Plugin for Unmanaged Racks

Overview

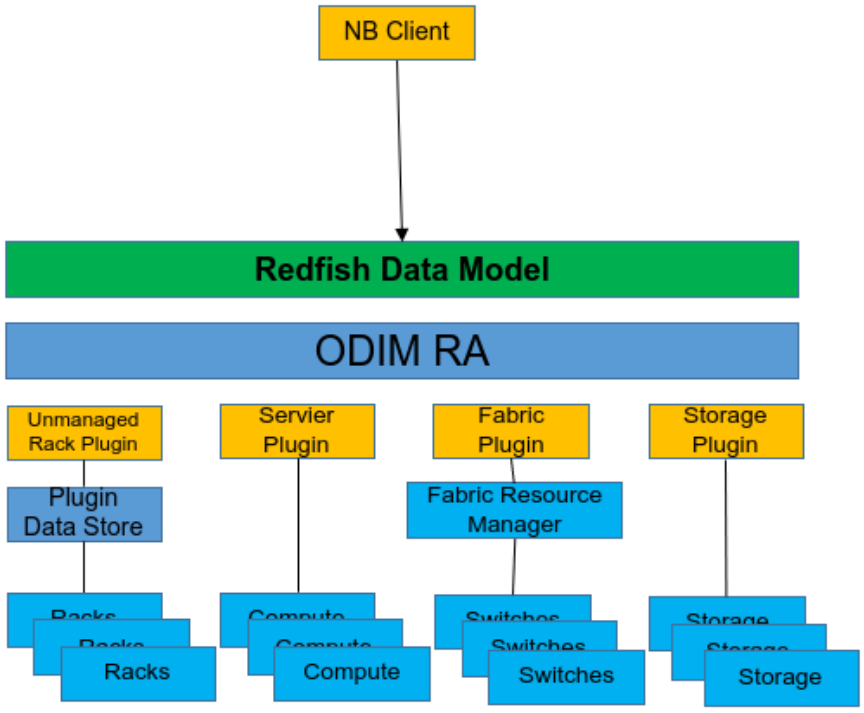
The Open Distributed Infrastructure Management (ODIM) framework provides a way to manage heterogeneous infrastructure resources using a single, standardized interface and data model. Infrastructure resources typically include stand-alone resources such as servers, storage and Ethernet networking equipment and composite resources such as racks, which in turn contain stand-alone resources. The relationships between those resources needs to be modeled in Redfish at a Rack/Row level and as such. We are proposing an 'Unmanaged Rack' plugin to fulfil this function. The need for a special plugin for unmanaged racks is due to the requirement to host resources under these racks. This plugin will emulate a resource manager for these racks.

The unmanaged Rack plugin that will allow management applications to set and get resources contained within a rack as well as dynamically sync lifecycle events such as adding and removing resources. It will not be extended to support racks with dedicated resource managers as they usually have this functionality themselves. It is expected that additional 'Managed' rack plugins will be contributed by each vendor as part of the open source community and as such are out of scope for this development effort.

Solution

The Unmanaged Rack plugin will have two components – the API server that conforms to the Plugin Developer's Guide (PDG) and a Redfish backend 'plug in data store' that simulates a resource manager. The API server is a stateless entity while the Redfish backend would be expected to maintain state of the groups of Racks and associated resources they contain. The Rack Manager plugin's API server will conform to the mandatory functions set in the PDG. It will also need to have a management interface with username and credentials so that the interface can be added using the Aggregation Service.

Figure 1



Use Cases

Adding Resource Manager

The following sequence describes the system lifecycle of the Unmanaged Rack plugin and data store in relation to the rest of the resource specific plugins and their associated resources for which they provide management communications.

1. A Northbound (NB) Redfish client uses a POST REST call to add an AggregationSource to the AggregationService collection of AggregationSources. This will add a new resource manager for Racks and RackGroups. This resource manager will not have any resources under it though.
2. The Unmanaged Rack plugin subscribes to Chassis collection resource change events. It will maintain this subscription throughout its life. This subscription is needed in order to clean up links under its resources when resources like servers and switches are removed (in which case a ResourceRemoved event will be emitted).

Adding a Rackgroup

A RackGroup (a Chassis with ChassisType=RackGroup) is needed to be able to add a Rack to the resource collection. Each Rack (ChassisType=”Rack”) must have a “ContainedBy” link pointing to the containing RackGroup. With this in mind the first thing that needs to be added is a RackGroup

1. A NB Redfish client POSTS a new Chassis with ChassisType=RackGroup to the /redfish/v1/Chassis collection. The Actual Chassis object that is created needs to follow the profile ODIMServerHardwareManagement.v1_0_0. It is to be noted that some of these properties are ReadOnly and therefor need to be part of the POSTS (OnCreate). What those properties are will differ for different ChassisType and the engineering team should therefor use the Profiles for guidance
2. The plugin will store the new RackGroup in its persistent storage.

Adding a Rack

1. A NB Redfish client POSTS a new Chassis with ChassisType=Rack to the /redfish/v1/Chassis collection. The POSTS needs to follow the profile ODIMServerHardwareManagement.v1_0_0 and therefore have things like Manufacturer, Model, ContainedBy and such populated. Any failure to comply with this profile should be rejected. The ContainedBy link needs to be verified to make sure it is pointing to an existing RackGroup. If it does not the operation should be rejected.
2. The plugin will store the new RackGroup in its persistent storage.

Patching a Rack with a New Server, Switch or Storage Device

When a new Chassis resource with ChassisType=Rackmount is introduced in the resource model, independently of how it got there, one should add it to a Rack. This is done by PATCH:ing the Rack with a new Contains element pointing to the new Chassis.

1. A NB Redfish client uses a PATCH operation on the Rack object including a new Contains element that points to a new Chassis resource.
2. The plugin will verify that the new link points to an existing Chassis. If the Chassis does not exist it should return an error and emit a ResourceNotFound event. It will also verify that this Chassis is not already linked from a different or the same Rack. In such case it should return an error and issue a ResourceAlreadyExists event.
3. The plugin will store the links in its persistent storage.
4. The plugin will test to see if this new Chassis supports ContainedBy. If so the plugin will create a link on that Chassis ContainedBy property that points to the Rack.
5. If supported, The Northbound (NB) Redfish client PATCHes location (including Placement) information for all resources. For switch ports, the NB client can optionally PATCH each Switch Port Schema “ConnectedPort” with the “EthernetInterface” or “NetworkPort” on the server or storage node NIC it is cabled to. This step will be implementation specific.

Removing a Server, Switch or Storage Device

If a resource is removed, the Unmanaged Rack plugin detects this by subscribing to ResourceRemoved Events. The plugin will then need to remove the as sociated ‘Contains’ link for ChassisType=”Rack”.

1. The plugin detects a ResourceRemoved event for one of the Chassis
2. It will look at the referred to resource and find out if it is a resource that has the ChassisType=Chassis. If so it will look through all Rack objects to see if it has a Contains link pointing to this Chassis
3. If it does it will remove that link. If it does not it will issue a ResourceNotFound Event pointing to the Resource

Removing a Rack or Rackgroup

When removing a Rack or a Rackgroup the principle is that the operation will only work if it is empty, which in other words means that the Contains list is empty.

1. The NB Redfish client issues a DELETE operation on /redfish/v1/Chassis where ChassisType is set to either RackGroup or Rack.
2. The plugin verifies that the Contains list is empty. If it is it will remove the resource and return success and sends out a ResourceRemoved Event. If the Contains list is not empty the plugin will return an error and sent out a ResourceCannotBeDeleted Event.

Mockups for different Servers

DL385 HPE Gen 10 - 1024022.zip