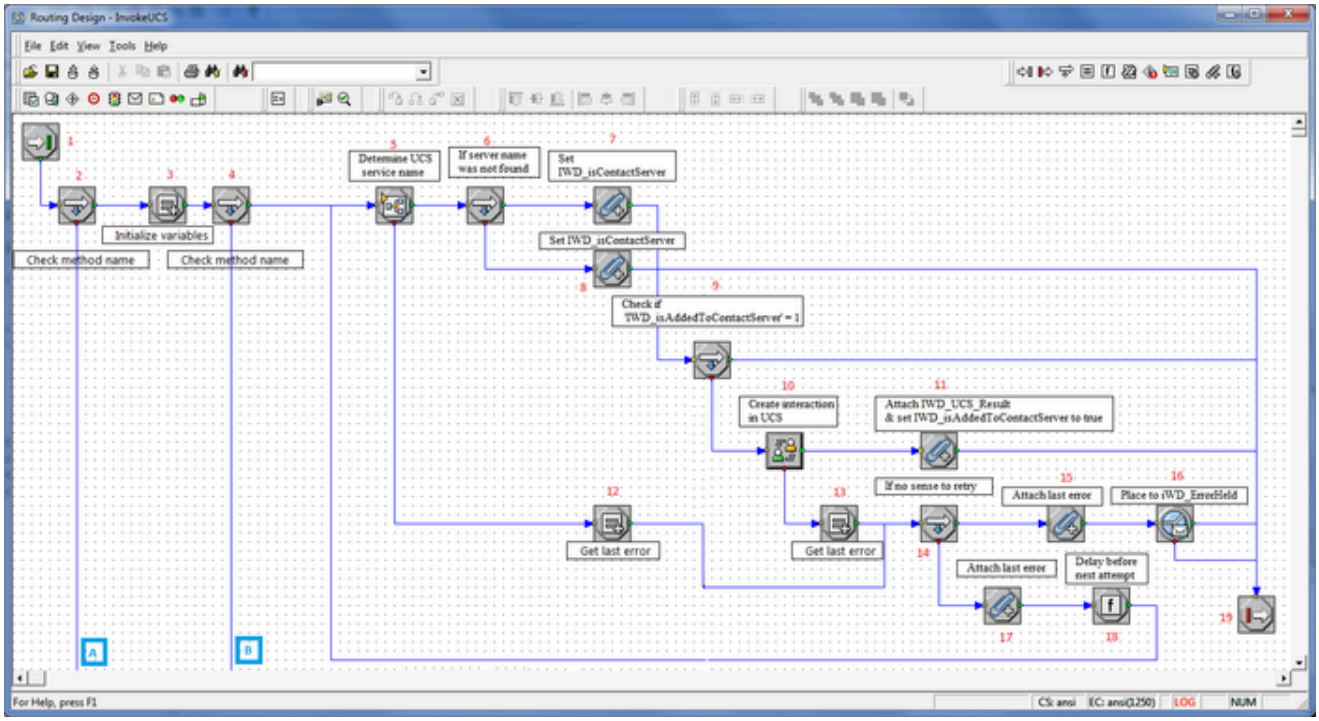
1. Start.
2. When interactions enter the Removal strategy, the processing of the interaction is stopped. This means that the interaction is deleted from the Interaction Server database.

Invoke UCS Strategy

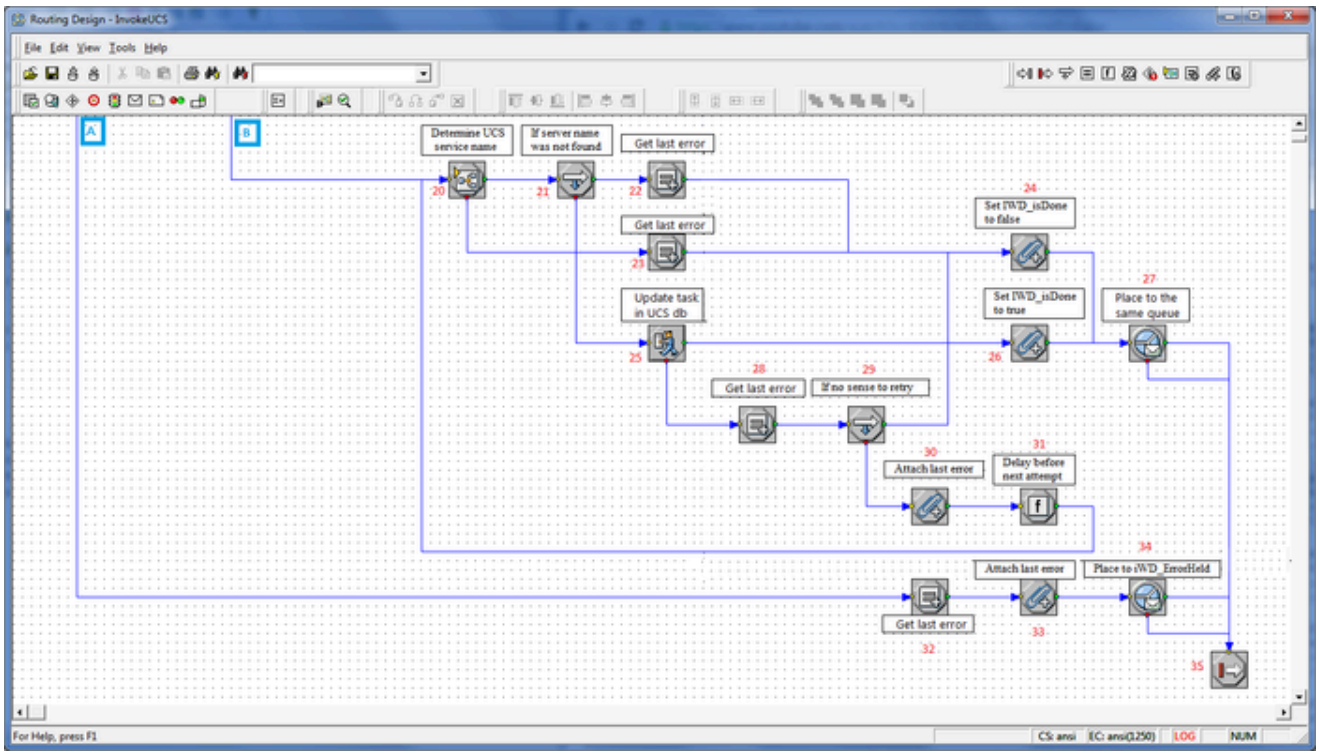
Flow Summary

Part 1

Click to enlarge.



Part 2



Click to enlarge.

Flow Detail

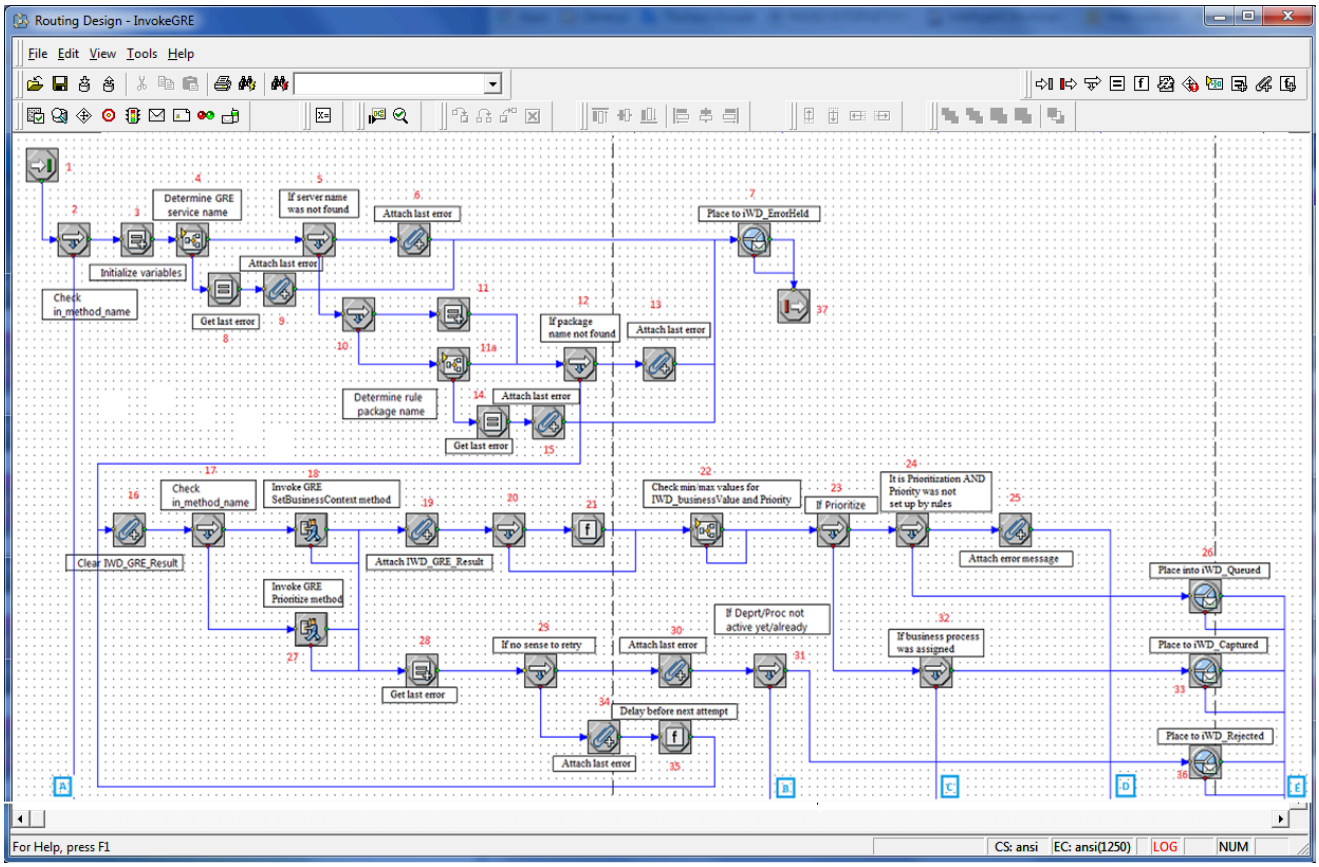
1. Entry to InvokeUCS strategy.
2. Check if `in_method_name = 'Create' | in_method_name = 'OMInteractions'`.
3. Initialize variables:
   - `_tenant_id`—Read from task attribute `TenantId`.
   - `_interaction_id`—Read from task attribute `InteractionId`.
   - `delay_ms`—Specifies the delay (in milliseconds) between attempts to invoke UCS.
4. Check if `in_method_name = 'Create'`.
5. The DetermineESPServerName subroutine is invoked to determine the correct ESP server name to use. The subroutine uses the List Object list called `ContactServerList`. This subroutine also sets up cases when there is reason to retry to invoke the ESP server.
6. If the subroutine was successful, a check is done to ensure the existence of the ESP Server name that was returned by the subroutine.
7. The value of the user data key `IWD_isContactServer` is set to 1.
8. The value of the user data key `IWD_isContactServer` is set to `0`.
9. URS checks to see if the value of the user data key `IWD_isAddedToContactServer` is equal to 1, indicating that the task is already written into the interaction history in the UCS database.
10. A new interaction is created in the UCS database for this iWD task. If that function is successful, flow goes to 11. 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 is written to the variable `IWD_UCS_Result`.
11. 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`.
12. If the subroutine fails an error is extracted.
13. If the creation of the interaction in UCS was unsuccessful, an error is extracted from user data.
14. A check is done to see if the error code is related to the ESP server communication.
15. This error is attached to user data as a key-value pair with the key `IWD_UCS_Error`.
16. The interaction is placed in the `iWD_ErrorHeld` queue.
17. This error is attached to user data as a key-value pair with the key `IWD_UCS_Error`.
18. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to 5 to retry the connection to the ESP server.
19. Exit InvokeUCS strategy
20. The DetermineESPServerName subroutine is invoked to determine the correct ESP server name to use. The subroutine uses the List Object list called ContactServerList. This subroutine also sets up cases when there is reason to retry to invoke the ESP server.
21. If the subroutine was successful, a check is done to ensure the existence of the ESPserver name that was returned by the subroutine.

22. An error is extracted from user data.
23. An error is extracted from user data.
24. The value of the user data key `IWD_isDone` is set to 0.
25. 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.
26. The value of the user data key `IWD_isDone` is set to 1.
27. The interaction is returned to its previous queue.
28. If the invocation of the method on the UCS fails, an error is extracted.
29. If it makes sense to retry updating the interaction record in UCS.
30. The last error code is attached to the interaction with the user data key `IWD_UCS_Error`.
31. A delay is introduced into the processing. Flow returns to step 20.
32. The last error is attached to user data as a key-value pair with the key `IWD_UCS_Error` when `in_method_name` is not set to `'Create'` or `in_method_name = 'OMInteractions'`.
33. The last error code is attached to the interaction with the user data key `IWD_UCS_Error`.
34. The interaction is placed in the `iWD_ErrorHeld` queue.
35. Exit InvokeUCS strategy.

Invoke GRE Strategy

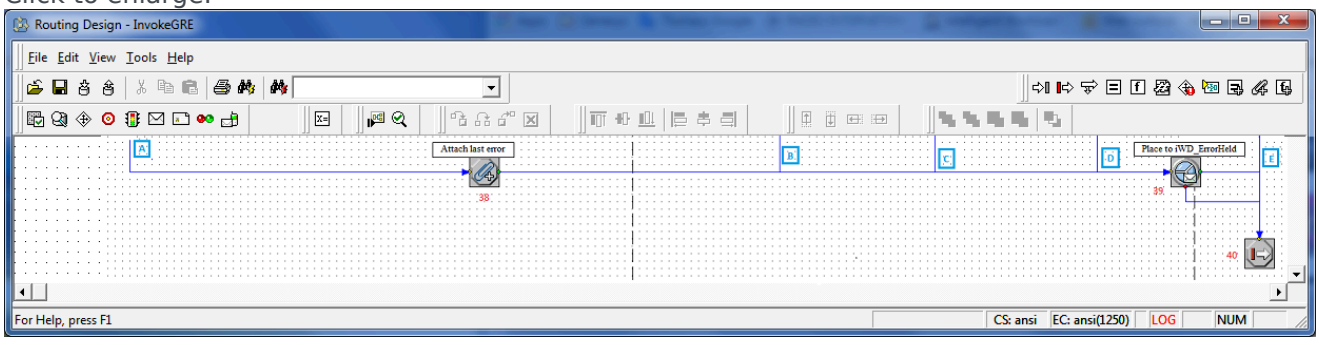
Part 1

Click to enlarge.



Part 2

Click to enlarge.



Flow Detail

1. Entry to InvokeGRE strategy.
2. Check if in_method_name is set to `SetBusinessContext` or `Prioritize`.
3. A variable is initialized—`_delay_ms` specifies the delay (in milliseconds) between attempts to invoke rules.

4. The DetermineESPServerName subroutine is invoked to determine the name of the Genesys Rules Engine Application. The subroutine uses the List Object list `GREServerList`.
5. If the subroutine was successful, a check is done to ensure the existence of the ESP server name that was returned by the subroutine. If the ESP server name was found, the flow goes to 10. The DetermineRulePackageName subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules.
6. If the ESP server name was not found, this error is attached to user data as a key-value pair with the key `IWD_GRE_Determination_Error`.
7. The interaction is placed in the `iWD_ErrorHeld` queue.
8. If the subroutine fails an error is extracted.
9. This error is attached to user data as a key-value pair with the key `IWD_GRE_Determination_Error`.
10. Check if `in_custom_package_name` was published to this subroutine. If it is set then `in_custom_package_name` will be run. Otherwise package name needs to be found in `Iwd_Package_List`.
11. Assign `in_custom_package_name` to `_gre_package_name` and set `_return_code` to `0`.
    11a. The DetermineRulePackageName subroutine is invoked to determine the name of the rule package that the Genesys Rules Engine will be invoking to evaluate the classification rules. If the rule package name was found, the flow goes to Step 16.
12. If the rule package name was found, the flow goes to Step 16.
13. If the rule package name was not found, this error is attached to user data as a key-value pair with the key `IWD_Rule_Package_Determination_Error`.
14. If the subroutine fails an error is extracted.
15. This error is attached to user data as a key-value pair with the key `IWD_Rule_Package_Determination_Error`.
16. Set the value for `RulePhase` and `IWD_GRE_Result` to empty string.
17. Check if `in_method_name` = `SetBusinessContext`.
    - If `in_method_name` is set to `SetBusinessContext` then the process calls classification rules in GRE.
    - If `in_method_name` is set to Prioritize then the process calls prioritization rules in GRE.
18. An ESP request is sent to the Genesys Rules Engine to evaluate the classification rules.
19. The ESP result is attached to user data as a key-value pair with the key `IWD_GRE_Result`.
20. Check if `IWD_reprioritizeDateTime` was changed by rules.
21. Delete `IWD_reprioritizeDateTime` from interactin if it was not changed by rules.
22. Invoke CheckBusinessValueAndPriority subroutine to verify if `IWD_businessValue` and `Priority` have correct values.
23. Check if `in_method_name` = `Prioritize`.
24. Check if `Priority` was changed by rules in `Prioritization`.
25. Attach `IWD_Prioritization_Error` to interaction with message `Priority is not set up by rules`.
26. The interaction is placed in the `iWD_Queued` queue.
27. An ESP request is sent to the Genesys Rules Engine to evaluate the prioritization rules.
28. The last Interaction Server-related error is extracted from a variable.

29. A check is done to see if the error code is related to the ESP server communication.
30. The last error is attached to user data as a key-value pair with the key `IWD_GRE_Error`.
31. Check if Department/Process is active.
32. Check if business process was assigned.
33. The interaction is placed in the `iWD_Captured` queue.
34. The last error is attached to user data as a key-value pair with the key `IWD_GRE_Error`. If not, the value of the `_counter` variable is incremented by 1.
35. A delay is introduced, based on the value of the `_delay_ms` variable. The flow goes back to 18 to retry the connection to the ESP server. The result from the ESP call to the Genesys Rules Engine is attached to the interaction as user data, with the key `IWD_GRE_Result`. This key-value pair will have the following format:

    ```
    return:ok| NumberOfRulesApplied:<number of applied rules>|RulesApplied:<rule 1 id>
    <rule1 name>, <rule2 id> <rule2 name>, ...
    ```

    The following example shows what the result might look like:

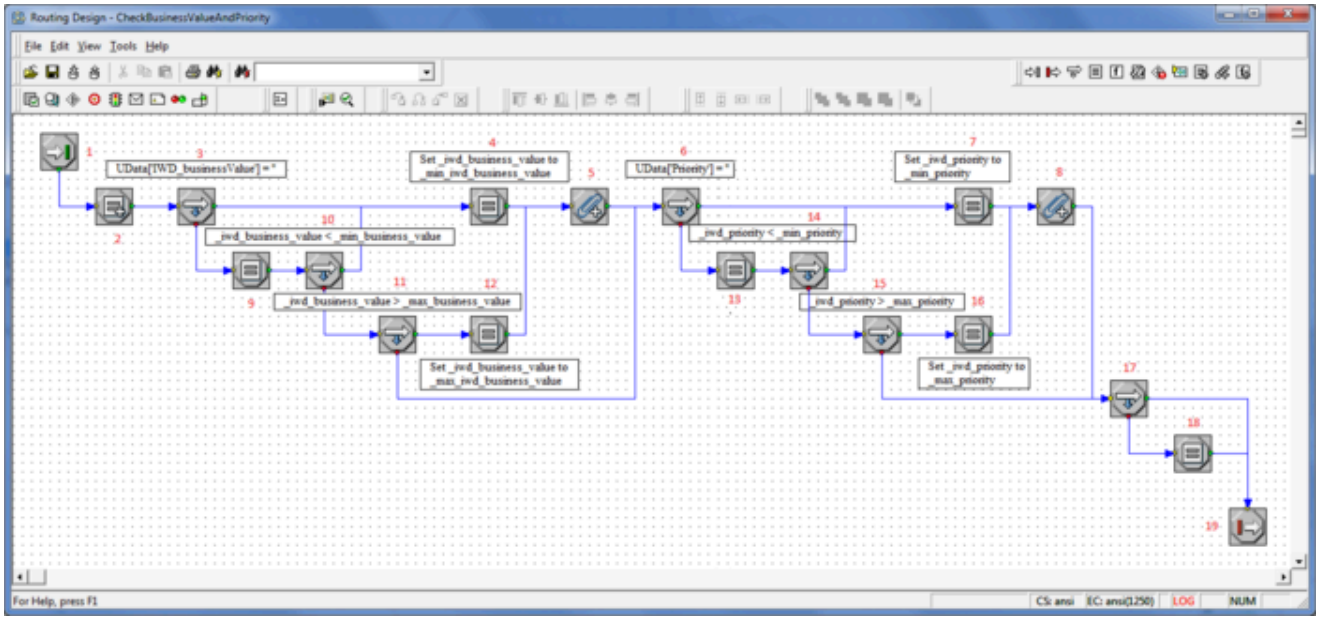
    ```
    AttributeUserData [list, size (unpacked)=168] = 'ESP_Result' [str] =
    "return:ok|NumberOfRulesApplied:12|RulesApplied:McrSlt1GlbClsf1McrSlt1GlbClassific
    ation1, McrSlt1GlbClsf2McrSlt1GlbClassification2"
    ```

36. The interaction is placed in the `iWD_Captured` queue.
37. Exit InvokeGRE subroutine.
38. The last error is attached to user data as a key-value pair with the key `IWD_GRE_Error` when `in_method_name` is not set to `SetBusinessContext` or `Prioritize`.
39. The interaction is placed in the `iWD_ErrorHeld` queue.
40. Exit InvokeGRE subroutine.

CheckBusinessValueAndPriority Subroutine

Flow Summary

Click to enlarge.



Flow Detail

1. Entry to the CheckBusinessValueAndPriority subroutine.
2. Variables are initialized:

- `out_return_code`—Return code returned by this subroutine at the exit.
- `_min_business_value`—Minimum available business value.
- `_max_business_value`—Maximum available business value.
- `_min_priority`—Minimum available priority.
- `_max_priority`—Maximum available priority.

1. Check if task has set `IWD_businessValue` attribute.
2. If `IWD_businessValue` attribute was not set then it will be set to `_min_business_value`.
3. `IWD_businessValue` attribute is attached to the task.
4. Check if task has set `Priority` attribute.
5. If `Priority` attribute was not set then it will be set to `_min_priority.`
6. `Priority` attribute is attached to the task.
7. Read `IWD_businessValue` attribute from the task and put it in `_iwd_business_value` variable.
8. Check if `_iwd_business_value` is leass than `_min_business_value`. If true then flow goes to 4.
9. Check if `_iwd_business_value` is greater than `_max_business_value`.
10. If `_iwd_business_value` is greater than `_max_business_value` then set `_iwd_business_value` to `_max_business_value`.

11. Read `Priority` attribute from the task and put it in `_iwd_priority` variable.
12. Check if `_iwd_priority` is less than `_min_priority`. If true then flow goes to 7.
13. Check if `_iwd_priority` is geater than `_max_priority`.
14. If `_iwd_priority` is reater than `_max_priority` then set `_iwd_priority` to `_max_priority`.
15. Check if `_iwd_business_value` and `_iwd_priority` are in range of allowed values.
16. If `_iwd_business_value` and `_iwd_priority` are out of scope then `out_return_code` will be set to 1.
17. Exit CheckBusinessValueAndPriority subroutine.

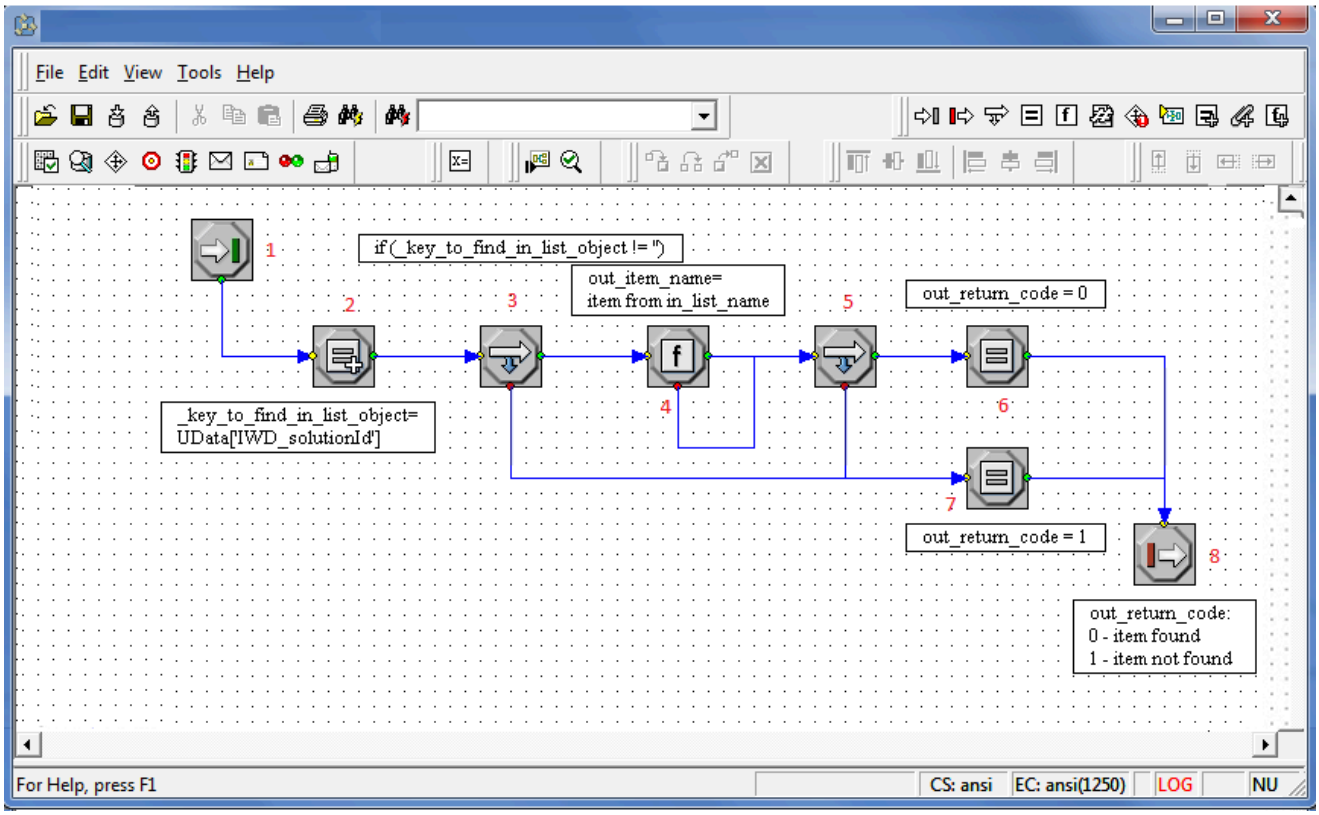
FindListObjectItem Subroutine

This subroutine amalgamates two subroutines that were previously separate in release 8.5.104:

- DetermineESPServerName
- DetermineRulePackageName

Flow Summary

Click to enlarge.



Flow Detail

1. Entry to FindListObjectItem subroutine.
2. Initialize variables:

- `_key_to_find_in_list_object`—Assign task IWD_SolutionId.
- `out_item_name`—Set its value to empty string.

1. Check if `_key_to_find_in_list_object` is not empty.
2. Search `_key_to_find_in_list_object` in `in_list_name`. Result will be assigned to `out_item_name`.
3. Check if `out_esp_name` is empty.
4. Set `out_return_code` to `0`.
5. Set `out_return_code` to 1.
6. Exit FindListObjectItem subroutine

Configuring List Objects

The iWD Business Process (IWDBP) uses two Configuration Server List Objects.

- The first List Object, `Iwd_Esp_List`, has two lists:
  - The first maps the iWD Solution ID to the name of the Genesys Rules Engine application.
  - The second maps the iWD Solution Runtime ID to the name of the Universal Contact Server (UCS) application. This is optional, and is used to allow the business logic in IWDBP to update the interaction record in the UCS database to mark the interaction as done (that is, the value of the `Status` column in the `Interaction` table will be set to 3) when it enters the `iWD_Completed`, `iWD_Rejected`, or `iWD_Canceled` queues.
- The second List Object, `Iwd_Package_List`, maps the iWD Solution ID to the rules package that will be evaluated when the Genesys Rules Engine is invoked from the IWDBP business process.

Both of these List Objects must be correctly configured for IWDBP to work.

One business process can serve several solutions under the same tenant. You need to create these two list objects during manual setup of IWD for your Solution IWD GAX Plug-in. In environments with only one solution, no further configuration needs to be done on the List Objects. If you have multiple solutions (or add one at a later time) these two List Objects need to be updated.

Iwd_Esp_List

GREServerList

The `GREServerList` list looks like a list of pairs:

`Solution_1`	`GREApplication_1`
`Solution_2`	`GREApplication_2`
`Solution_3`	`GREApplication_3`

Where the Solution ID is the key, and the name of the Genesys Rules Engine `Application` is the value.

ContactServerList

The `ContactServerList` list looks like a list of pairs:

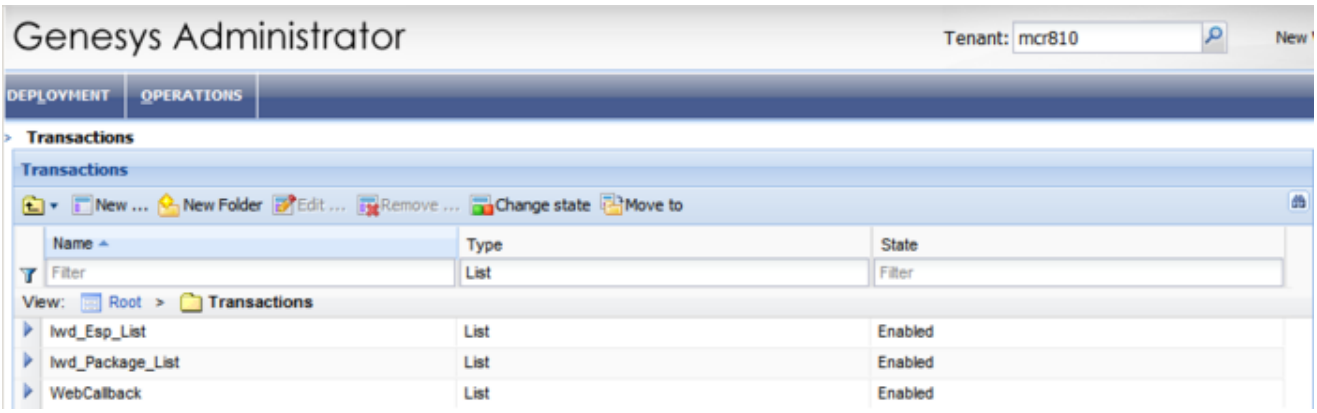
`iWD Solution Runtime_1`	`ContactServer_1`

iWD Solution Runtime_2	ContactServer_2
iWD Solution Runtime_3	ContactServer_3

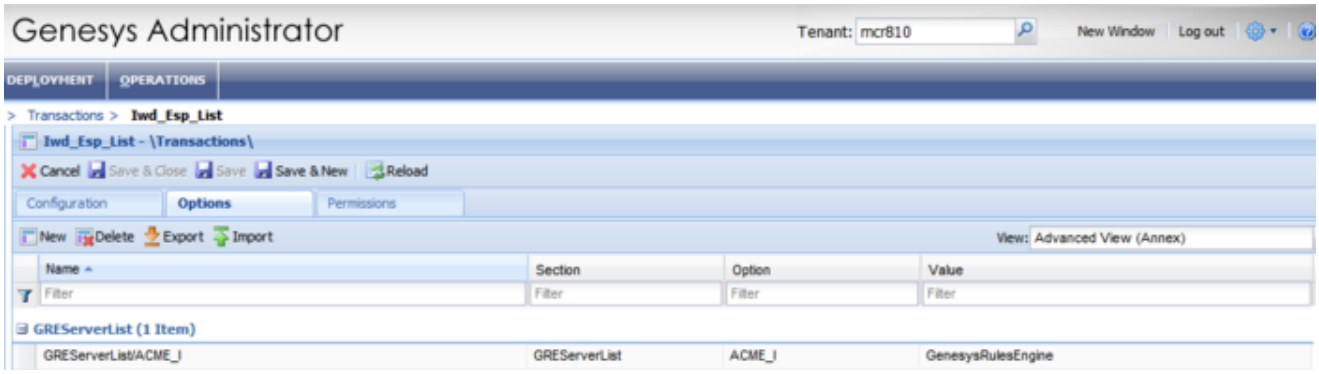
Where iWD Solution Runtime ID is the key and the name of a Universal Contact Server associated with Interaction Server is the value.

> **Important**
>
> It is very important that the pairs are set up correctly. If, for example, `Solution_1` is mapped to ESP`Service_2` instead of to ESP`Service_1`, business rules for `Solution_2` will be applied to all interactions which were submitted by Capture Points from `Solution_1`. Similar issues will occur if the Genesys Rules Engine application or the Universal Contact Server application are incorrectly mapped.

These key-value pairs in a List Object need to be set up only once per tenant, and can be configured in Interaction Routing Designer (IRD) or Genesys Administrator.



List Objects in IRD

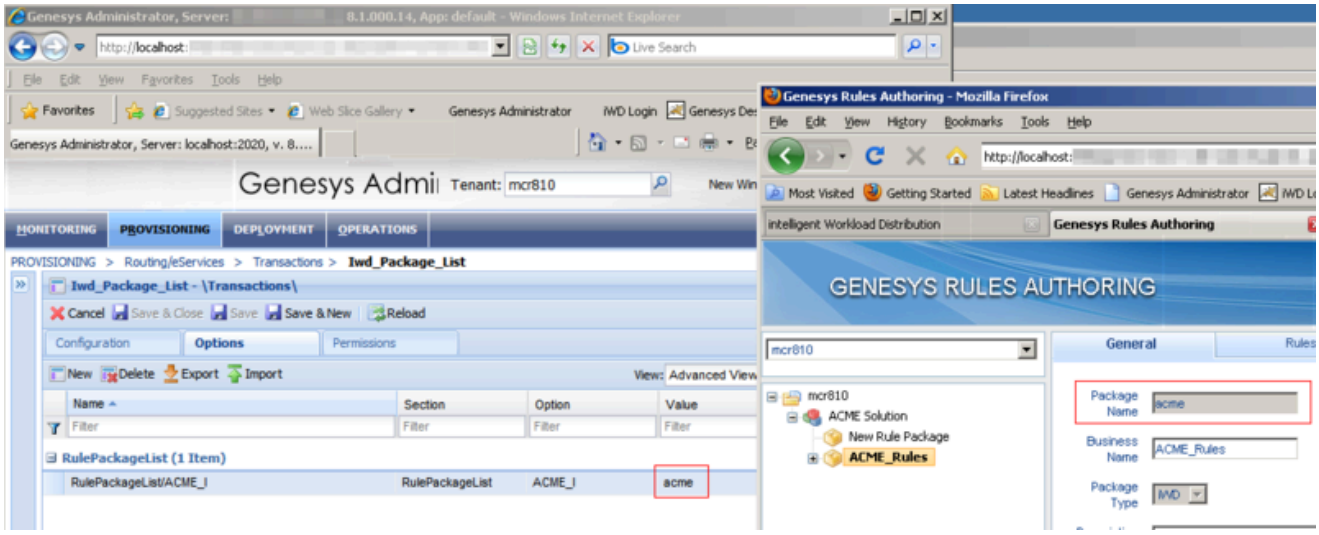


List Object Details

Iwd_Package_List

The `Iwd_Package_List` List Object is used to correlate the IWD Solution ID (`IWD_SolutionId`) to the name of the rule package that will be evaluated when requests are made to the Genesys Rules Engine from the IWDBP business process.

The `Iwd_Package_List` List Object contains a single list, `RulePackageList`. Create a new key/value pair for each iWD Solution that you have configured under your Configuration Server tenant, where the key or option is the IWD Solution ID and the value is the *Package Name* of the rules package.



iWD Package List

Cloning the IRD iWDBP to Create New Business Processes

For details on exporting objects from and importing objects to IRD, see **Universal Routing 8.1 Business Process User's Guide**.

To clone an iWDBP 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.

To clone the IWDBP, the high-level process is as follows:

1. Export the existing BP to a file.
2. Rename the BP.
3. Rename queues, views, submitters and strategies in the configuration database.
4. Rename queues in strategies and the BP.
5. Restore the original business process.

A detailed procedure is below.

> ### Warning
> If this procedure is not executed correctly, the BP is likely to break or be inconsistent.

Preparation

1. Open the IWDBP in Interaction Routing Designer (IRD).
2. Export IWDBP to a file, for example, `IWDBP_orig.wie`.
3. Deactivate all the strategies in the IWDBP.
4. Change the name of the IWDBP (for example, `IWDBP_custom`).
5. Close IRD.

Renaming objects

1. Open Configuration Manager.
2. For the IWDBP object (`[tenant]/Scripts/IWDBP`):
    1. Change the name to the same name as in IRD.
    2. Change `Annex/Namespace/Name` to the same new name.
3. For all iWD queues (`iWD_Canceled`, `iWD_Captured`, `iWD_Completed`, `iWD_ErrorHeld`, `iWD_New`, `iWD_Queued`, `iWD_Rejected`):
    1. Rename the script (for example, `iWD_Canceled_custom`)
    2. Change **Annex > Namespace > Name** to the same new name.
    3. Change **Annex > Namespace >BusinessProcess** to the new BP name (for example, `IWDBP_custom`).
4. For all iWD views (with names starting with queue names, for example `iWD_Canceled/Mark as Done`):
    1. Rename the script consistently with the corresponding queue (`iWD_Canceled_custom/Mark as Done`).
    2. Change **Annex > View> Queue** to the new name of the corresponding queue (for example, `iWD_Canceled_custom`).
5. For all iWD strategies (`Classification, Distribution, MarkInteractionAsDone, Prioritization, Removal`) and subroutines (`DetermineESPServerName, DetermineRulePackageName, CheckBusinessValueAndPriority`):
    1. Rename the script (for example, `Classification_custom`).
    2. Change all queue and subroutine names in **Annex > CFGScript** (for example, name: `0`, value: iWD_ErrorHeld_custom).
    3. Change all subroutine names in **Annex > Subroutines** (for example, name: `0`, value: DetermineESPServerName_custom).
    4. For all iWD submitters (with names starting with iWD views):
        1. Rename the script (`iWD_Canceled_custom/Mark as Done/-/MarkInteractionAsDone_custom`).
        2. Change **Annex > Submitter >Strategy** consistently with the corresponding strategy (for example, `MarkInteractionAsDone_custom`).
        3. Change **Annex > Submitter >View** consistently with the corresponding view (for example, `iWD_Canceled_custom/Mark as Done`).
    5. For the BPscript:
        1. Change all the iWD queues in **Annex > Queues** to the new names.
        2. Change all the iWD strategies in **Annex > Strategies** to the new names.
        3. Change all the iWD submitters in **Annex > Submitters** to the new names.
        4. Change all the iWD views in **Annex > Views** to the new names.

Modifying iWD Routing Strategies

1. For all iWD routing strategies:
    1. Open the strategy in IRD.
    2. For all **Call subroutine properties** objects edit the `Subroutine name` property and set it to the

appropriate new subroutine name. Remember to properly set `Input parameters` and `Output parameters`.

3. For all **Queue Interaction** objects edit the `Interaction Queue` property and set it to the appropriate new queue name.
4. (Distribution strategy only) For all **Route Interaction** objects edit the `Interaction Queue/Queue for Existing Interaction/Queues` property and set it to the new name for the `iWD_Completed` queue (that is, `iWD_Completed_custom`).
5. Save the strategy using the new strategy name. Choose the original **RBN File Path** where the BP was imported.

2. Save all iWD subroutines using the new subroutine name. Choose original **RBN File Path** where the BP was imported.
3. Save the BP.

Restoring the original business process

1. In IRD import the previously exported file (`IWDBP_orig.wie`).
2. Activate the strategies in both business processes.

> **Important**
>
> Subroutines can be shared between business processes until there is a need to change them separately.

Additional Configuration

Interaction Queues

iWD recognizes seven interaction queues. By default they are created by the standard iWD business process (IWDBP) and have the following names in the IWD Business Process for IRD:

- `iWD_New`
- `iWD_Captured`
- `iWD_Queued`
- `iWD_Completed`
- `iWD_Rejected`
- `iWD_Canceled`
- `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.

Open the Business Structure and navigate to the Solution in the iWD GAX Plug-in and complete the complete the iWD-related Interaction Custom Properties.

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_Completed` or `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:

- `Canceled`
- `Captured`
- `Completed`
- `ErrorHeld`
- `New`
- `Queued`
- `Rejected`

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 in IRD
default-values/Queue	New	iWD_New
RestartQueues	New	iWD_New
CompleteQueues	Completed	iWD_Completed
RejectQueues	Rejected	iWD_Rejected
CancelQueues	Canceled	iWD_Canceled

Capture Point Option	iWD Solution's Queue	Default Value in IRD
ErrorHeldQueues	Error Held	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.

Modifying the iWDBP

For most environments, the only modification that will need to be made to the iWD Business Process 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_Queued/All` view to the `Distribution` routing strategy with a connection from `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_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_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_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_Rejected`
  - `iWD_Canceled`
  - `iWD_Completed`

or their equivalents defined in the Solution configuration.