

Homing and Allocation Service (HAS)

- [Introduction](#)
- [HAS in Service Instantiation workflows](#)
- [HAS Architecture \(R2\)](#)
- [HAS - Active-Active High Availability Architecture](#)
- [Lifecycle of a Homing request in HAS](#)
- [Use cases](#)
 - [Residential vCPE: vCPE Homing Use Case](#)
 - [5G RAN: Homing 5G RAN VNFs](#)
- [A sample heuristic greedy algorithm of HAS](#)
- [Guide to writing Homing Specifications](#)
- [Triaging Homing decisions](#)
- [HAS Developer guide](#)
- [HAS Code information](#)
 - [Gerrit: https://gerrit.onap.org/r/#/admin/projects/optf/has](https://gerrit.onap.org/r/#/admin/projects/optf/has)
 - [Master Branch : https://git.onap.org/optf/has/](https://git.onap.org/optf/has/)
 - [Beijing Branch: https://github.com/onap/optf-has/tree/beijing](https://github.com/onap/optf-has/tree/beijing)
 - [Beijing Release Artifacts:](#)

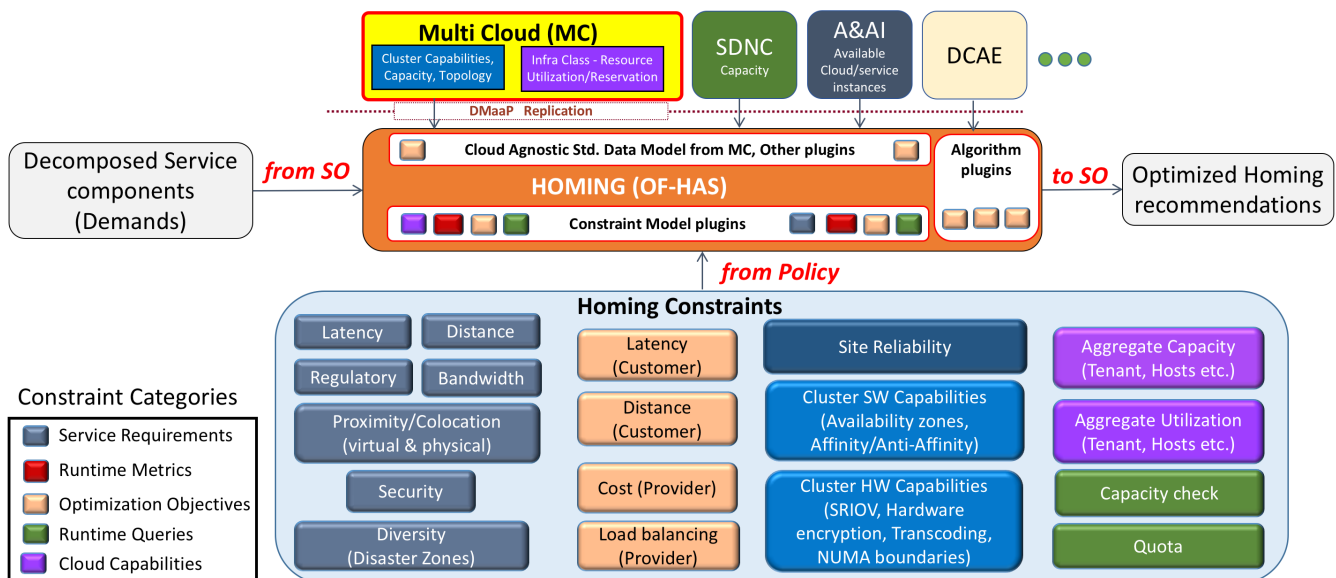
Introduction

OOF-HAS is an policy-driven placement optimizing service (or homing service) that allows ONAP to deploy services automatically across multiple sites and multiple clouds. It enables placement based on a wide variety of policy constraints including capacity, location, platform capabilities, and other service specific constraints.

HAS is a distributed resource broker that enables automated policy-driven optimized placement of services on a global heterogeneous platform using ONAP. Given a set of service components (based on SO decomposition flows) and requirements for placing these components (driven by policies), HAS finds optimal resources (cloud regions or existing service instances) to **home** these service components such that it meets all the service requirements. HAS is architected as an extensible homing service that can accommodate a growing set of homing objectives, policy constraints, data sources and placement algorithms. It is also service-agnostic by design and can easily onboard new services with minimal effort. Therefore, HAS naturally extends to a general policy-driven optimizing placement platform for wider range of services, e.g., DCAE micro-services, ECOMP control loops, server capacity, etc. Finally, HAS provides an traceable mechanism for what-if analysis which is critical for ease of understanding a homing recommendation and resolving infeasibility scenarios.

HAS in Service Instantiation workflows

Below is an illustration of HAS interactions with other ONAP components to enable Policy driven homing. The homing policy constraints have been expanded (and categorized) to highlight the range of constraints that could be provided to HAS for determining the homing solution. The figure also shows how HAS uses a plugin-based approach to allow an extensible set of constraints and data models.

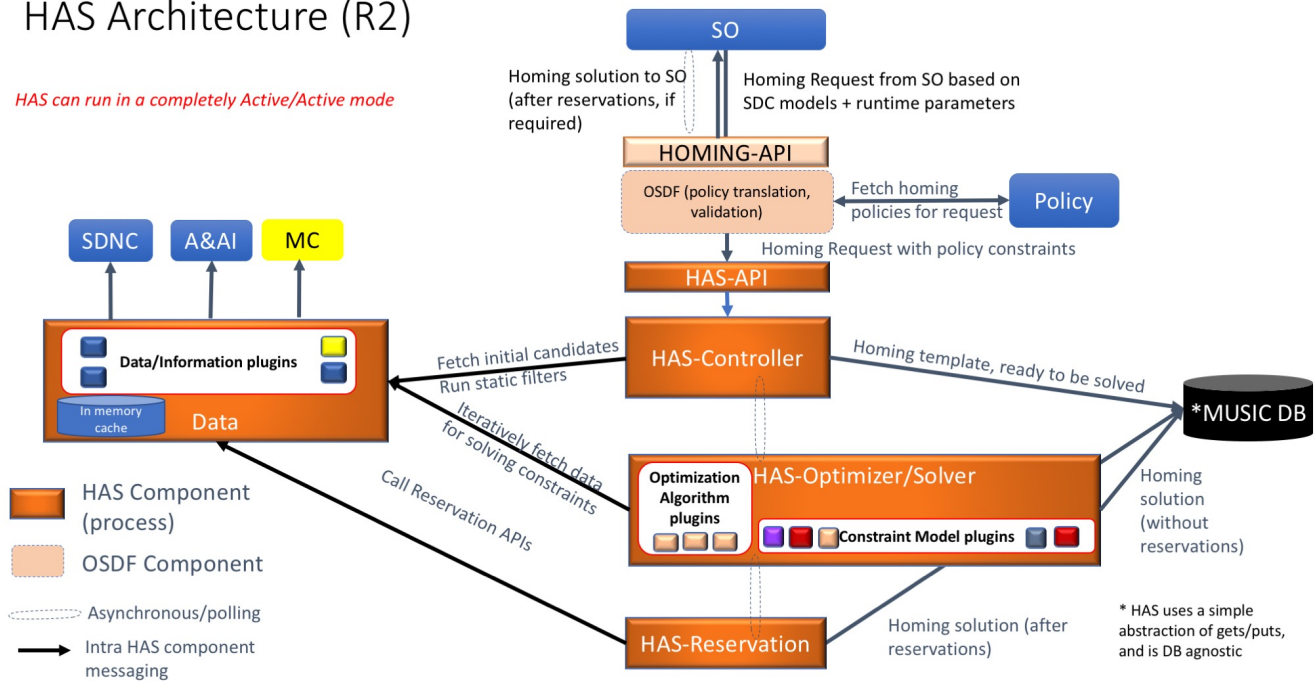


More information on how homing constraints are specified can be found at [OOF-HAS Homing Specification Guide](#), and a sample homing template has been drawn up for residential [vCPE Homing Use Case](#).

HAS Architecture (R2)

HAS Architecture (R2)

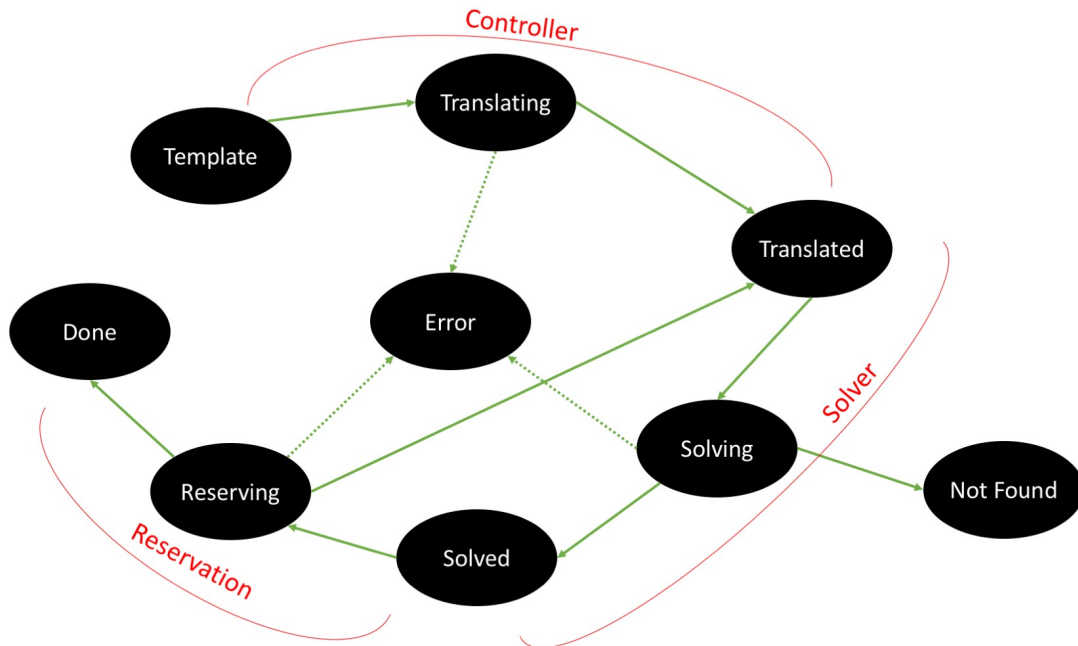
HAS can run in a completely Active/Active mode



HAS - Active-Active High Availability Architecture

HAS is implemented based on an active/active high availability architecture ([HAS - Active-Active High Availability Architecture](#)), where each component of HAS can scale horizontally independent of the others. This also allows HAS to have highly scalable hybrid deployment architectures ([OOF Beijing R2 OOM Deployment Planning](#)).

Lifecycle of a Homing request in HAS

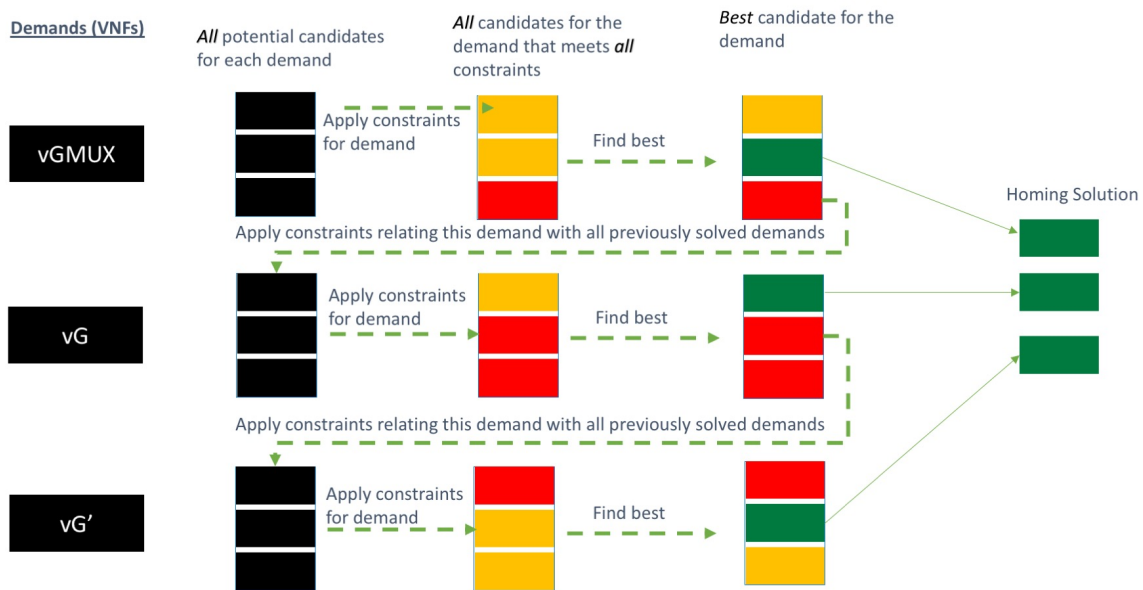


Use cases

Residential vCPE: **vCPE Homing Use Case**

5G RAN: **Homing 5G RAN VNFs**

A sample heuristic greedy algorithm of HAS



Guide to writing Homing Specifications

HAS is service agnostic by design and the [OOF-HAS Homing Specification Guide](#) can help create homing specifications for new services.

Triaging Homing decisions

[Triaging HAS homing decisions](#)

HAS Developer guide

The [HAS Developer Guide](#) will give more information on how to deploy HAS, and contributing to the HAS code.

HAS Code information

Gerrit: <https://gerrit.onap.org/r/#/admin/projects/optf/has>

Master Branch : <https://git.onap.org/optf/has/>

Beijing Branch: <https://github.com/onap/optf-has/tree/beijing>

Beijing Release Artifacts:

<https://nexus.onap.org/content/repositories/releases/org/onap/optf/has/optf-has-conductor/1.1.1/> (binaries)

<https://nexus.onap.org/content/repositories/releases/org/onap/optf/osdf/optf-osdf/1.1.1/> (binaries)

nexus3.onap.org:10001/onap/optf-has:1.1.1 (docker image)

nexus3.onap.org:10001/onap/optf-osdf:1.1.1 (docker image)