EMCO Operations

Table of Contents

After EMCO has been installed in a central cluster and some 'edge' clusters have been prepared, this section describes the basic operational sequences that are used to onboard clusters and create and deploy composite applications.

EMCO API

Interaction with the EMCO REST API is primary interface to EMCO.

View the EMCO API documentation with the swagger editor: https://editor.swagger.io/?url=https://raw.githubusercontent.com/onap/multicloud-k8s/master/docs/emco_apis.yaml

A Postman collection can be found here: https://github.com/onap/multicloud-k8s/blob/master/docs/EMCO.postman_collection.json

The EMCO REST API is the foundation for the other interaction facilities like the EMCO CLI and EMCO GUI.

EMCO CLI

EMCO has a CLI tool called *emcoctl*. More information can be found here: https://github.com/onap/multicloud-k8s/tree/master/src/tools/emcoctl

EMCO GUI

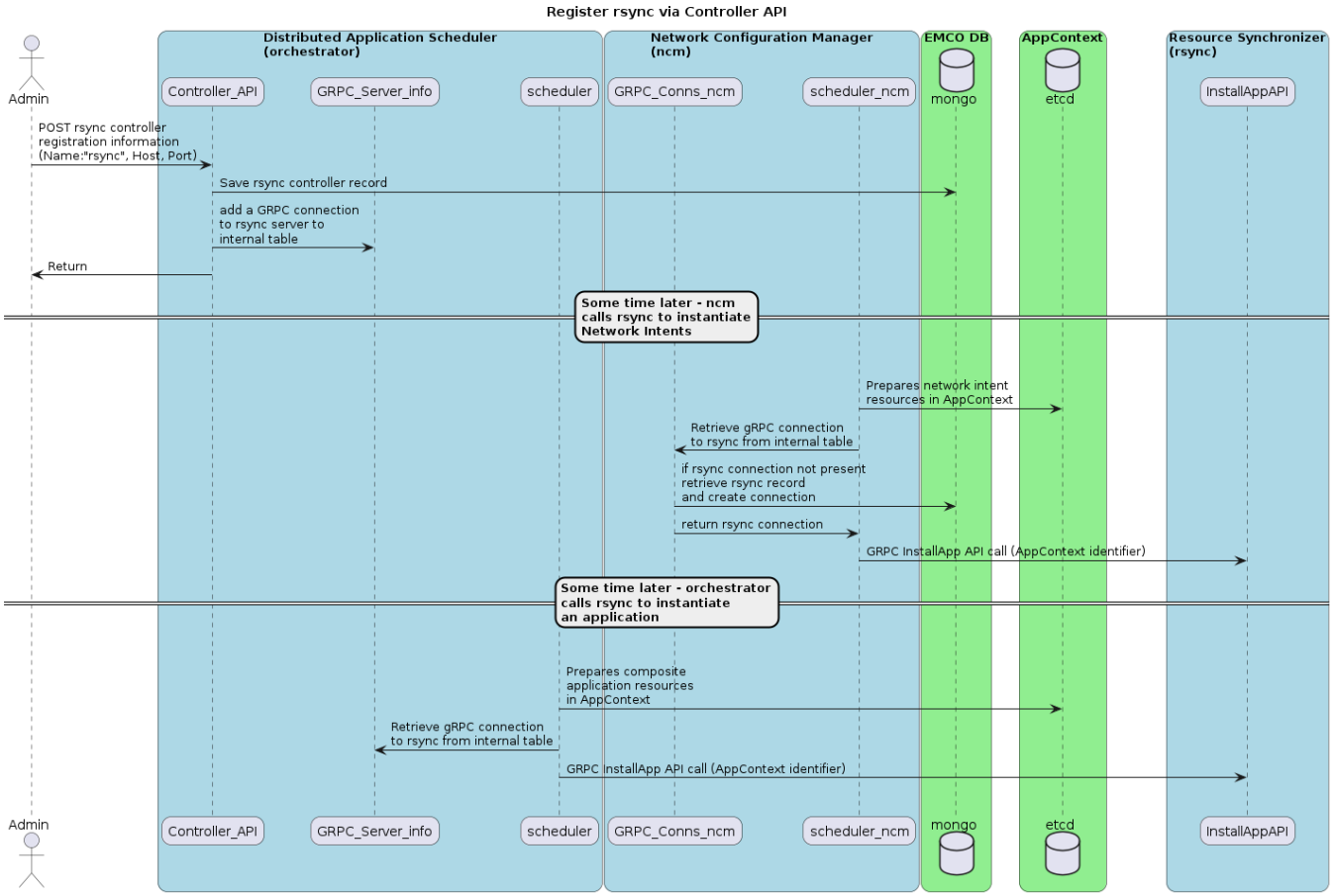
EMCO has a GUI - details: TBD

EMCO Setup

The EMCO architecture is extensible via the use of controllers which can be used to handle specific placement and configuration (i.e. actions) operations. The EMCO orchestrator communicates with these controllers via a gRPC interface. EMCO supports a controller API to allow the administrator to register these controllers with EMCO to provide the necessary connection information (name, port) as well as controller type and relative priority.

The EMCO *rsync* microservice also exposes it's API via gRPC to the EMCO microservices. So, while *rsync* is not a placement or action controller, it is also registered with the controller API so that EMCO microservices that interact with *rsync* as a gRPC client can obtain the gRPC connections details in the same manner as with other controllers.

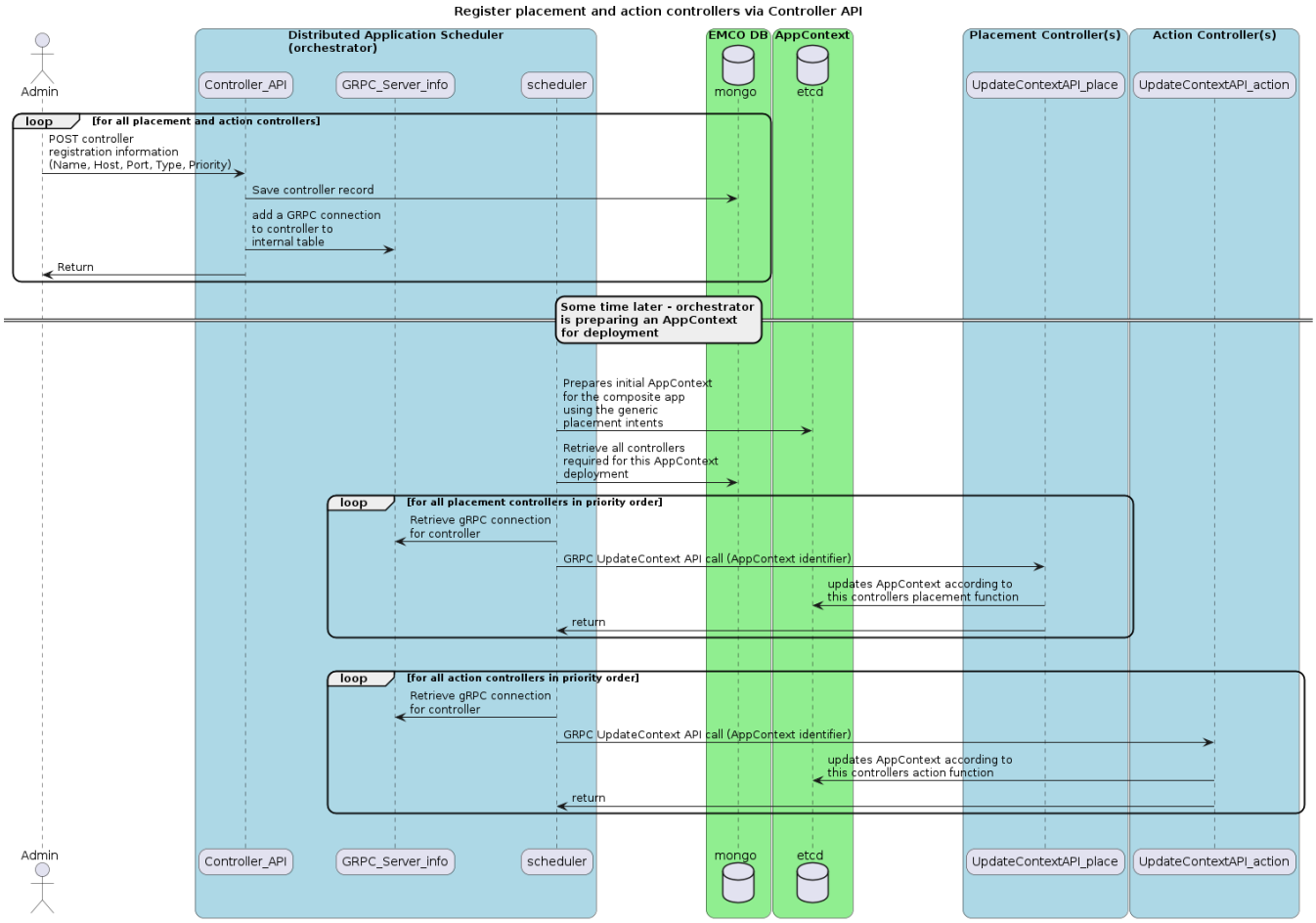
The sequence diagram illustrates the process of registering *rsync* with the *orchestrator* via the Controller API. The diagram also shows two scenarios of how the rsync registration information is used. In the first case, the *ncm* component will obtain the *rsync* controller record to set up its own GRPC connection table when it needs to communicate with *rsync* to deploy network intents. In the second case, *orchestrator* obtains the *rsync* client connection - which will already be in its internal client table - during the sequence of installing a composite application.



Controller Registration

As mentioned above, EMCO provides for the addition of controllers to perform specific placement and action operations on a Composite Application prior to it being deployed to a set of clusters.

This sequence diagram illustrates how action and placment controllers are registered with *orchestrator*. Also shown is the part of a sequence where the *orc hestrator* is preparing the AppContext for a composite application and the controllers are invoked in priority order to update the AppContext per their specific function.



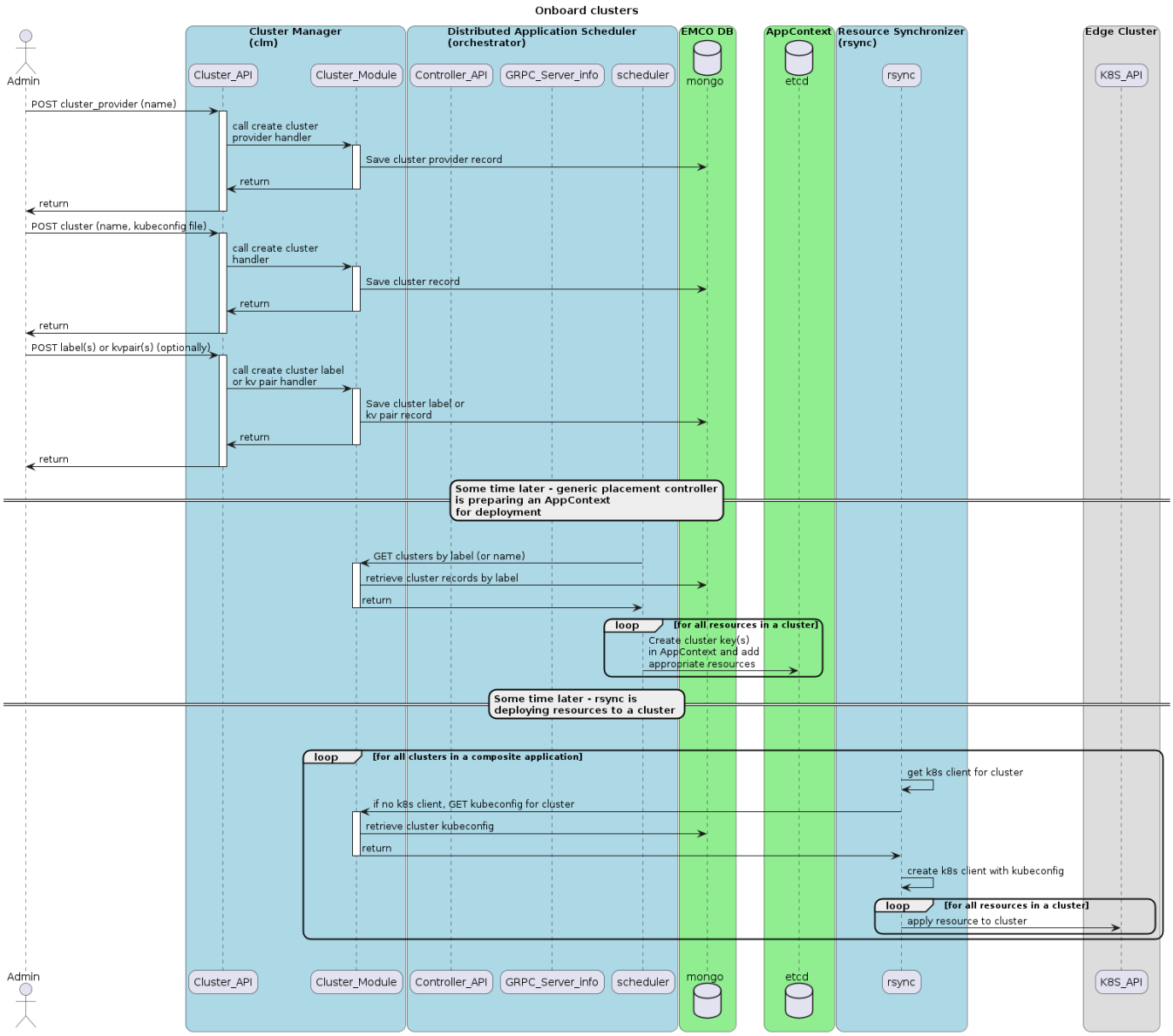
Onboard Clusters

Clusters are onboarded to EMCO by first creating a Cluster Provider and then adding Clusters to the Cluster Provider.

When a cluster is created, the KubeConfig file for that cluster is provided as part of the multi-part POST call to the Cluster API.

Additionally, once a Cluster is created, labels and key value pairs may be added to the Cluster via the API. Clusters can be specified by label when preparing placement intents.

The sequence diagram illustrates the process onboarding the cluster and a couple examples of how other sequences in the EMCO system obtain cluster information during operation.



Create Network Intents

(show how network intents are defined and deployed to clusters)

Create Composite Application

(show how composite applications are prepared along with the intents required to deploy them)

Basic Composite Application Lifecycle

(show instantiate, terminate, status operations of a composite application)