Recovery Guru Meeting Notes

August 11, 2010

# Resolved Issues

What is the functional model – what are the different functions required for RG and other failure state-based features, and how to they map to subsystems?

There are three major components to the RG functional model for OSA:

* Managed subsystems, such as DPL, BVL, NAS, and dom0. Managed subsystems are responsible for:
  + Updating a collection of failed subsystem components and objects, and presenting a client interface for retrieving the collection contents.
  + Sending a system event, with the needs-attention flag set, each time a failure is added or removed from the collection.
* A failure state aggregator that is presents a failure retrieval interface for to management clients. The aggregator passes management client requests to the appropriate subsystems, combining results as necessary. A plug-in architecture allows the aggregator to adapt to the native interfaces; each plug-in interacts with a specific subsystem interface, and is responsible for converting the failure entries to normal form.
* An event router that send alerts to management requirements in response to system events for failure state transitions.
* Management clients that receive alerts indicating failure state transitions, and respond by retrieving failure results from the aggregator.

Currently there is loose coupling between a failure state transition and a system event that announces it, and it is problematic when one is generated without the other. What can we do to ensure that we always get both?

Event sources are responsible for implementing mechanisms for maintaining this invariant. Neither the event aggregator nor the management clients can enforce this rule.

A possible solution is to convert to using a convenience method for signaling a failure state transition, where a system event is a required parameter of the method.

How are object references represented at different components in the functional model?

The managed subsystem provides native object references to the failure state aggregator, which provides these references to the management clients. Management clients are responsible for converting these references to normal (CIM reference) form.

# Unresolved Issues

What is the information model?

* What CIM model does Amelia want for RG and other failure state-based features?
* How should the component that aggregates fault state information from managed subsystems represent that information to the provider?

What component owns the failure type – recovery action map?

I suggest that the management client owns and defines this.

What is the normalized data representation for a failure entry? Do we need a common definition file for failure information, as we have for event types?

One advantage of this approach is that the failure entry could be associated with an event type ID, so that the code to add a failure entry could automatically select the appropriate event and send it.

# Meeting Comments

Scott: How this should be:

Sent out Monitoring ER what has additions to RG. Upon further analysis, there are some features that would be lower priority. It would be good to get to a common understanding on what are the required features.

In terms of flow: Today, we have very loose coupling between event logging, needs-attention indication, and failure list insert/remove. Desired model: subsystem event (with severity), also contains field stating that “action is required”. Trigger point for management client is event. After user takes care of the problem, there is a “recheck” (just retrieve failure list again), and failure entry drops out of list, and the result is a “resolution event”.

Eric:

Does this move us away from a subsystem-managed recovery failure list?

Scott:

There would still be a failure list. It would be populated via the event.

Ray:

Good concept to have the first approximation of the solution is to have serviceability components adapt to the managed subsystems.

Scott:

We just need to make sure that we achieve the desired model, and address the existing customer concerns.

* Loose coupling between events and failure state transitions. CRITICAL implied user action required. Failure type insertion was completely separate, so there were cases where we didn’t have 1:1 match between these two things.
* Lifecycle of problems. We log when the problem occurs, but not when the problem is resolved.
* We need to support more severities than two, and decouple “user action required” from a specific severity (the OEMs may change severity for an event).

Greg:

* Separate needs-attention and alert flags.
* For DPL at least, failure entry types aren’t persisted.

Eric:

What about NAS?

Chris:

NAS generates event when volume goes offline. Event contains reason why subsystem went offline. There is no failure state that says that the event is offline due to a failure (as opposed to an administrative offline).

Ray:

Add OSA application, we write an adapter to what the managed subsystem provides. Adapter is responsible for generating appropriate fault list entry.

Scott:

Today we have a controller offline failure type, could happen for a number of different reasons. We can’t distinguish why. Could we just say that a NAS volume admin offline is a failure that requires user attention?

One thing we want to do is we want to more accurately specify recovery actions, this requires more detailed failure types (more accurate isolation of the failure reason).

Chris:

If there is a component that needs attention, what are the side effects? What operations might I not be able to do? (upgrade, create new objects, delete old objects, etc.) Reason is that customer needs to be able to install a patch even if a needs attention condition exists.

Scott:

These cases do exist (such as PF out-of-compliance conditions).

Question back to Eric, Ray, Gili: How did this work with BVL?

Eric:

BVL implemented a failure interface similar to DPL. QoS Monitor had to handle some failure types that spanned BVL/DPL.

Scott:

We can perform optimizations to get additional failure information for our components, but what about 3rd parties?

Gili:

3rd parties need to accommodate us to the point that we can write an adapter. We should have a set of minimum requirements that a managed subsystem should satisfy for it to be a good “OSA citizen”. It’s better that the third party owns the adapter (this implies that the adapter API be an external feature).

Liz:

If a subsystem performs a function like SNMP alerts, do we require that the subsystem disable this so that only the serviceability component does this?

Ray:

A black box app may have the ability to generate SNMP alerts, but this capability could be made latent instead of having to perform surgery. We could use the SNMP events as an event source, too.

Gili:

Controller states are among the hardest integration problems. Was one of the most difficult issues with BVL.

Scott:

Someone has to determine whether event requires action. One developer may be working on event, another working on failure list?

Greg:

We always generate an event, and only insert a failure if needs\_attention is set for the event type. Mapping can be multiple event types to a particular recovery action. We’re not at the point yet where we have one call that does everything that’s necessary.

Scott:

OEMs want to say whether action is required or not? Should they be able to override this?

Greg:

They want to hide the “lesser” of multiple recovery actions for the same cause, so there’s only one recovery action. This isn’t necessarily the best plan, might be rarely used or will have problems.

Scott:

Agree we want to show only root cause recovery actions. Seems best to be able to vary severity but not needs-attention. Higher level logic should determine whether needs-attention needs to be suppressed due to root-cause filtering.

Eric:

We choose in the subsystem whether to send events/failures based

Scott:

With vdrive failure, we could get needs-attention at vdrive, block vol, and NAS vol. We can only address the vdrive.

Ray:

If a CIM component knows that a failure has dependent objects, it might be good for the CIM-level logic to show only the failure for the lowest level component.

Eric:

Sorted all hw failures to top of RG list, and prioritized these fixes.

Another way would be to fully cook the recovery list provided to CIM. Problem is correlating between subsystems. However, CIM model may be incomplete in this area, also.

Sridhar:

CIM should cover association between BVL and DPL objects, but it’s somewhat roundabout.

Ray:

Joe has said that he’ll put whatever the client needs into the CIM model ☺