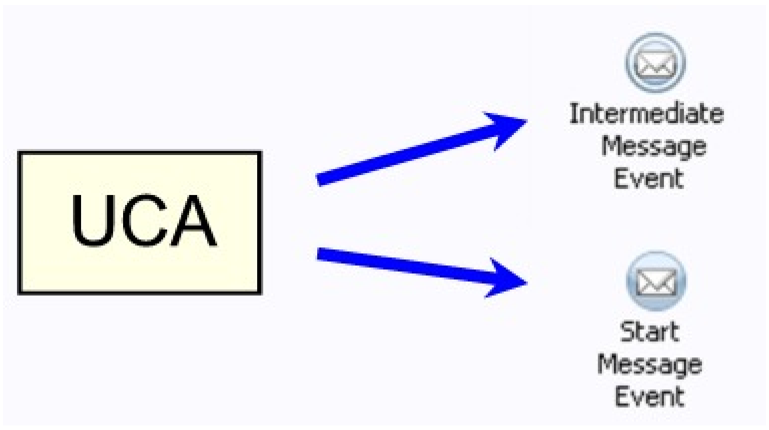
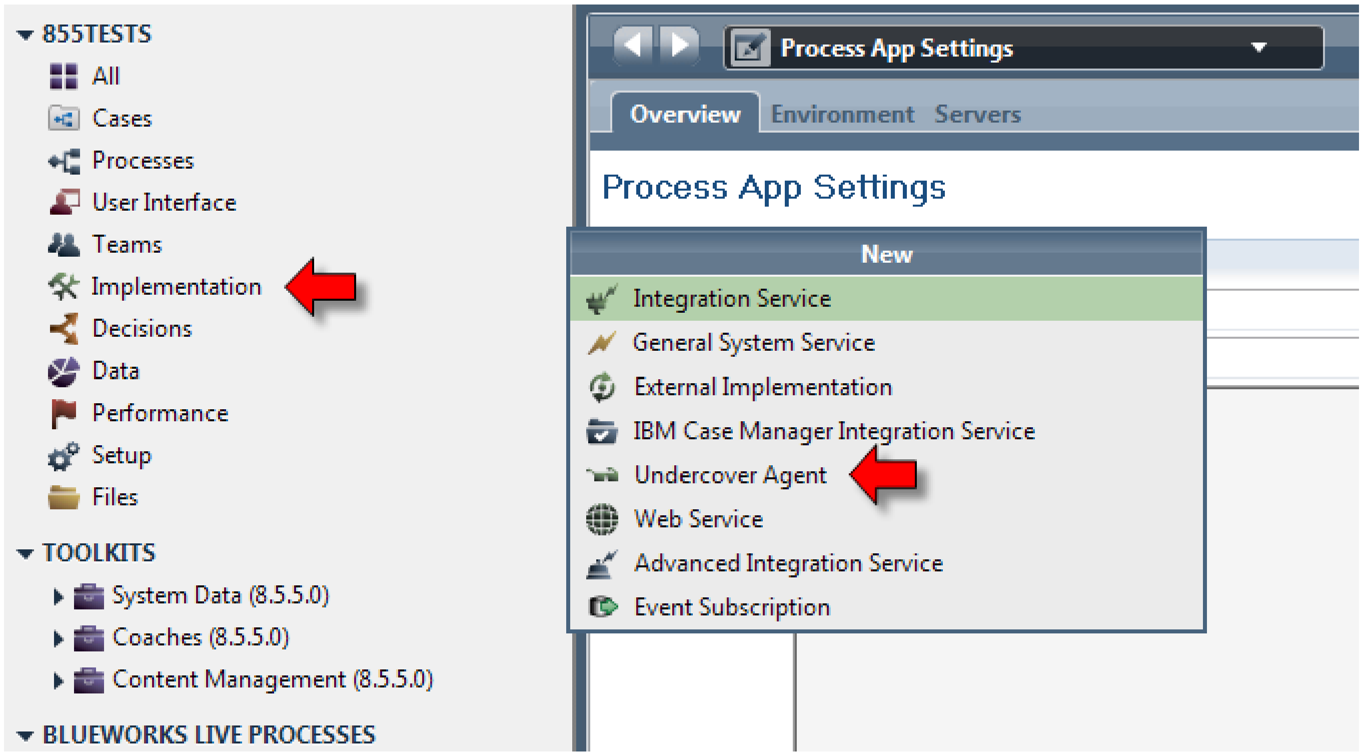
Undercover Agents (UCAs)

An Undercover Agent (UCA) listens for triggers from external systems, explicitly coded Invoke UCA service components or for scheduled time based events. UCAs are also responsible for starting a process instance through a Message Start Event or for waking up a Message Intermediate Event that is contained within a BPD.

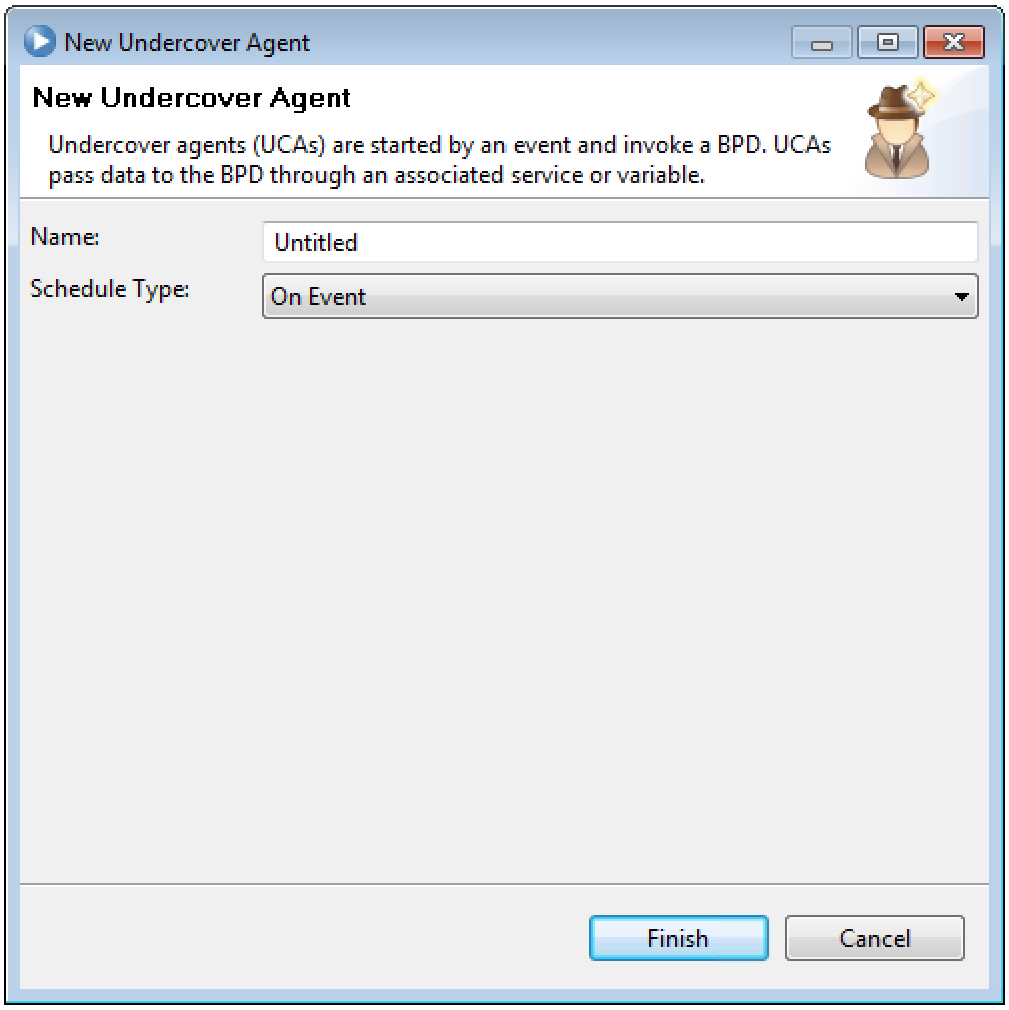
Note: The term UCA (Under Cover Agent) is a particularly poor name for this concept. In actuality, what we have here is BPM event processing. There are many alternative names that could have been used for UCA. Possibilities might have included "Event". When you read the phrase UCA, try and think of it as an event. It is hoped that somewhere in the future, the phrase "UCA" will become a historic footnote and be replaced with something more consistent with its intent.



A UCA definition is created from the implementation category within Process Designer.



Once selected, a wizard will appear. In this wizard we get to give the new UCA a name and define how it is started.


New Undercover Agent

New Undercover Agent

Undercover agents (UCAs) are started by an event and invoke a BPD. UCAs pass data to the BPD through an associated service or variable.

Name: Untitled

Schedule Type: On Event

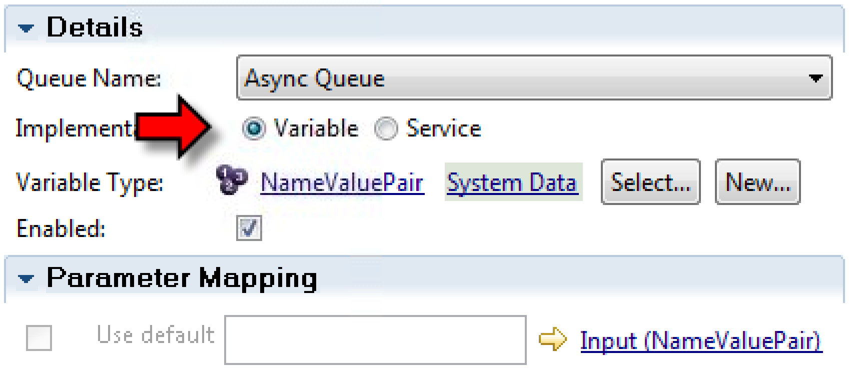
The "`Schedule Type`" attribute has two possible options:

- `On Event` – Fired as a result of an explicit Invoke UCA in a service
- `Time Elapsed` – Fired on schedule from the run-time

Event Initiated UCAs

When an Event Initiated UCA is defined, it looks as follows:

When a UCA is activated, it will basically represent an event. That event can carry payload data with it. IBPM provides two distinct mechanisms to define the payload data that may be carried with a UCA event. The simplest way is to select "variable".


Details

Queue Name: Async Queue

Implement: [✓ Variable] Service

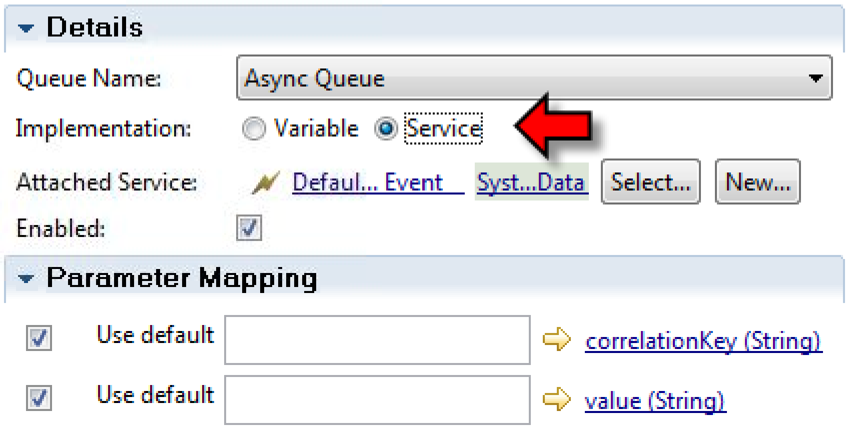
Variable Type: NameValuePair System Data Select... New...

Enabled: ✓

Parameter Mapping

Use default Input (NameValuePair)

With that radio button checked, we get to supply a single variable data type. When a UCA is invoked, it will expect an instance of that type of data to be passed along with it. The second option is available when `Service` is selected:


Details

Queue Name: Async Queue

Implementation: Variable [✓ Service]

Attached Service: Defaul... Event Syst...Data Select... New...

Enabled: ✓

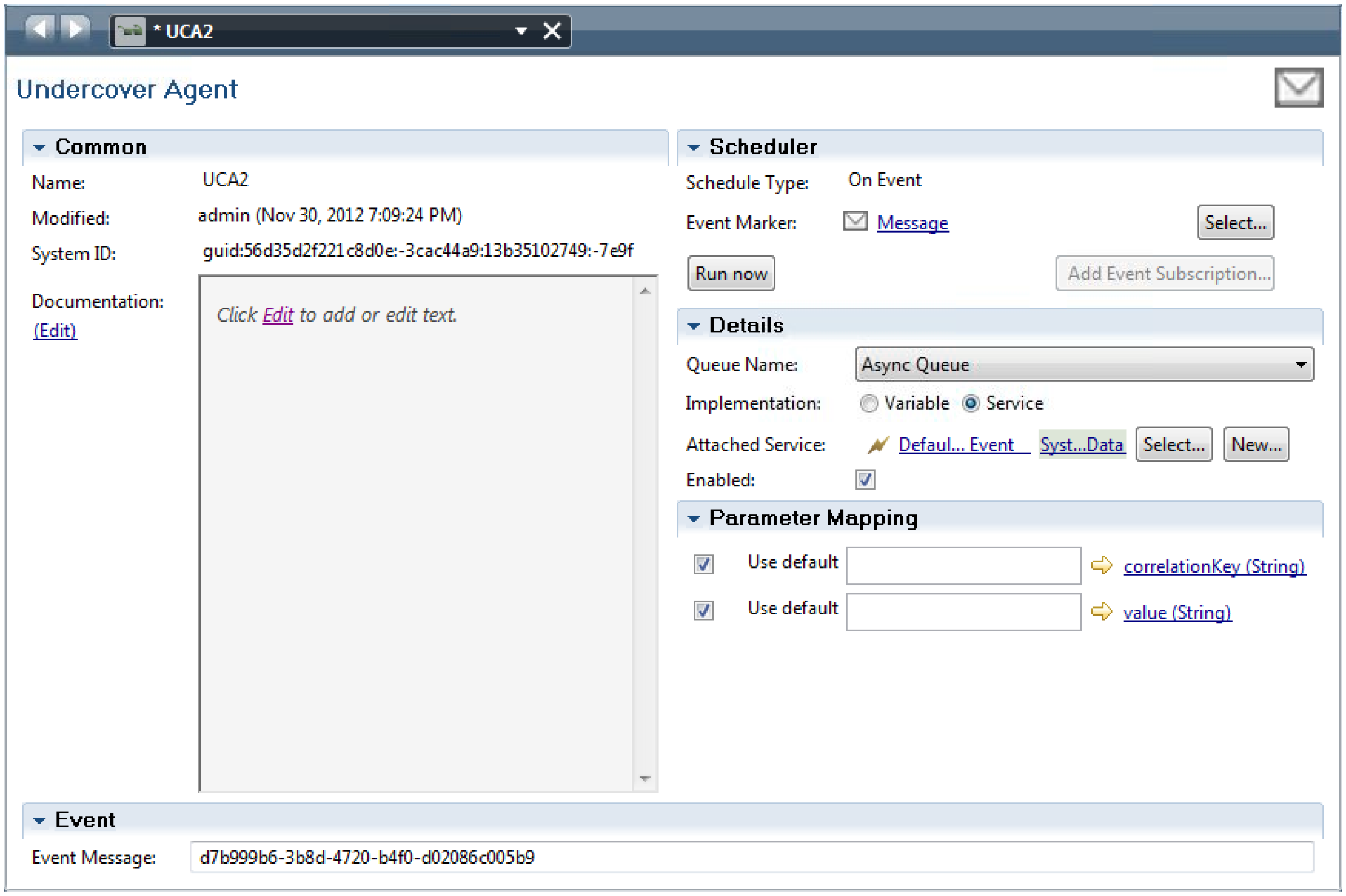
Parameter Mapping

✓ Use default correlationKey (String)

✓ Use default value (String)

Now we get to supply the reference to a service (commonly a general service definition). The input data to the general service is then used to build some output data and the output data is what is passed along with the UCA. It is extremely common to have a general service map directly from the input parameters to the output parameters without modification. This allows a UCA to pass more than one parameter.

Once created, the UCA definition will look as follows:



Undercover Agent

Common

Name:	UCA2
Modified:	admin (Nov 30, 2012 7:09:24 PM)
System ID:	guid:56d35d2f221c8d0e:-3cac44a9:13b35102749:-7e9f
Documentation: (Edit)	Click Edit to add or edit text.

Scheduler

Schedule Type:	On Event
Event Marker:	Message

Details

Queue Name:	Async Queue
Implementation:	Variable [✓ Service]
Attached Service:	Defaul... Event Syst...Data
Enabled:	✓

Parameter Mapping

✓	Use default		correlationKey (String)
✓	Use default		value (String)

Event

Event Message:	d7b999b6-3b8d-4720-b4f0-d02086c005b9

The `Enabled` check box can be used to enable or disable the execution of the associated service when an event arrives.

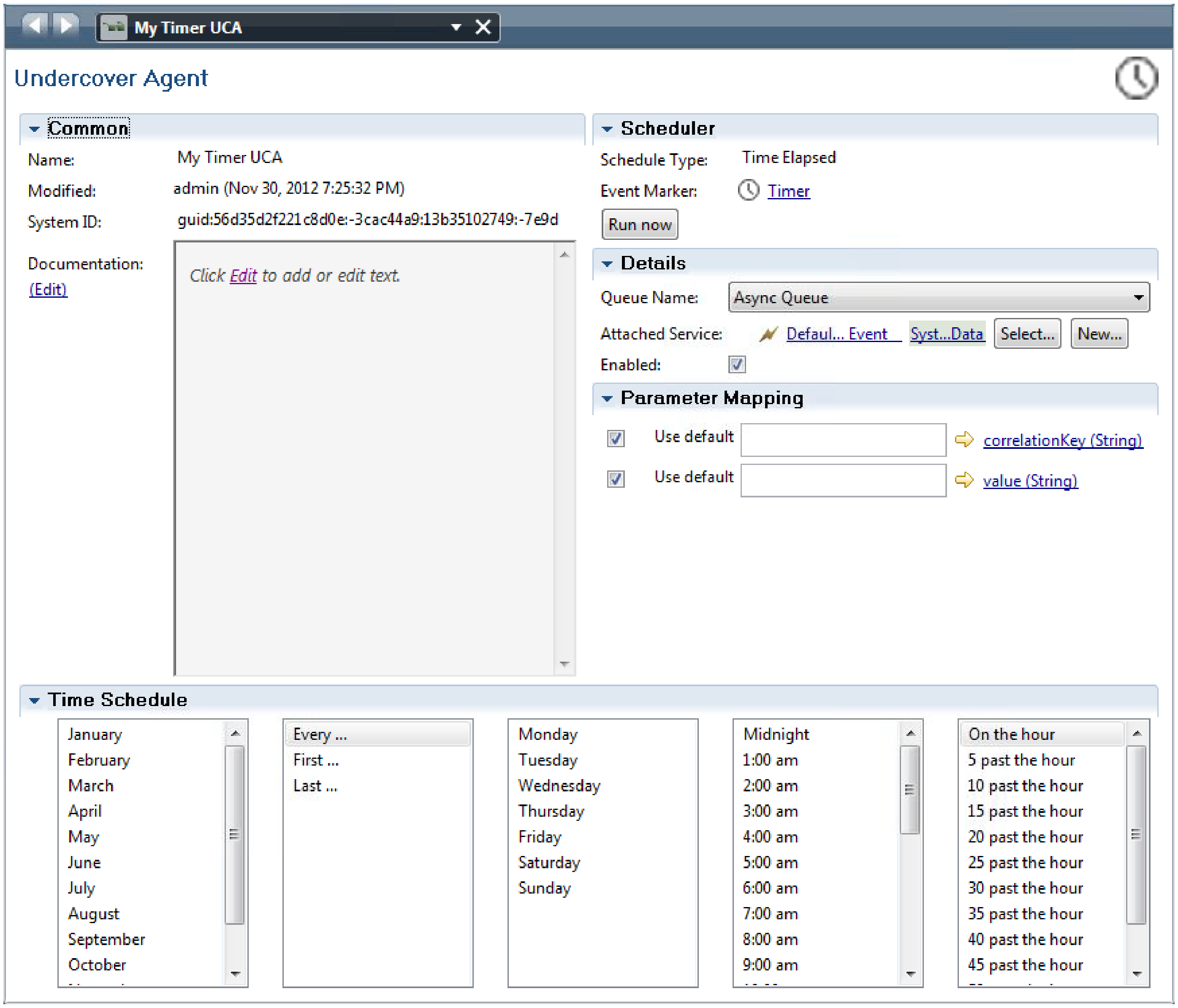
In the `Event` section, we see a correlation value called `Event Message` that is used to associate an incoming event with the instance of a UCA that is watching for that kind of events. The majority of times this can be ignored. This is of particular importance when using JMS to initiate a UCA. See: Java Message Service – JMS.

When an event arrives and triggers a UCA, the UCA in turn will execute an IBPM service. This is specified by the `Attached Service` attribute and is commonly an IBPM General Service. The way that the UCA causes the service to be called is not by invoking that service explicitly, instead, the UCA queues the request to start the service on a logical queue. It must be stressed that this is indeed a logical queue and has no relationship to JMS or SI Bus. The queuing mechanism here is used to serialize the execution of UCA originated events. When the UCA places a request on a queue, any further requests placed on that queue are not processed until the previous events have completed which means that the associated services have completed. One exception to this is the special queue called the `Async Queue`. If the UCA start service request is placed on this queue, further requests to start services will **not** wait for previous services to complete.

The number and names of these queues can be changed in the IBPM Process Admin Console.

Schedule initiated UCAs

A schedule (time based) UCA has an additional set of attributes:


My Timer UCA

Undercover Agent

Common

Name: My Timer UCA

Modified: admin (Nov 30, 2012 7:25:32 PM)

System ID: guid:56d35d2f221c8d0e:-3cac44a9:13b35102749:-7e9d

Documentation: (Edit) Click Edit to add or edit text.

Scheduler

Schedule Type: Time Elapsed

Event Marker: Timer

Run now

Details

Queue Name: Async Queue

Attached Service: Defaul... Event Syst...Data Select... New...

Enabled: ✓

Parameter Mapping

✓ Use default → correlationKey (String)

✓ Use default → value (String)

Time Schedule

January	Every ...	Monday	Midnight	On the hour
February	First ...	Tuesday	1:00 am	5 past the hour
March	Last ...	Wednesday	2:00 am	10 past the hour
April		Thursday	3:00 am	15 past the hour
May		Friday	4:00 am	20 past the hour
June		Saturday	5:00 am	25 past the hour
July		Sunday	6:00 am	30 past the hour
August			7:00 am	35 past the hour
September			8:00 am	40 past the hour
October			9:00 am	45 past the hour

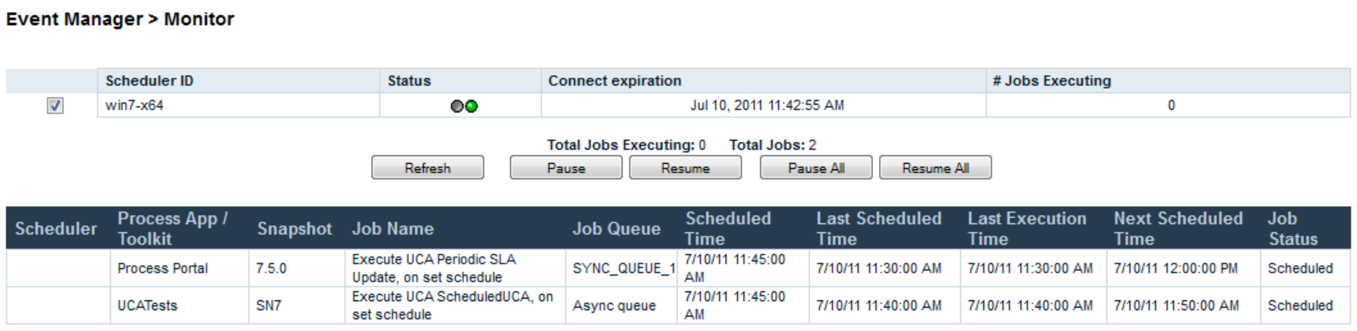
The Time Schedule defines recurring periods of when the UCA should fire. The Time Schedule is quite flexible in its configuration. Each time the schedule is reached, the UCA fires.

The icon for a Timer Event UCA looks as follows:

My Timer UCA

Within Process Center there is a Process Server used for testing. It doesn't make much (if any) sense for a UCA to fire for each of the schedules held in the repository in the Process Center Server. As such, they simply don't. For testing, the `Run now` button can be used to explicitly fire an instance of the UCA.

When a UCA is scheduled, we can see it in the Process Admin Console under Event Manager > Monitor:


Event Manager > Monitor

	Scheduler ID	Status	Connect expiration	# Jobs Executing
☑	win7-x64	●●	Jul 10, 2011 11:42:55 AM	0

Total Jobs Executing: 0 **Total Jobs:** 2

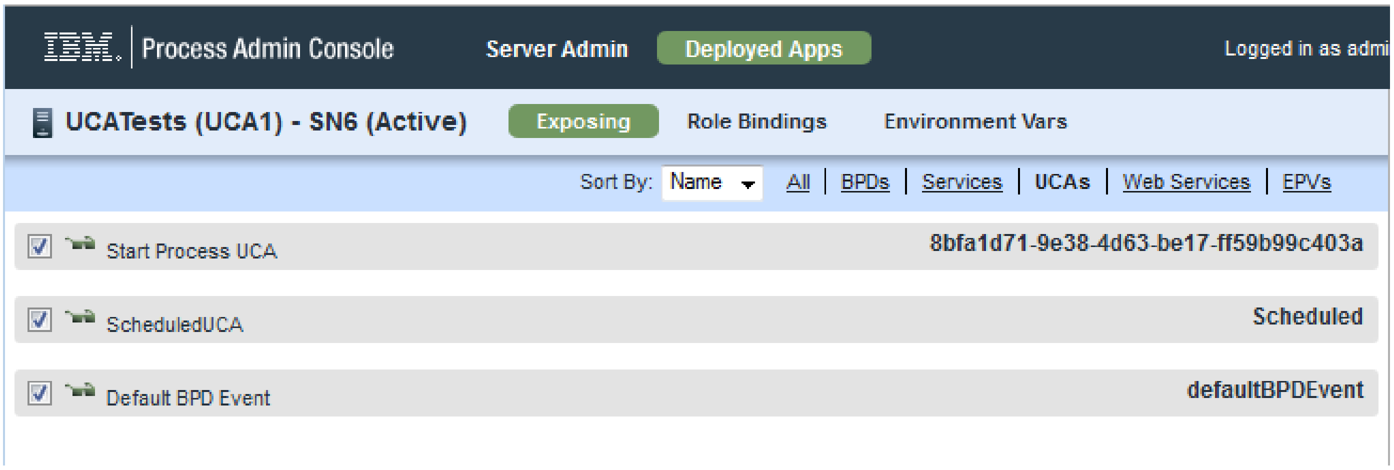
Scheduler	Process App / Toolkit	Snapshot	Job Name	Job Queue	Scheduled Time	Last Scheduled Time	Last Execution Time	Next Scheduled Time	Job Status
	Process Portal	7.5.0	Execute UCA Periodic SLA Update, on set schedule	SYNC_QUEUE_1	7/10/11 11:45:00 AM	7/10/11 11:30:00 AM	7/10/11 11:30:00 AM	7/10/11 12:00:00 PM	Scheduled
	UCATests	SN7	Execute UCA ScheduledUCA, on set schedule	Async queue	7/10/11 11:45:00 AM	7/10/11 11:40:00 AM	7/10/11 11:40:00 AM	7/10/11 11:50:00 AM	Scheduled

UCAs and queued events

Consider a UCA firing which results in an event being generated for processing by either a Start Message Event or an Intermediate Message Event.

Disabling UCA processing

Within the Process Admin Console under the Deployed Apps section, we can see lists of the UCAs associated with an application.



UCATests (UCA1) - SN6 (Active)

☑ Start Process UCA	8bfa1d71-9e38-4d63-be17-ff59b99c403a
☑ ScheduledUCA	Scheduled
☑ Default BPD Event	defaultBPDEvent

From here we can selectively enabled or disable their processing.

UCAs and Toolkits

One common usage pattern for UCAs is to have them used in a publish/subscribe model. This means that a publisher publishes an event by invoking a UCA. Any interested subscribers then listen on that UCA to be triggered if and when a UCA is invoked. The implication of this is that both the publisher and subscriber must share the UCA definition. If the publisher and subscriber are in different Process Applications then the UCA definition must be stored in a toolkit that is common to both Process Applications. Care should be taken that UCAs added to Toolkits be **only** event processing UCAs and should **not** be time based UCAs. This technique is a powerful solution to having one BPD invoke another BPD when both BPDs are in different process apps. There is no known alternate solution to that puzzle.

Revisión #1

Creado el hace 2 años por Admin

Actualizado el hace 2 años por Admin