Understanding Components

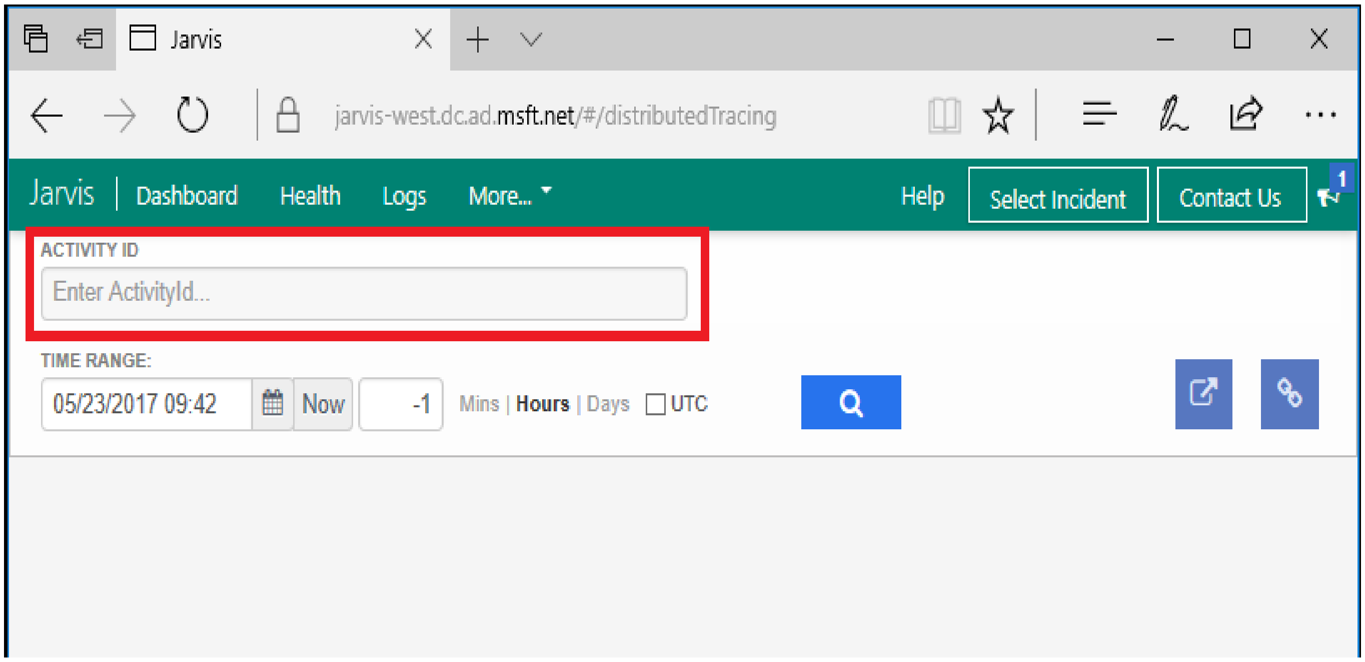
Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:41 PM PDT

Contents

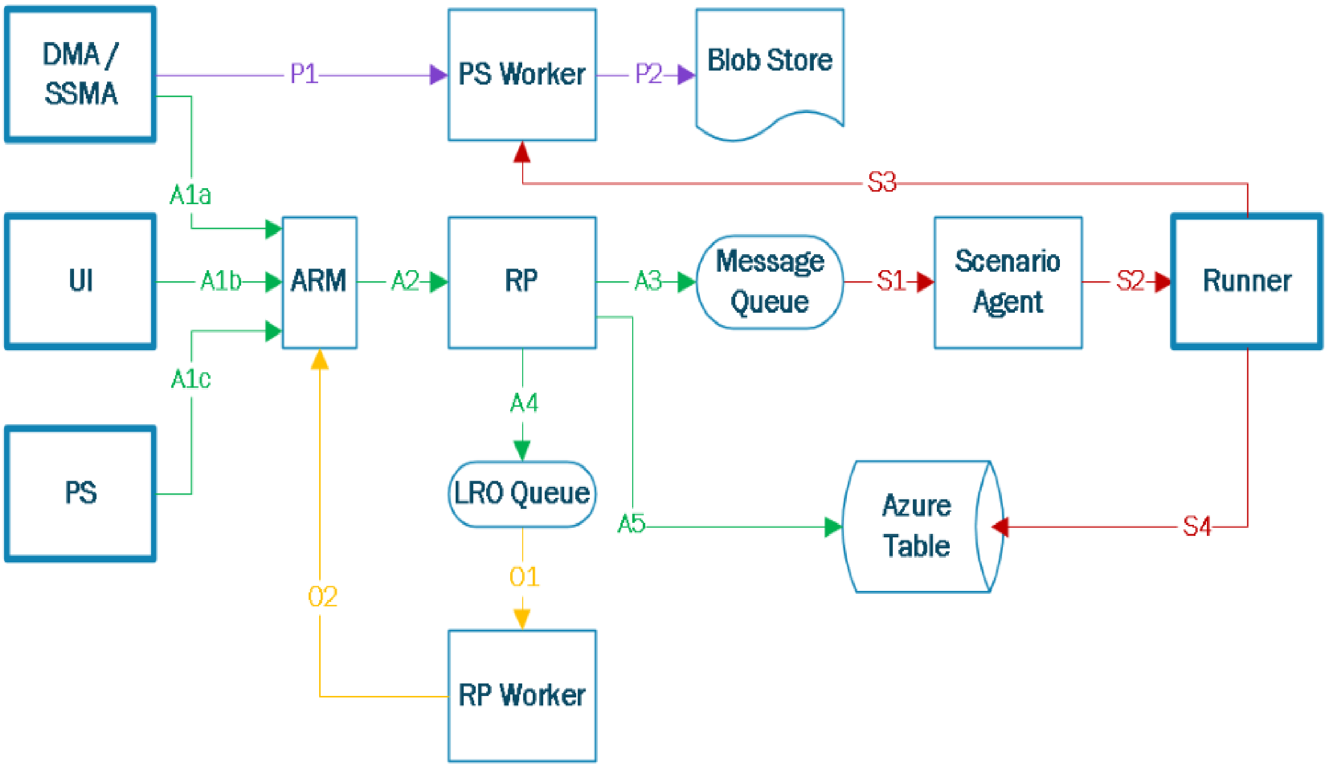
Goal:

Track down individual requests or activity in **DMS** (and possibly **SSMA** and **DMA**) using a unique id across machines, roles, tenants and other boundaries. Consider all the clients such as DMS UI, DMS PS, SSMA/DMA.

Example: Track down execution of RunMigrationCL scenario (from user starting the scenario till the scenario completion).


Jarvis
jarvis-west.dc.ad.msft.net/#/distributedTracing
Jarvis | Dashboard Health Logs More...
Help Select Incident Contact Us
ACTIVITY ID
Enter ActivityId...
TIME RANGE:
05/23/2017 09:42
Now
-1
Mins | Hours | Days
UTC

Component view



Project Loop:
P1) Request to Download/Upload project
P2) Get/Set project file from blob store

RP Loop:
A1a) GetUrl for blob store to Download/Upload project
A1b) Provision service, start task, get result, etc.
A1c) Provision service, start task, etc.
A2) Request RP to schedule work and return within 15s.
A3) Queue scenario/task execution.
A4) Queue LRO execution (provision service, delete service, etc.).
A5) Get scenario result

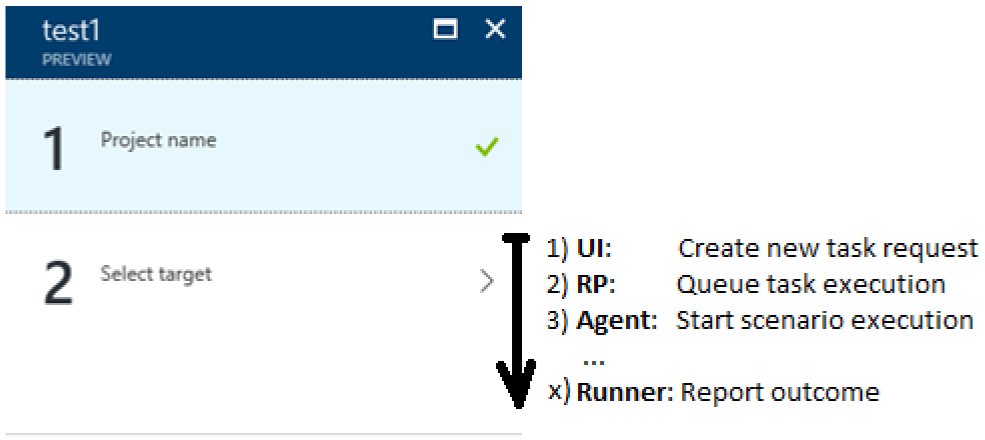
Scenario Loop:
S1) Dequeue scenario/task execution and deserialize
S2) Start scenario/task execution
S3) Request to download project (SSMA/DMA)
S4) Report scenario/task execution outcome

Operation Loop:
O1) Dequeue LRO execution
O2) Start LRO execution

Options

We can only use one field to correlate events for E2E tracing. The field is called **ActivityID** and we have the following options:

1. **ClientRequestID**


test1
PREVIEW
1 Project name
2 Select target
1) UI: Create new task request
2) RP: Queue task execution
3) Agent: Start scenario execution
...
x) Runner: Report outcome

To correlate all the events for a user request (GUID).

Troubleshooting:

1. Get **ClientRequestId** from the user to identify specific request
   i.e. <GUID> representing "Select Target" task

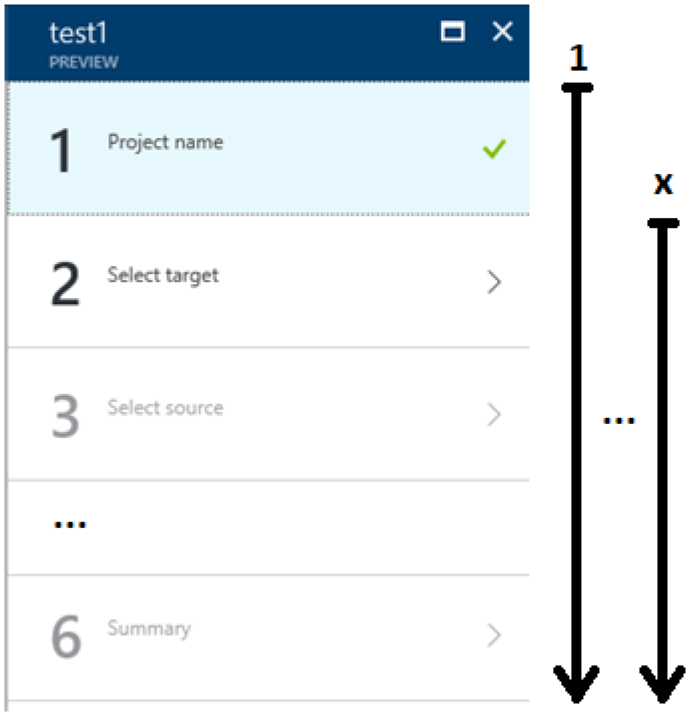
2. See all the related events for the given request from all the loops (RP Loop, Scenario Loop, ...)
3. If more context is needed, use **ProjectId** to find other requests

Concerns:

Auto-correlating events across async loops (i.e. RP Loop vs. Scenario Loop) won't help much with troubleshooting in many cases. Imagine user initiating RunCLMigration scenario (=> ClientRequestId 1) and then clicking Refresh button 3 times (=> ClientRequestId 2,3,4). A failure will appear on the UI for the last Refresh click hence user provides ClientRequestId4 to troubleshoot. But that won't help us – we need ClientRequestId1 instead to be able to figure out what went wrong.

In this scenario, we have 4 ClientRequestIds generated on the RP Loop and only execution (~ one TaskId) on the Scenario Loop. We cannot include all 4 requests for correlation purposes and picking only one is questionable.

2. ProjectId



To correlate all the events for a project (**SubId/ResourceGroup/ServiceName/Project**)

Troubleshooting:

1. Get **ProjectID** and **timeframe** from the user to identify subset of relevant events for the given migration project
2. See all the related events for the given project (could be several months worth of data)
3. Zoom in to focus on the given timeframe

Loop-specific ID

To correlate all the events generated on the given loop only.

The idea is similar to option1 – we start with ClientRequestId on the RP Loop but we won't propagate this value to the other loops. We only establish mapping between ClientRequestId (from RP Loop) to another ID (on the other loops) as follows:

1. **ClientRequestId to TaskId to map between RP Loop and Scenario Loop**
2. **ClientRequestId to OperationId to map between RP Loop and Operation Loop**
3. **ClientRequestId to another ClientRequestId to map between RP Loop and Project Loop**

This means E2E Tracing would only allow you to correlate events within one loop.

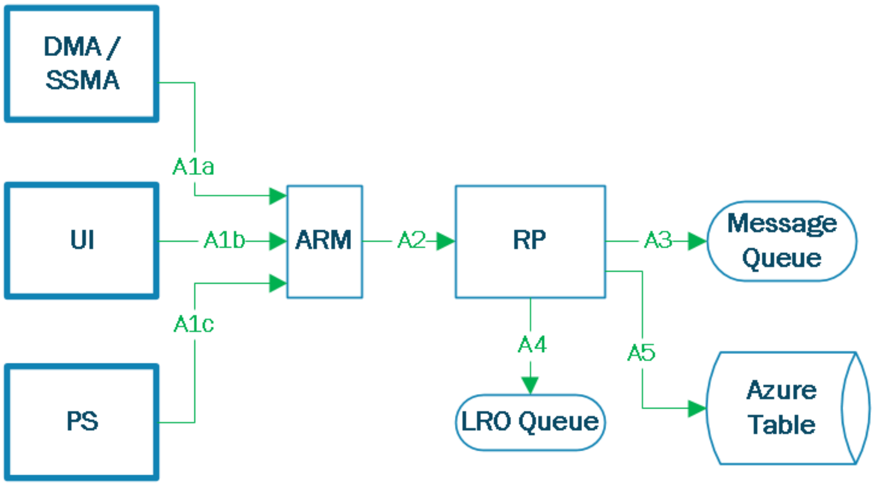
And the correlation ID would be different for each loop:

1. **ClientRequestId for RP Loop and Project Loop**
2. **TaskId for Scenario Loop**
3. **OperationId for Operation Loop**

Troubleshooting:

1. Get **ClientRequestId** from the user to identify specific request
   i.e. <GUID> representing "Select Target" task

2. See all the related events for the given request on the RP Loop
3. Map **ClientRequestId** from RP Loop to another loop ID as needed (**TaskId, OperationId, ClientRequestId**)
4. See all the related events for the other loop based on the other loop ID (**TaskId, OperationId, ClientRequestId**)
5. If more context is needed, use **ProjectId** to find other requests

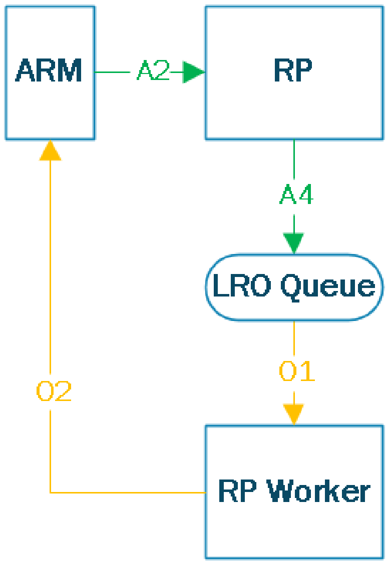


RP Loop (ClientRequestID)

- All the clients (UI, PS, SSMA, DMA) would generate unique ID (**ClientRequestID**) for each request and include it in http header
- **ClientRequestId** is already supported by ARM as one of the three correlation IDs so that we can propagate it to the AgentService naturally (without updating scenario's input or other hacks)
- Use the **ClientRequestId** to correlate events generated on the RP Loop

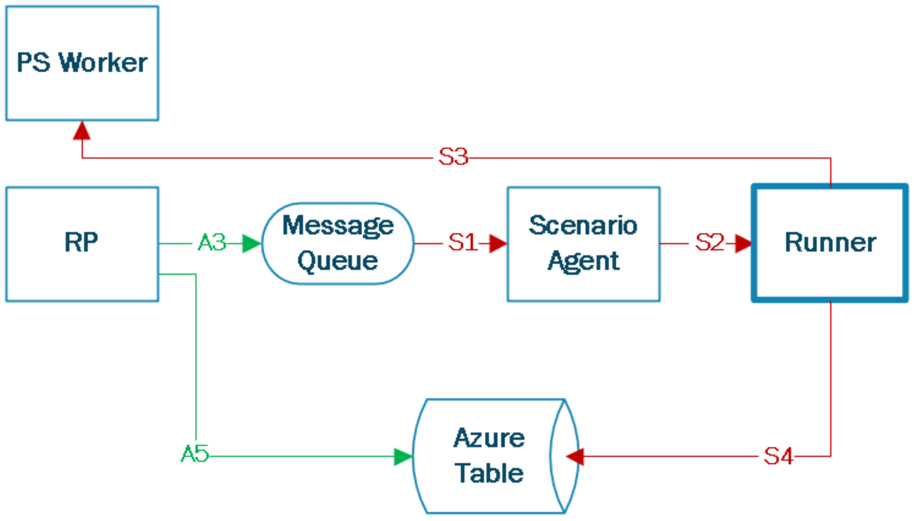
(and to map to other loop IDs)

- RP generates **OperationId** for **Operation Loop** or **TaskId** for **Scenario Loop**



LRO Loop (OperationID)

- Map **ClientRequestID** generated in the **RP Loop** to **OperationID**
- Add **OperationID** generated by RP to identify long running operation



Scenario Loop (TaskID)

- Map **ClientRequestID** generated in the **RP Loop** to **TaskID**
- Add **TaskId** generated by RP to identify task (SelectSource, RunMigration, ...)
- Update Agent Service protocol:

AgentStartScenarioInput

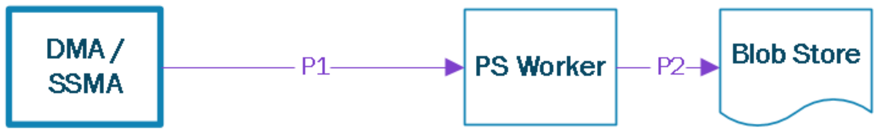
(ScenarioId, **TaskId**, Input, **ClientRequestID**)

TBD: Update RP to pass it to AgentService

ScenarioExecutionContext

(ScenarioId, **TaskId**, ScenarioTask, **ClientRequestID**)

TBD: Update AgentService to pass it to ScenarioRunner



Project Loop (ClientRequestID)

- DMA/SSMA will generate new **ClientRequestID**

How good have you found this content?