WOKFLOW FOUNDATION MEISTER 2.0

MEISTER V2.0 ORCHESTRATION ENGINE

Andre Rosenthal
Gateway Architects, LLC
2600 Dallas Parkway suite # 600, Frisco, TX

Table of Contents

Product Document

Document Name:	WFM with Meister 2.0
Document ID:	MV2MWF0001
Document Owner:	Andre Rosenthal
Document Version:	1.0
Document Date:	Saturday, 24 August 2019
Document Status:	Draft

Introduction

Workflow enablement is accomplished via Meister's Workflow Foundation support. In its own implementation, Meister provides the toolkit for enablement of an existing SAP workflow as an endpoint accessible by any UX via the SDK website.

MWF Integration

We will borrow yet again our mapping function defined at the Integration with Meister 2.0 document:

$$g\left(\sum f_i\,(x)\right) \mid \exists\, x\ in\ X\ and\ y\ in\ Y \rightarrow\ \exists\ map\ M(f,g)\ defining\ a\ transition\ from\ t\ to\ g.$$

We called this function $g()$ before as the Integrator function, or in simple form, the orchestration engine behind the mapping of discrete data from different systems. We could therefore create another function as above such that the transition portion would be from an endpoint for Decision steps of any existing SAP workflow. The new transition function would receive a workflow item number as part of the Request envelope and a decision field to inform the Workflow of the user decision, which normally is Acceptance or Rejection, and for the latter, a reason for the rejection.

Injection of Meister into existing workflows

Meister 2.0 provides a class to be added as part of the header of the workflow definition at SAP. Once injected into the header of the workflow, all the events fired by SAP to the runtime instance of the workflow definition are trapped by Meister and provided as part of the development effort for specific needs. By default, six discrete events are trapped by Meister and made available as BADI extensions that are easily enhanced by the ABAP resources. These events are welcomed as part of the definition set provided by Meister for the sake of visibility of BUS unique support. Additionally, the ABAP develop may require to be send to the UX layer other fields from different sources (either SAP or not) and aggregated to the Response envelope as the final set of execution. In these cases, it is advisable for the outside-in designer of the Json document to receive a simple string as Response to support polymorphism. The usage of the injection process is independent of the support for User Decision Tasks. The former allows the developer to extract deep information from the objects before a decision takes place, and the latter supports the execution of the decision task by itself. As such, it is possible to use the Decision step by itself when the existing information suffice for the sound

execution of said decision. In other cases, where a deep crawl is required by the business, the Injection mode is the key to provide a rich real-time hook for up to date information of the underlying objects being driven by the workflow definition in used.

The decision steps

Meister exposes an endpoint named Meister.MWF.Decider to execute any decision task of workflows regardless of the fact that Meister was injected or not to the workflow definition file. As more than one decision may be required (such as release approval chains) the workflow itself should be responsible for the sound execution of the release chain and Meister is solely responsible for the atomic execution of decision steps. Given that, the rejection of one participant of the release chain should invalidate the approval of previous decision steps and terminate the workflow work item. There is no support for release chain management inside Meister itself, being that the implementation of the logic of the release chain unique for customers as part of their customization steps on SAP.