Generic Action Controller (GAC)

The EMCO orchestrator supports placement and action controllers to control the deployment of applications. Placement controllers allow the orchestrator to choose the exact locations to place the application in the composite application. Action controllers can modify the state of a resource (create additional resources to be deployed, modify, or delete the existing resources).

GAC is an action controller registered with the central orchestrator. GAC allows you to create a new Kubernetes object and deploy it with a specific application. The app should be part of the composite application. GAC allows you to modify an existing Kubernetes object which may have deployed using the helm chart for an app or GAC itself. Modification may correspond to specific fields in the YAML definition of the Kubernetes object.

Components of GAC

- **GenericK8sIntent** - Specifies the parent intent supports the resource and their customizations.
- **Resource** - Specifies the Kubernetes object that needs to be created. You can create a resource template file with all the required configurations for the resource and upload it. GAC reads the configurations and applies them to the resource. A resource template file is mandatory except for ConfigMap /Secret.
- **Customizations** - Specifies the configurations for the Kubernetes object. Customization allows you to customize an existing Kubernetes object or a newly created one. Customization supports uploading multiple data files with configuration data for a ConfigMap/Secret. Customization also helps modify existing Kubernetes objects using a JSON patch or Kubernetes strategic merge patch.
    - **JSON patch** - GAC supports the standard JSON patch. You can modify the specific fields in the Kubernetes resource definition using the JSON patch. You need to specify the path, operation, and value in the GAC customization. You can also create a ConfigMap/Secret using the JSON patch. You can pass the JSON patch data in the customization. GAC will create the ConfigMap/Secret using the values provided in the resourceGVK (APIVersion, Kind, Name) and apply the configuration data provided in the JSON patch.
    - **Strategic Merge patch** – GAC is integrated with the Kubernetes Strategic Merge patch. This is specific to Kubernetes and is an extension of the standard JSON merge patch. The main advantage of using a Strategic Merge patch over a JSON patch or JSON Merge patch is to merge lists strategically. You can only modify specific fields using a standard JSON patch. A simple JSON Merge patch will always replace the list. For example, the container in a Deployment is a list. A simple JSON Merge patch will replace the original document's containers with the patch's list. Strategic Merge internally uses the go struct tag to determine how the patch should be applied (merge/delete /replace). Since the containers field in the go struct has a declarative tag *patchStrategy=merge, and patchMergeKey=name* when we apply the Strategic Merge patch, the list will be merged, not replaced. This is feasible only for specific fields. Please see notes on the strategic merge patch for more details.

> *Please refer to GAC design document for more details*

Please see the `Generic Controller Intent` section in the OpenAPI for the detailed definition of APIs exposed by GAC.

Please refer to Creating Kubernetes Objects Using GAC for the step by step guide to create a Kubernetes resource using GAC

Please refer to EMCO Resource Lifecycle and Status for more details on the GAC resources lifecycle.

Please see the examples in test-gac for a complete walkthrough of GAC customizations

Please refer to the Kubernetes official Strategic Merge Patch documentation for more details about Strategic Merge Patch.