Arrowhead Orchestrator Core Service 4.1.3 Release Notes

In Arrowhead version 4.1.3, the whole codebase of the Arrowhead framework has been rewritten using Spring Boot technology. We believe the new version is better both in code quality, performance and robustness. It is easier to use, maintain and improve.
However, this means that Arrowhead version 4.1.3 is NOT compatible with providers and consumers written for version 4.1.2 because the changes affect not just the backend but the public interface of the core services, too.
This document describes the new public API of the Arrowhead Orchestrator Core Service 4.1.3 and shows the key differences between the new interface and the old one.

Client Services

These services can be used by providers and consumers of the Arrowhead Cloud. All date fields contain the text representation of a UTC timestamp.

Uri: /orchestrator/echo
Type: GET

Returns a “Got it” message with the purpose of testing the core service availability.

4.1.2 version: GET /orchestrator/orchestration
It was basically the same with a slightly different return message.

Uri: /orchestrator/orchestration
Type: POST

Initializes the orchestration process in which the Orchestrator Core System tries to find providers that match the specified requirements (and the consumer have right to use them).

Input JSON structure:

```
{
  "requesterSystem": {
    "systemName": "string",
    "address": "string",
    "port": 0,
    "authenticationInfo": "string"
  },
  "requestedService": {
    "serviceDefinitionRequirement": "string",
    "interfaceRequirements": [
      "string"
    ],
    "securityRequirements": [
      "NOT_SECURE", "CEERTIFICATE", "TOKEN"
    ],
    "metadataRequirements": {
      "additionalProp1": "string",
```

```
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "versionRequirement": 0
    "maxVersionRequirement": 0,
    "minVersionRequirement": 0
  },
  "preferredProviders": [
    {
      "providerCloud": {
        "operator": "string",
        "name": "string"
      },
      "providerSystem": {
        "systemName": "string"
        "address": "string",
        "port": 0
      }
    }
  ],
  "orchestrationFlags": {
    "additionalProp1": true,
    "additionalProp2": true,
    "additionalProp3": true
  }
}
```

Orchestrator can be used in two ways. The first one uses predefined rules (coming from the orchestrator store DB) to find the appropriate providers for the consumer. The second option is the dynamic orchestration in which case the core service searches the whole local cloud (and maybe some other clouds) to find matching providers.

Store orchestration:

- requester system is mandatory,
- requested service and all the other parameters are optional,
- if requested service is not specified, then this service returns the top priority local provider of all services contained by the orchestrator store database for the requester system.
- if requested service is specified, then you have to define the service definition and exactly one interface (all other service requirements are optional). In this case, it returns all accessible providers from the orchestrator store database that provides the specified service via the specified interface to the specified consumer.

Dynamic orchestration:

- requester system is mandatory,
- requested service is mandatory, but just the service definition part, all other parameters of the requested service are optional,
- all other parameters are optional

Orchestration flags:

- matchmaking: the service automatically selects exactly one provider from the appropriate providers (if any),
- metadataSearch: query in the Service Registry uses metadata filtering,
- onlyPreferred: the service filters the results with the specified provider list,
- pingProviders: the service checks whether the returning providers are online and remove the unaccessible ones from the results,
- overrideStore: Services uses dynamic orchestration if this flag is true, otherwise it uses the

orchestration store,

- enableInterCloud: the service can search another clouds for providers if none of the local cloud providers match the requirements,
- triggerInterCloud: the service skipped the search in the local cloud and tries to find providers in other clouds instead.

Returned JSON structure:


```
{
  "response": [
    {
      "provider": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "service": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceUri": "string",
      "secure": "TOKEN",
      "metadata": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "interfaces": [
        {
          "id": 0,
          "createdAt": "string",
          "interfaceName": "string",
          "updatedAt": "string"
        }
      ],
      "version": 0,
      "authorizationTokens": {
        "interfaceName1": "token1",
        "interfaceName2": "token2",
      },
      "warnings": [
        "FROM_OTHER_CLOUD", "TTL_UNKNOWN"
      ]
    }
  ]
}
```


- authorization tokens object only appears if the provider requires token authentication,
- authorization token is interface-specific
- warnings array can contains the following texts: FROM_OTHER_CLOUD (if the provider is in an other cloud), TTL_EXPIRED (the provider is no longer accessible), TTL_EXPIRING (the provider will be unaccessible in a matter of minutes), TTL_UNKNOWN (the provider does not specified expiration time)

4.1.2 version: POST /orchestrator/ochestration
It was basically the same, however security requirement was not available.

Uri: /orchestrator/orchestration/{id}
Type: GET

If the consumer knows its id, it can used this service as shortcut for store-based orchestration when the service returns the top priority local provider of all services contained by the orchestrator store database for the requester system (identified by the id).

Returned JSON structure:

```
{
  "response": [
    {
      "provider": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "service": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceUri": "string",
      "secure": "CERTIFICATE",
      "metadata": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "interfaces": [
        {
          "id": 0,
          "createdAt": "string",
          "interfaceName": "string",
          "updatedAt": "string"
        }
      ],
      "version": 0
      "warnings": [
        "TTL_UNKNOWN"
      ]
    }
  ]
}
```

Management Services

These services can only be used by the system operator of the local cloud. All date fields contain the text representation of a UTC timestamp.

Uri: /orchestrator/mgmt/store/echo

Type: GET

Returns a “Got it” message with the purpose of testing the core service availability.

4.1.2 version: GET /orchestrator/mgmt/store
It was basically the same with a slightly different return message.

Uri: /orchestrator/mgmt/store
Type: GET
Query params:

- *page* – zero-based page index (optional),
- *item_per_page* – maximum number of items returned (optional),
- *sort_field* – sort field (optional, default: id, possible values: id, createdAt, updatedAt, priority),
- *direction* – direction of sorting (optional, default: ASC, possible values: ASC or DESC)
-

Returns a page of orchestrator store rule records. If page and item_per_page are not defined, returns all records.

Returned JSON structure:


```
{
  "count": 0,
  "data": [
    {
      "id": 0,
      "serviceDefinition": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "consumerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "foreign": true,
      "providerCloud": {
        "id": 0,
        "operator": "string",
        "name": "string",
        "authenticationInfo": "string",
        "secure": true,
        "neighbor": true,
        "ownCloud": false,
        "createdAt": "string",
        "updatedAt": "string"
      },
      "providerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
```

```
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceInterface": {
        "id": 0,
        "interfaceName": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "priority": 1,
      "attribute": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "createdAt": "string",
      "updatedAt": "string"
    }
  ]
}
```

Rules are a little stricter than before: the service interface is also part of it. But the defaultEntry flag is no longer supported; now, entries with priority 1 is considered as defaults.

4.1.2 version: GET /orchestrator/mgmt/store/all
This version always returned all records in an array of JSON objects. The objects did not contain any time information. Rules didn't depend on interface.

Uri: /orchestrator/mgmt/store
Type: POST

Creates orchestrator store rule records and returns the newly created records.

Input JSON structure:

```
[
  {
    "serviceDefinitionName": "string",
    "consumerSystemId": 0,
    "attribute": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "providerSystemDTO": {
      "systemName": "string",
      "address": "string",
      "port": 0
    },
    "cloudDTO": {
      "operator": "string",
      "name": "string"
    },
    "serviceInterfaceName": "string"
    "priority": 1,
  }
]
```

The cloudDTO parameter is only specified if the provider system is not from the local cloud.

Returned JSON structure:


```
{
  "count": 0,
  "data": [
    {
      "id": 0,
      "serviceDefinition": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "consumerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "foreign": true,
      "providerCloud": {
        "id": 0,
        "operator": "string",
        "name": "string",
        "authenticationInfo": "string",
        "secure": true,
        "neighbor": true,
        "ownCloud": false,
        "createdAt": "string",
        "updatedAt": "string"
      },
      "providerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceInterface": {
        "id": 0,
        "interfaceName": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "priority": 1,
      "attribute": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "createdAt": "string",
      "updatedAt": "string"
    }
  ]
}
```


4.1.2 version: POST /orchestrator/mgmt/store

This version required whole JSON objects as consumer instead of id and didn't contains interface names. Also, it used defaultEntry flags.

Uri: /orchestrator/mgmt/store/{id}
Type: GET

Returns the orchestrator store rule record specified by the id path parameter.

Returned JSON structure:


```
{
  "id": 0,
  "serviceDefinition": {
    "id": 0,
    "serviceDefinition": "string",
    "createdAt": "string",
    "updatedAt": "string"
  },
  "consumerSystem": {
    "id": 0,
    "systemName": "string",
    "address": "string",
    "port": 0,
    "authenticationInfo": "string",
    "createdAt": "string",
    "updatedAt": "string"
  },
  "foreign": true,
  "providerCloud": {
    "id": 0,
    "operator": "string",
    "name": "string",
    "authenticationInfo": "string",
    "secure": true,
    "neighbor": true,
    "ownCloud": false,
    "createdAt": "string",
    "updatedAt": "string"
  },
  "providerSystem": {
    "id": 0,
    "systemName": "string",
    "address": "string",
    "port": 0,
    "authenticationInfo": "string",
    "createdAt": "string",
    "updatedAt": "string"
  },
  "serviceInterface": {
    "id": 0,
    "interfaceName": "string",
    "createdAt": "string",
    "updatedAt": "string"
  },
  "priority": 1,
  "attribute": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "createdAt": "string",
  "updatedAt": "string"
```

```
}
```

4.1.2 version: GET /orchestrator/mgmt/store/{id}
The returned structure did not contain time information and interface names.

Uri: /orchestrator/mgmt/store/{id}
Type: DELETE

Removes the orchestrator store rule record specified by the id path parameter.

4.1.2 version: DELETE /orchestrator/mgmt/store/{id}
It was basically the same than the new version.

Uri: /orchestrator/mgmt/store/all_by_consumer
Type: GET
Query params:

- *page* – zero-based page index (optional),
- *item_per_page* – maximum number of items returned (optional),
- *sort_field* – sort field (optional, default: id, possible values: id, createdAt, updatedAt),
- *direction* – direction of sorting (optional, default: ASC, possible values: ASC or DESC)

Returns a page of orchestrator store rule records related to consumer, service definition and optionally service interface. If page and item_per_page are not defined, no paging is involved.

Input JSON structure:

```
{
  "consumerSystemId": 0,
  "serviceDefinitionName": "string",
  "serviceInterfaceName": "string"
}
```

Consumer system id and service definition name are mandatory parameters, the service interface name is optional.

Returned JSON structure:

```
{
  "count": 0,
  "data": [
    {
      "id": 0,
      "serviceDefinition": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "consumerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
```

```
      "createdAt": "string",
      "updatedAt": "string"
    },
    "foreign": true,
    "providerCloud": {
      "id": 0,
      "operator": "string",
      "name": "string",
      "authenticationInfo": "string",
      "secure": true,
      "neighbor": true,
      "ownCloud": false,
      "createdAt": "string",
      "updatedAt": "string"
    },
    "providerSystem": {
      "id": 0,
      "systemName": "string",
      "address": "string",
      "port": 0,
      "authenticationInfo": "string",
      "createdAt": "string",
      "updatedAt": "string"
    },
    "serviceInterface": {
      "id": 0,
      "interfaceName": "string",
      "createdAt": "string",
      "updatedAt": "string"
    },
    "priority": 1,
    "attribute": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "createdAt": "string",
    "updatedAt": "string"
  }
 ]
}
```


4.1.2 version: PUT /orchestrator/mgmt/store
This version always returned all matching records in an array of JSON objects. The objects did not contain any time information. Of course, filtering by interface name was not available.

Uri: /orchestrator/mgmt/store/all_top_priority
Type: GET
Query params:

- *page* – zero-based page index (optional),
- *item_per_page* – maximum number of items returned (optional),
- *sort_field* – sort field (optional, default: id, possible values: id, createdAt, updatedAt),
- *direction* – direction of sorting (optional, default: ASC, possible values: ASC or DESC)

Returns a page of orchestrator store rule records whose priority is 1. If page and item_per_page are not defined, no paging is involved.

Returned JSON structure:


```
{
  "count": 0,
  "data": [
    {
      "id": 0,
      "serviceDefinition": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "consumerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "foreign": true,
      "providerCloud": {
        "id": 0,
        "operator": "string",
        "name": "string",
        "authenticationInfo": "string",
        "secure": true,
        "neighbor": true,
        "ownCloud": false,
        "createdAt": "string",
        "updatedAt": "string"
      },
      "providerSystem": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceInterface": {
        "id": 0,
        "interfaceName": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "priority": 1,
      "attribute": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "createdAt": "string",
      "updatedAt": "string"
    }
  ]
}
```


4.1.2 version: GET /orchestrator/mgmt/store/all_default

This version always returned all records where defaultEntry flag is true in an array of JSON objects. The objects did not contain any time information.

Uri: /orchestrator/mgmt/store/modify_priorities
Type: POST

Changes the priority field of the specified rules.

Input JSON structure:

```
{
  "priorityMap": {
    "id1": 1,
    "id2": 2,
    "id3": 3
  }
}
```

The keys of the map are orcherstrator store rule ids, the values are the new priorities.

4.1.2 version: PUT /orchestrator/mgmt/store/priorities
It was basically the same.

The following services are no longer exist:

- GET /orchestrator/mgmt/store/default/{id}
- PUT /orchestrator/mgmt/store/update/{id}
- DELETE /orchestrator/mgmt/store/consumerId/{systemId}

Private Services

These services can only be used by other core services, therefore they are not part of the public API.