serviceOrchestrationPushManagement

Service Description

Abstract
This document provides service description for the **serviceOrchestrationPushManagement** service.

Contents

1 Overview

This document describes the **serviceOrchestrationPushManagement** service, which enables systems (with operator role or proper permissions) to manage data and activities related to the push type of service orchestration process in bulk. An example of this interaction is that a higher entity (a dedicated system directly or a human operator indirectly via some tool) consumes this service to subscribe other consumer systems to receive push orchestration results and also to trigger the push orchestration process when it is necessary.

The orchestration process can be implemented by using different strategies:

- **simple-store:** when the matching service instances are stored in a database as peer-to-peer rules that are maintained by higher entities and no actual service details are obtained from the Local Cloud;
- **flexible-store:** when the service requirements are stored in a database as service and provider attribute rules that are maintained by higher entities and actual service details are obtained from the Local Cloud;
- **dynamic:** when the matching service instances and their actual service details are discovered on the fly from the Local Cloud based on the service requirements specified by the initiator party (consumer system or higher entity).

The **serviceOrchestrationPushManagement** service contains the following operations:

- *subscribe* creates subscriptions in bulk for other consumer systems that can be triggered anytime to perform the orchestration process and push newly orchestrated matching service instances to the subscribed consumers;
- *unsubscribe* removes subscriptions in bulk;
- *trigger* initiates service orchestration processes based on consumer system names or subscription identifiers or creator system names;
- *query* lists the existing subscriptions that match to the filtering requirements

The rest of this document is organized as follows. In Section 2, we describe the abstract message operations provided by the service. In Section 3, we end the document by presenting the data types used by the mentioned operations.

1.1 How This Service Is Meant to Be Used

The purpose of the service is to handle the push orchestration related data and activities centrally and in bulk.

In certain use cases, the right of deciding when and how the service orchestration process should be executed is not belonged to the consumer system itself, but is assigned to a higher level entity (a dedicated system or a human operator). This higher entity invokes the *trigger* operation when it deems that a re-orchestration is necessary.

Due to certain device or software constraints, it is also possible that consumer systems are not capable to define their own service requirements (supported interface, security level, service metadata, etc...). In these situations, the task and responsibility of creating - and removing - proper subscriptions, could also belong to the higher entity. To manage these tasks, the *subscribe* and *unsubscribe* operations could be invoked.

For discovering the existing subscriptions and their details, the *query* operation can be used.

1.2 Important Delimitations

The requester has to identify itself to use any of the operations. Only those subscriptions can be revoked which were created by the requester.

1.3 Access policy

The service is only available for operators, dedicated Core/Support systems and those who have the proper authorization rights to consume it.

2 Service Operations

This section describes the abstract signatures of each operations of the service. The **serviceOrchestrationPushManagement** service is used to *subscribe*/*unsubscribe* consumer systems for push orchestrations, *trigger* the orchestration process and *query* for existing subscriptions. In particular, each subsection names an operation, an input type and one or two output types (unsuccessful operations can return different structure), in that order. The input type is named inside parentheses, while the output type is preceded by a colon. If the operation has two output types, they are separated by a slash. Input and output types are only denoted when accepted or returned, respectively, by the operation in question. All abstract data types named in this section are defined in Section 3.

2.1 operation subscribe (OrchestrationSubscriptionListRequest) : OrchestrationSubscriptionListResponse / ErrorResponse

Operation *subscribe* creates subscription records for the given consumer systems with the targeted services and the defined service requirements. It requires at least the requester's identity, the consumer systems' names, their targeted service definitions and their notify interfaces. The results of this operation are the created subscription records.

Note: As long as a subscription exists, the orchestration process to the given system and for its targeted service can be triggered anytime by via the *trigger* operation.

2.2 operation unsubscribe (OrchestrationUnsubscribeListRequest) : OperationStatus / ErrorResponse

Operation *unsubscribe* requires the requester's identity, a list of subscription identifier and removes the associated subscription records. Only those subscriptions can be revoked which were created by the requester.

2.3 operation trigger (OrchestrationPushTriggerRequest) : OrchestrationPushJobListResponse / ErrorResponse

Operation *trigger* creates orchestration job records that will be executed in a working queue. It requires the requester's identity and optionally either a subscription id list or a consumer system name list. If the subscription id list is provided, then the orchestration jobs will be created by the associated subscription records. If the consumer system name list is provided, then all the subscription record will be queried where the given consumer system name is the target system and the orchestration jobs will be created based on these subscription records. If none of the previous ones are provided, then all the subscription record will be queried where the requester system is the creator of the subscription record and the orchestration jobs will be created based on those.

The result of this operation is the created orchestration job records.

2.4 operation query (OrchestrationSubscriptionQueryRequest) : OrchestrationSubscriptionListResponse / ErrorResponse

Operation *query* lists the subscription records that match the filtering requirements. The query data must meet the following criteria:

- The operation returns results in pages. There are default page data settings, but the requester can provide a custom specification.
- If page number is specified, the page size must be specified as well and vice versa.
- In some Local Clouds there is a maximum page size.
- If a filter expects a list, there is an OR relation between the elements of the filter.
- There is an AND relation between different kind of filters.

3 Information Model

Here, all data objects that can be part of the **serviceOrchestration** service are listed and must be respected by the hosting System. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is used to denote a collection of named fields, each with its own data type. As a complement to the explicitly defined types in this section, there is also a list of implicit primitive types in Section 3.21, which are used to represent things like hashes and identifiers.

3.1 struct OrchestrationSubscriptionListRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
subscriptions	List<OrchestrationSubscriptionRequest>	yes	List of subscription request details.

3.2 struct Identity

An Object which describes the identity of a system. It also contains whether the identified system has higher level administrative rights.

3.3 struct OrchestrationSubscriptionRequest

Field	Type	Mandatory	Description
targetSystemName	SystemName	yes	The targeted consumer system.
orchestrationForm	OrchestrationForm	yes	Orchestration request details.
notifyInterface	NotifyInterface	yes	Interface details for sending push notifications.
duration	Number	no	The interval while the subscription is active.

3.4 struct OrchestrationForm

Field	Type	Mandatory	Description
serviceRequirement	ServiceRequirement	yes	Details of the targeted service.
orchestrationFlags	Set<OrchestrationFlag>	no	Set of orchestration flags.
qosRequirements	QoSRequirementMap	no	Quality of service requirements map.
exclusivityDuration	Number	no	The interval the service wanted to be exclusive.

Note: *simple-store* strategy supports only to provide the `serviceRequirement` and `orchestrationFlags` fields.

3.5 struct ServiceRequirement

Field	Type	Mandatory	Description
serviceDefinition	ServiceName	yes/no	The required service definition name. Mandatory in case of **dynamic** strategy.
operations	List<ServiceOperationName>	yes/no	The required service operation names. Exactly one operation must be defined, when the following orchestration flags are true: `ONLY_INTERCLOUD`, `ALLOW_INTERCLOUD`, `ALLOW_TRANSLATION`
versions	List<Version>	no	The required service versions.
alivesAt	DateTime	no	The orchestrated service must be alive by this time.
metadataRequirements	List<MetadataRequirements>	no	The orchestrated service must meet at least to one of the specified metadata requirement.
interfaceTemplateNames	List<InterfaceName>	no	The orchestrated service must offer at least one from the specified interface template names.
interfaceAddressTypes	List<AddressType>	no	The orchestrated service must offer at least one from the specified interface address types.
interfacePropertyRequirements	List<MetadataRequirements>	no	The orchestrated service must offer at least one interface that meets with one of the specified property requirements.
securityPolicies	List<SecurityPolicy>	no	The orchestrated service must meet with one of the specified security policies.
preferredProviders	List<SystemName>	no	Provider system names specified here have priority.

Note: *simple-store* strategy supports only to provide the `serviceDefinition` field.

3.6 struct MetadataRequirements

A special Object which maps String keys to Object, primitive or list values, where

- Keys can be paths (or multi-level keys) which access a specific value in a Metadata structure, where parts of the path are delimited with dot character (e.g. in case of ”key.subkey” path we are looking for the key named ”key” in the metadata, which is associated with an embedded object and in this object we are looking for the key named ”subkey”).

- Values are special Objects with two fields: an operation (e.g. less than) and an actual value (e.g. a number). A metadata is matching a requirement if the specified operation returns true using the metadata value referenced by a key path as first and the actual value as second operands.
- Alternatively, values can be ordinary primitives, lists or Objects. In this case the operation is equals by default.

3.7 struct OrchestrationFlag

Specific String:Boolean pair to control the orchestration process. Possible values:

- `MATCHMAKING`:

  If `true`, orchestration process includes a matchmaking process in order to return only one matching service instance if any.

  If `false`, orchestration process returns all the matching service instance if any.

  Supporting strategies: *simple-store*, *flexible-store*, *dynamic*

- `ONLY_PREFERRED`:

  If `true`, orchestration process considers only those matching service instances that are provided by a preferred provider if any.

  If `false`, but preferred providers are specified and have matching service instances, then orchestration process considers only those service instances that are provided by a preferred provider. Otherwise, non-preferred providers are considered.

  Supporting strategies: *flexible-store*, *dynamic*

- `ONLY_EXCLUSIVE`:

  If `true`, orchestration process considers only those matching service instances that are allows exclusivity. It automatically results `MATCHMAKING:true` as well.

  If `false`, but exclusivity duration is specified and there are matching services with exclusivity allowed, then orchestration process considers only those service instances that allows exclusivity. Otherwise, service instances without exclusivity are considered.

  Supporting strategies: *flexible-store*, *dynamic*

- `ALLOW_INTERCLOUD`:

  If `true`, orchestration process considers matching service instances from neighbor clouds when there are no local hits. Orchestrating from neighbor clouds automatically results `MATCHMAKING:true`

  If `false`, orchestration process doesn't consider matching service instances from neighbor clouds when there are no local hits.

  Supporting strategies: *flexible-store*, *dynamic*

- `ONLY_INTERCLOUD`:

  If `true`, orchestration process considers matching service instances only from the neighbor clouds. It automatically results `MATCHMAKING:true` as well.

  If `false`, orchestration process considers matching service instances from the local could in first hand and only considers matching service instances from the neighbor clouds when `ALLOW_INTERCLOUD:true`.

  Supporting strategies: *flexible-store*, *dynamic*

- `ALLOW_TRANSLATION`:

  If `true`, orchestration process considers matching, but non-native service instances when there are no native hits. Matching, but non-native service instance means that all the requirements are fulfilled except the interface related requirements (protocol, data-format, etc.). It automatically results `MATCHMAKING:true` as well.

  If `false`, orchestration process considers only native matching service instances.

  Supporting strategies: *flexible-store*, *dynamic*

3.8 struct QoSRequirementMap

An Object which maps String keys to String values.

3.9 struct NotifyInterface

Field	Type	Mandatory	Description
protocol	Protocol	yes	Communication protocol to be used for sending notification.
properties	NotifyInterfacePropertyMap	yes	Interface properties belonged to the specified protocol.

3.10 struct NotifyInterfacePropertyMap

An Object which maps String keys String values.

3.11 struct OrchestrationSubscriptionListResponse

Field	Type	Description
status	OperationStatus	Status of the operation
entries	List<OrchestrationSubscriptionResponse>	List of subscription records.
count	Number	Total number of subscription records.

3.12 struct OrchestrationSubscriptionResponse

Field	Type	Description
id	OrchestrationSubscriptionId	Unique identifier of the subscription record.
ownerSystemName	SystemName	Unique identifier of the requester system.
targetSystemName	SystemName	Unique identifier of the subscribed system.
orchestrationForm	OrchestrationForm	Orchestration request details.
notifyInterface	NotifyInterface	Interface details for sending push notifications.
expiredAt	DateTime	The subscription is active until this time.
createdAt	DateTime	The subscription was created at this timestamp.

3.13 struct OrchestrationSubscriptionId

Unique String identifier.

3.14 struct ErrorResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
errorMessage	String	Description of the error.
errorCode	Number	Numerical code of the error.
type	ErrorType	Type of the error.
origin	String	Origin of the error.

3.15 struct OrchestrationUnsubscribeListRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
subscriptionIds	List<OrchestrationSubscriptionId>	yes	List of subscription identifiers.

3.16 struct OrchestrationPushTriggerRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
targetSystems	List<SystemName>	no (yes)	List of target consumer systems. Mandatory if *subscriptionIds* are not provided.
subscriptionIds	List<OrchestrationSubscriptionId>	no (yes)	List of subscription identifiers. Mandatory if *targetSystems* are not provided.

3.17 struct OrchestrationPushJobListResponse

Field	Type	Description
jobs	List<OrchestrationJobResponse>	List of orchestration push jobs.

3.18 struct OrchestrationJobResponse

Field	Type	Description
id	OrchestrationJobId	Unique job identifier.
status	OrchestratioJobStatus	Actual working state of the job.
type	OrchestrationType	Type of orchestration.
requesterSystem	SystemName	Name of the system that started the orchestration process.
targetSystem	SystemName	Name of the system for which the orchestration is executed.
serviceDefinition	ServiceName	Name of the service that the orchestration job is targeting.
subscriptionId	OrchestrationSubscriptionId	Unique identifier of associated subscription record.
message	String	Additional error or warning information.
createdAt	DateTime	The job was created at this timestamp.
startedAt	DateTime	The job was started at this timestamp.
finishedAt	DateTime	The job was finished at this timestamp.

3.19 struct OrchestrationJobId

Unique String identifier.

3.20 struct OrchestrationSubscriptionQueryRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
pageNumber	Number	no (yes)	The number of the requested page. It is mandatory, if page size is specified.
pageSize	Number	no (yes)	The number of entries on the requested page. It is mandatory, if page number is specified.
pageSortField	String	no	The identifier of the field which must be used to sort the entries.
pageDirection	Direction	no	The direction of the sorting.
ownerSystems	List<SystemName>	no	Requester is looking for subscriptions with any of the specified creator system names.
targetSystems	List<SystemName>	no	Requester is looking for subscriptions with any of the specified target consumer system names.
serviceDefinitions	List<ServiceName>	no	Requester is looking for subscriptions with any of the specified service definition names.

3.21 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

Type	Description
AddressType	Any suitable type chosen by the implementor of service.
DateTime	Pinpoints a specific moment in time.
ErrorType	Any suitable type chosen by the implementor of service.
InterfaceName	A string identifier of an interface descriptor. Must follow snake_case naming convention.
List<A>	An *array* of a known number of items, each having type A.
Number	Decimal number.
Object	Set of primitives and possible further objects.
OperationStatus	Logical, textual or numerical value that indicates whether an operation is a success or a failure. Multiple values can be used for success and error cases to give additional information about the nature of the result.
OrchestrationJobStatus	Predefined values indicating working states.
OrchestrationType	Predefined values indicating orchestration type (*pull* or *push*).
Protocol	A string representation of a communication protocol.
SecurityPolicy	Any suitable security policy chosen by the implementor of service.
ServiceName	A string identifier that is intended to be both human and machine-readable. Must follow camelCase naming convention.
ServiceOperationName	A string identifier that is intended to be both human and machine-readable. Must follow kebab-case naming convention.
String	A chain of characters.
SystemName	A string identifier that is intended to be both human and machine-readable. Must follow PascalCase naming convention.
Version	Specifies a service instance version. Version must follow the Semantic Versioning.

4 References

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	YYYY-MM-DD	5.0.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	5.0.0	