iWD Deployment Guide

Task Distribution and Routing

5/11/2025

Task Distribution and Routing

Contents

This section includes information about best practices to consider when you are planning and configuring the task distribution and routing components of your iWD solution.

Use Scheduling in the Queue Views

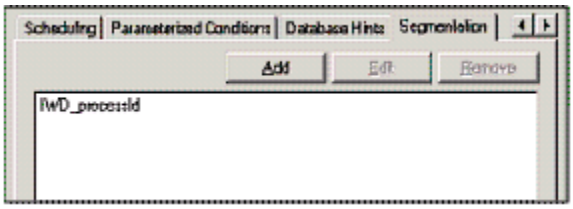
Queue views define the criteria that must be met for a task to be submitted from a queue to a Distribution routing strategy. Using the **Scheduling** tab can prevent a task from bouncing between Interaction Server and Universal Routing Server (URS), especially when there are no agents logged in to handle tasks. In this case, the **ScheduledAt** attribute can be used to reschedule submission of tasks back to URS.

> ### Important
> For information about the **ScheduledAt** property, see the topic “Setting the ScheduledAt Property” in the *Universal Routing 8.1 Business Process User's Guide.*

Scheduling Tab—URS

Use Segmented Views to Keep Agents Busy

Consider using segmentation on the **Queue** view from the iWD_Queued queue (or custom interaction queue in your iWD business process) to the **Distribution** routing strategy. Segmenting interactions, based on iWD Process (IWD_processId), or by skill might be a good idea. This ensures that all types of work (tasks) will be submitted for distribution even if these tasks (in a specific process) have low priorities. There might be specific agents that are dedicated to manage or handle these low priority tasks. However, if they have low priorities, they might never be submitted to a distribution strategy.



Segmentation Tab

Segmentation Feature

Segmentation ensures that all types of work (tasks) will be submitted for distribution even if the tasks (in a specific process) have low priorities. Segmentation also ensures that all agents are busy by distributing tasks in each segment separately, thus reducing agents' idle time. Each agent group can have its own task queue, and each segment can have its own limit.

Segmentation settings appear in the **ToDistribute** view in the **Distribution** strategy— now, the **Distribution** strategy can identify an interaction attribute (**IWD_Segment**) in those settings and use its value to route the interaction to the correct agent/agent group.

There is more information in these documents:

- IWD BP for IRD
- IWDBP for Composer/ORS

Scheduling will not be used in this example.

Use Triggers in the Routing Strategy

There are times when a task is updated by the source system—for example, an agent who is not iWD-enabled has modified the task or an external system has modified the task. This task update might impact the classification, prioritization, or distribution of the task. Setting triggers on the **Distribution** strategy to react to specific changes in attached data elements enables you to reclassify, reprioritize, and redistribute tasks, as required.

There is a function in IRD/URS 8.x called **SetUpdateTrigger[]** which enables you to specify an attached data key that will be monitored for dynamic changes while an interaction is sitting in the **Target** block, waiting for an available agent.

Example: Specifying Attached Data Key

An interaction is waiting for an available agent in a **Target** block in the **Distribution** strategy, with a timeout of `30` minutes before it goes to the red port. During this time, it possible that the source system will send an **UpdateTask** message (for example, if another agent pulled the task manually from the source system). If the **SetUpdateTrigger[]** function is enabled in the strategy for a specific attached data element, and then the value is updated, the interaction goes to the red port automatically, enabling you to evaluate the **UpdateTask** message and take control of the interaction in the strategy.

If **SetUpdateTrigger[]** is not set up in the **Distribution** routing strategy to handle this, then the interaction might be stuck in the **Target** block and could get distributed to other agents, which could cause a loss of synchronization between the source system and IWD. A recommended approach is to create a custom attribute such as **iwdAction** in the **SetUpdateTrigger[]** function, with possible values of `CreateTask`, `UpdateTask`, and so on. This attribute would be set by the source system in any `CreateTask` or `UpdateTask` messages that would be generated from the source system. After evaluating the message within the IWDBP business process, its value can be set to `CLEARED`.

Apart from using the **iwdAction** custom attribute in the **SetUpdateTrigger[]** function, this attribute can also be used in most of the **Interaction Queue** or **Workbin** views to determine if the task has been updated (by using the `UpdateTask` message) while sitting in those Queues or Workbins. This can be done by creating a new View from these Queues and Workbins called, for example, `BackEnd Update`, with this condition:

```
iwdAction !='CLEARED'
```

If this condition is met, the interaction can be submitted to a routing strategy that evaluates the update task message and performs the appropriate actions, such as distributing it to a specific agent, applying rules, or ignoring the update and sending it back its original location.

Avoid Looping in Strategies

It is important to prevent looping within your routing strategies. It can place a lot of load on the Genesys Rules Engine (GRE) if reclassification and reprioritisation are occurring over and over again. If these requests fail because GRE is down, or a rule is not applied successfully due to a syntax error that cannot be caught during design time, consider taking one of the following actions:

- Use scheduling to delay the next attempt at rules evaluation.
- Place the task into the iWD_ErrorHeld queue (IRD business process) or iwd_bp_comp.Main.iWD_ErrorHeld (Composer business process) to stop a *bad* task from impacting the overall system. For example, you might check the value of the **IWD_processId** attribute to verify that it has been classified correctly—that is, it is not `NULL`. From the iWD_ErrorHeld queue or iwd_bp_comp.Main.iWD_ErrorHeld queue you might resubmit the task into an error-handling strategy, which then sends the task back to the iWD_New queue or iwd_bp_comp.Main.iWD_New queue or performs other integrity checking on the task's attached data.

Consider Pull Versus Push Task Distribution

Many businesses like iWD's push-based model of task distribution. It helps avoid the common problem of employees picking the easiest task to work on next, and to ensure that employees are always working on the highest-priority task. However, it is likely that some business workflows will require a pull (workbin) model. So consider the role that Agent Group Workbins might play in your solution.

Also consider that agents might need to hold on to a task for a period of time if they cannot complete it immediately. This might require the agent to open and close the task multiple times before finally completing, or otherwise dispositioning, the task. In this case, you will probably want to use personal Agent Workbins. In both cases, you will need to consider the reporting and distribution implications.