NPW3C2011-003

ORIENT Software Development Kit User's Manual - For Application Developer –

(Revision 1.0)

NEC Corporation

Overview of this document: The usage method of API of ORIENT SDK for AP developer is shown with the help of processing method and sample code.

1. Introduction

This document describes the usage method of API of ORIENT SDK for an application developer.

1.1 Objective

The objective is to explain usage method of API of ORIENT SDK for an application developer with the help of processing method and sample code to develop an understanding about usage method of ORIENT SDK.

1.2 Target

This document is meant for an application developer, who is the user of API of ORIENT SDK.

1.3 Scope

This document explains usage method of API, which is used by request sender of ORIENT SDK. However, it does not explain procedure for creating DSM resource. The details regarding creation of DSM resource are provided in “OSDK-GUIDE-DSMResourceCreation-APIUsageGuide”.

1.4 Explanation of terminologies

- DSM(Domain Specific Model)

  DSM defines the following three items.
  1. Name space URI: URI and its abbreviations to identify DSM itself
  2. Model Scheme: Vocabulary definition required to express Entity
  3. Reference Model: Model which enables to refer to structure of model that expresses Entity

- ORIENT

  Following three items are defined regarding RDF graph structure and controlled vocabulary for ARMOR interface.
  1. Operation specific data (Operation)
  2. Restriction specific data (Restriction)
  3. Analysis result specific data (Entities)

1.5 Related documents

- ORIENT Software Development Kit User's Manual

  1. OSDKManual-ForAnalyzerDeveloper.doc
  2. OSDKManual-ForModelDesigner.doc
  3. OSDKManual-ForDataOperator.doc

- ORIENT Specifications

  1. ORIENTSpecification.doc

1.6 Notes

- Exception handling
  - Exception used in ORIENT SDK is the subclass of RuntimeException. Exception handling is carried out if error handling is required to catch the occurring RuntimeException
  - IllegalArgumentException occurs when input value of argument of method is null. Though it is not mentioned in Throws, Exception handling is carried out if error handling is required
- Thread safety
  - Thread safe items are mentioned as thread safe in JavaDoc. Attention should be paid to the items having no description as they are not thread-safe

2. Overall guidelines of API usage of ORIENT message creation

All the usage guidelines of API, which are common to the guidelines used at the time of creating ORIENT resource of ORIENT using API of ORIENT SDK, are explained below.

2.1 Message and ORIENT resource creation responsibility

Entire message

- ORIENTMethod is the class, which indicates entire message of ORIENT. Object indicating entire message can be created by generating required Operation Resource (Query, Write and Notify) from ORIENTMethodFactory.

Class that can create ORIENT Resource

- ORIENT resource is created and configured for created ORIENT Method. In the below comparative table, Sources of each resource and the Resources that can be created from them are shown. However, creation of only the resources, related to resource which is to be generated, is possible. (Refer to Table 1 Comparative table of resource creation (Which class creates which resource).

Table 1 Comparative table of resource creation (Which class creates which resource)

Source of resource creation	Possible resource creation
ORIENTMethod	Condition
	Scope
	Entities
	ProcessFlow
	EngineOperationStatus
	OperationStatus
Condition	Each Filter
Entities	Entity
ProcessFlow	ProcessPhase
ProcessPhase	EngineOperation

2.2 Guidelines related to assembling method of resources

- ORIENT resource has to be configured in target ORIENT resource after creation.
  - For example, the relation between Entities and Entity is 1: many. However, when Entity is created from Entities, addition of created Entity (set Type APIaddEntity) has to be called for Entities resource.
- The API, used at the time of configuring ORIENT resource, configures resource in “API (set, add) of set type)”.
  - When resource related to created resource is configured, it is a prerequisite condition that necessary data property and object property of resource to be configured is set in the resource. Exception occurs in case settings (configuration of ORIENT resource in setTypeAPI) are configured in creation resource without configuring necessary properties of resource. It is for preventing creation of incomplete resource.

3. Overall procedure for creation of ORIENT method and transmission processing

In this section, the entire processing method wherein engine developer creates ORIENT method, transmits it to ARMOR and receives response from ARMOR is explained with the help of processing method and sample code.

Detailed procedure of creation method, etc. of each resource is explained in the following sections.

3.1 Request and response of Query Operation

3.1.1 Creation method of request and response of Query Operation (in case of transmission to ARMOR)

Method to create a request transmitted to ARMOR by user application and method to receive response from ARMOR is explained below.

(1) Processing method

Request and response are created by the following processing method.

1) ORIENTMethodFactory is created.

2) ORIENTMethod of Query is created.

3) DSM to be used is configured.

4) Condition is created.

5) Filter, which specifies search conditions, is created. In the sample code mentioned in the following section, it is to search whether the information containing string of “Central Labs” exists in resource corresponding to specified URI, or not.

6) Search conditions (Filter) are configured in Condition.

7) Condition is configured in ORIENTMethod.

8) Filter, which specifies sequence order of search results, is created. In the sample code of following section, the resource values of URI specified from search results are sorted in ascending order and first and second items are specified to be obtained.

9) Scope is created.

10) The information of properties, which you want as search result, is configured in scope.

11) Sequence order (Filter) of search results is configured in scope.

12) Scope is configured in ORIENTMethod.

13) ORIENTMethodSenderFactory is created.

14) OrientMethodSender is created.

15) Request is sent and response is obtained.

16) It is checked if the Status code configured in response is 200 (successful completion).

17) If the Status code is 200, the Entities configured in response are obtained.

18) Entity is obtained from Entities

19) Acquisition fails if Status code is other than 200.

(2) Sample code using API

The sample code shown below is the code which is used at the time of generating Query request given in (1) Query for Analysis Result and obtaining response of (4) Analysis Results Response in Section 8.1.5 of ORIENT Specifications.

```
try{
   /** 1) Create ORIENTMethodFactory. */
   ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

   /** 2) Create ORIENTMethod of Query. */
   ORIENTMethod method = orientMethodFactory.createQueryMethod(
           new URI("http://example.com/operation/1"), new URI(
                   "http://www.nec.com/ivcp/example/query/processflow"));

   /** 3) Set DSM to be used. */
   DSM personDSM = new PersonDSM();
   method.useDSM(personDSM);

   /** 4) Create Condition. */
   Condition condition = method.createCondition();

   /** 5) Create Filter, which specifies search conditions. */
   /** Create ContainsFilter */
   /** Condition mentioned below is to search whether information containing string of */
   /** “Central Labs” exists in resource corresponding to specified URI or not */
   ContainsFilter containsFilter = condition
       .createContainsFilter(
            new URI(
                "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.org/ds.Organization
                 new LiteralValueImpl(XSDDataType.STRING, "Central Labs"));

    /** 6) Set search conditions (Filter) in Condition.*/
    condition.setFilter(containsFilter);

    /** 7) Set Condition in ORIENTMethod. */
    method.setCondition(condition);

    /** 8) Create Filter, which specifies sequence order of search results. */
    /** Create OrderByFilter*/
    /** Condition mentioned below is to sort search results by value of resources corresponding to URI in */
    /** ascending order and then, obtain values of item 1 and 2 */
    OrderByFilter orderByFilter = condition
           .createOrderByFilter(
                   new URI(
                        "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday"),
                         1, 2, ORDER.ASCENDING);

    /** 9) Create Scope.*/
    Scope scope = method.createScope();

    /** 10) Set the information of properties, which you want as search result, in scope. */
    scope.setEntityType(PersonDSMResourceTypes.PERSON);
    String baseURI = "http://www.nec.com/ivcp/orient/da/general/ri/";
    URI[] SCOPE_RESOURCES = {
               new URI(baseURI + "ds.person"),
               new URI(baseURI + "ds.person/vCard.fn"),
               new URI(baseURI + "ds.person/vCard.org/ds.Organization"),
               new URI(baseURI+ "ds.person/vCard.org/ds.Organization/vCard.orgunit") };
    scope.addScopeReferenceModelNodeURIs(SCOPE_RESOURCES);

    /** 11) Set sequence order (Filter) of search results in scope.*/
    scope.setFilter(orderByFilter);
```

```
    /** 12) Set Scope in ORIENTMethod. */
    method.addScope(scope);

    /** 13) Create ORIENTMethodSenderFactory. */
    ORIENTMethodSenderFactorysenderFactory = new ORIENTMethodSenderFactoryImpl ();

    /** 14) Create OrientMethodSender. */
      OrientMethodSender sender = senderFactory.create();

    /** 15) Send request and obtain response. */
    ORIENTMethod response = sender.sendRequestSync(new URI(
                getServletContainerUrl() + sOperationPath), method,
                ORIENTConstants.RDF_XML_MESSAGE_TYPE);

    /** 16) Check if Status code configured in response is 200 (successful completion).*/
    if (response.getOperationStatus().getCode() == 200) {
          /** 17) If Status code is 200, obtain the Entities configured in response. */
          Set<Entities> entitiesSet = response.getAllEntities();
          for (Entities entities : entitiesSet) {
              /** 18) Obtain Entity from Entities*/
                List<Entity> entityList = entities.getAllEntities();
                Iterator<Entity> entityIterator = entityList.iterator();
                while (entityIterator.hasNext()) {
                      Entity entity = entityIterator.next();
                }
          }
    } else {
          /** 19) Acquisition fails if Status code is other than 200. */
    }
  }catch(OrientRuntimeException e){
       /**Exception handling is carried out appropriately by catching subclass */
  }
```

3.2 Request and response of Write Operation

3.2.1 Creation method of request and response of Write Operation

The method to create a request transmitted to ARMOR by Online Engine and to receive response from ARMOR is explained below.

(1) Processing method

Request and response are created by following processing method.

1) ORIENTMethodFactory is created.
2) ORIENTMethod of Write is created.
3) DSM to be used is configured.
4) Entities are created.
5) List to be saved in Entity is created.
6) Entity is created.
7) Entity is configured in List.
8) Entity is added in Entities collectively.
9) ORIENTMethod is configured in Entities.
10) ORIENTMethodSenderFactory is created.
11) OrientMethodSender is created.
12) Request is sent and response is obtained.
13) It is checked if the Status code configured in response is 200 (successful completion).
14) There is no response if Status code is 200.
15) It will be regarded as failure if Status code is other than 200.

(2) Sample code using API

The sample code shown below is the code, which is used at the time of creating Write request, mentioned in (1) Data write request and obtaining response, mentioned in (2) Response in Section 8.2.4 of ORIENT Specifications.

```
try {
	/** 1) Create ORIENTMethodFactory. */
	ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

	/** 2) Create ORIENTMethod of Write. */
	ORIENTMethod method = orientMethodFactory
			.createWriteMethod(new URI("http://example.com/operation/1"));

	/** 3) Set DSM to be used. */
	DSM personDSM = new PersonDSM();
	method.useDSM(personDSM);
	DSMResourceFactory dsmResourceFactory = method
			.getDSMResourceFactory();

	/** 4) Create Entities */
	Entities entities = method.createEntities(RDF.SEQ);

	/** 5) Create List to be saved in Entity. */
	List<Entity> persones = new ArrayList<Entity>();

	/** 6) Create Entity */
	// person(person 1) resource creation
	Entity person1 = entities.createEntity(
			PersonDSMResourceTypes.PERSON, new URI(
				"mailto:nippon-denki@dummy.nec.com"));
	person1.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Tarou");
	person1.setDataProperty(PersonDSMProperties.VCARD_NOTE, "Researcher");
	person1.setDataProperty(PersonDSMProperties.VCARD_BDAY,
			new DateTime(1968, 6, 1, 0, 0, 0, 0));

	// organization(person 1) resource creation.
	Resource organization1 = dsmResourceFactory.createResource(
			PersonDSMResourceTypes.ORGANIZATION, new URI(
				"http://www.nec.com/ivcp/orient/org/nec/kmg"));
	organization1.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
			"ABC Labs");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "NEC");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "Development and Research Unit");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "Central Labs");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "ABC Labs");
	person1.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
			organization1);

	/** 7) Set Entity in List. */
	persones.add(person1);
```

```
/** 6) Create Entity */
// person(person 2) resource creation
Entity person2 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                "mailto:t-nippondenki@dummy.nec.com"));
person2.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Hanako");
person2.setDataProperty(PersonDSMProperties.VCARD_NOTE, "Researcher");
person2.setDataProperty(PersonDSMProperties.VCARD_BDAY,
        new DateTime(1982, 11, 1, 0, 0, 0, 0));

// organization(person 2) resource creation.
Resource organization2 = dsmResourceFactory.createResource(
        PersonDSMResourceTypes.ORGANIZATION, new URI(
                "http://www.nec.com/ivcp/orient/org/nec/ctl"));
organization2.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
        "XYZ Labs");
organization2.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "NEC");
organization2.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "Development and Research Unit");
organization2.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "Central labs");
organization2.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "XYZ Labs");
person2.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization2);

/** 7) Set Entity in List. */
persones.add(person2);

/** 6) Create Entity */
// person(person 3) resource creation
Entity person3 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                "mailto:j-nichiden@dummy.nec.com"));
person3.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Jirou");
person3.setDataProperty(PersonDSMProperties.VCARD_NOTE, "PC Incharge");
person3.setDataProperty(PersonDSMProperties.VCARD_BDAY,
        new DateTime(1983, 1, 1, 0, 0, 0, 0));

//organization(person 3) resource creation
Resource organization3 = dsmResourceFactory.createResource(
        PersonDSMResourceTypes.ORGANIZATION, new URI(
                "http://www.nec.com/ivcp/orient/org/nec/pc"));
organization3.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
        "NEC Group");
organization3.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "NEC");
organization3.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "PC Division");
person3.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization3);
```

```
        /** 7) Set Entity in List. */
        persones.add(person3);

        /** 8) Set Entity in Entities. */
        entities.addEntities(persones);

        /** 9) Set ORIENTMethod in Entities. */
        method.addEntities(entities);

        /** 10) Create ORIENTMethodSenderFactory. */
        ORIENTMethodSenderFactory senderFactory = new ORIENTMethodSenderFactoryImpl();

        /** 11) Create OrientMethodSender. */
        OrientMethodSender sender = senderFactory.create();

        /** 12) Send request and obtain response. */
        ORIENTMethod response = sender.sendRequestSync(new URI(
                    getServletContainerUrl() + sOperationPath), method,
                    ORIENTConstants.RDF_XML_MESSAGE_TYPE);

        sLog.info("Response is    ¥n" + response.serialize());

        /** 13) Check if Status code configured in response is 200 (successful completion). */
        if (response.getOperationStatus().getCode() == 200) {
            /** 14) There is no response if Status code is 200. */
        } else {
            /** 15) It will be regarded as failure if Status code is other than 200. */
        }

} catch (OrientRuntimeException e) {
        /** Exception handling is carried out appropriately by catching subclass */
}
```

3.3 Notify Operation request

3.3.1 Creation method of Notify Operation request

Method to create a request, which ARMOR transmits to Controller, is explained below.

(1) Processing method

Request and response are created by following processing method.

1) ORIENTMethodFactory is created.

2) ORIENTMethod of Notify is created.

3) DSM to be used is configured.

4) Entities are created.

5) List to be saved in Entity is created.

6) Entity is created.

7) Entity is configured in List.

8) Entity is added in Entities collectively.

9) ORIENTMethod is configured in Entities.

10) ORIENTMethodSenderFactory is created.

11) OrientMethodSender is created.

12) Request is sent.

(2) Sample code using API

The sample code shown below is the code used for creating Notify request mentioned in (1) Notify of Section A.3.4 of ORIENT Specifications.

```
try {
	/** 1) Create ORIENTMethodFactory. */
	ORIENTMethodFactoryImpl orientMethodFactory = new ORIENTMethodFactoryImpl();

	/** 2) Create ORIENTMethod of Notify. */
	ORIENTMethod method = orientMethodFactory
			.createNotifyMethod(
					new URI("http://example.com/operation/1"),
					new URI(
							"http://orient_example.com/example/notify/processflow"),
					new URI[] { new URI(
							"http://example.com/notify_receiver") });

	/** 3) Set DSM to be used. */
	DSM personDSM = new PersonDSM();
	method.useDSM(personDSM);

	/** 4) Create Entities. */
	Entities entities = method.createEntities(RDF.SEQ);

	DSMResourceFactory dsmResourceFactory = method
			.getDSMResourceFactory();

	Resource organization1 = dsmResourceFactory.createResource(
			PersonDSMResourceTypes.ORGANIZATION, new URI(
					"http://www.nec.com/ivcp/orient/org/nec/pc"));
	organization1.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
			"NEC Group");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "NEC");
	organization1.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "PC Division");

	Resource organization2 = dsmResourceFactory.createResource(
			PersonDSMResourceTypes.ORGANIZATION, new URI(
					"http://www.nec.com/ivcp/orient/org/nec/crl"));
	organization2.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
			"XYZ Labs");
	organization2.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "NEC");
	organization2.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "Development and Research Unit");
	organization2.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "Central Labs");
	organization2.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "XYZ Labs");

	Resource organization3 = dsmResourceFactory.createResource(
			PersonDSMResourceTypes.ORGANIZATION, new URI(
					"http://www.nec.com/ivcp/orient/org/nec/kmg"));
	organization3.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
			"ABC Labs");
	organization3.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "NEC");
	organization3.addDataPropertyToContainer(
			PersonDSMProperties.VCARD_ORGUNIT, "Development and Research Unit");
```

```
organization3.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "Central Labs");
organization3.addDataPropertyToContainer(
        PersonDSMProperties.VCARD_ORGUNIT, "ABC Labs");

/** 5) Create List in which Entity is saved */
List<Entity> persones = new ArrayList<Entity>();

/** 6) Create Entity */
Entity person1 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                " mailto:nippon-denki@dummy.nec.com"));
person1.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Tarou");
person1.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization3);
person1.setDataProperty(PersonDSMProperties.VCARD_NOTE, "Researcher");
person1.setDataProperty(PersonDSMProperties.VCARD_BDAY,
        new DateTime(1968, 6, 1, 0, 0, 0, 0, DateTimeZone.UTC));

/** 7) Set Entity in List. */
persones.add(person1);

/** 6) Create Entity */
Entity person2 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                " mailto:t-nippondenki@dummy.nec.com"));
person2.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Hanako");
person2.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization2);
person2.setDataProperty(PersonDSMProperties.VCARD_NOTE, "Researcher");
person2.setDataProperty(PersonDSMProperties.VCARD_BDAY,
        new DateTime(1982, 11, 1, 0, 0, 0, 0, DateTimeZone.UTC));

/** 7) Set Entity in List. */
persones.add(person2);

/** 6) Create Entity */
Entity person3 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                "mailto:j-nichiden@dummy.nec.com"));
person3.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Jirou");
person3.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization1);
person3.setDataProperty(PersonDSMProperties.VCARD_NOTE, "PC Incharge");
person3.setDataProperty(PersonDSMProperties.VCARD_BDAY,
        new DateTime(1983, 1, 1, 0, 0, 0, 0, DateTimeZone.UTC));

/** 7) Set Entity in List. */
persones.add(person3);

/** 8) Set Entity in Entities. */
entities.addEntities(persones);

/** 9) Set ORIENTMethod in Entities. */
method.addEntities(entities);
```

```
        /** 10) Create ORIENTMethodSenderFactory. */
        ORIENTMethodSenderFactory senderFactory = new ArmorORIENTMethodSenderFactoryImpl();

        /** 11) Create OrientMethodSender. */
        OrientMethodSender sender = senderFactory.create();

        /** 12) Send request. */
        sender.sendRequestAsync(new URI(getServletContainerUrl()
                + sOperationPath), method,
                ORIENTConstants.RDF_XML_MESSAGE_TYPE);
} catch (OrientRuntimeException e) {
        /** Exception handling is carried out appropriately by catching subclass */
}
```

4. Details of creation procedure of ORIENT message (ORIENTMethod)

Orient Method, which indicates entire ORIENT message, and its creation method are explained below. Hiding of ORIENT Specifications is carried out by sending and receiving this ORIENTMethod.

4.1 Creation procedure of ORIENT Method

4.1.1 Processing method

Processing method of ORIENTMethod is shown below

1) ORIENTMethodFactory is created.

2) ORIENTMethod corresponding to Operation is created. The relation between Operation type and API to be used is shown in table below.

Operation Type	API
Query	createQueryMethod
Write	createWriteMethod
Notify	createNotifyMethod

4.1.2 Sample code using API

(1) Query

The sample code shown below is the code used when transmission is done by Query Operation (in case URI of ProcessFlow property is not specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Query. */
ORIENTMethod method = orientMethodFactory
        .createQueryMethod(new URI("http://example.com/operation/1"));
```

The sample code shown below is the code used when transmission is done by Query Operation (in case URI of ProcessFlow property is specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Query. */
ORIENTMethod method = orientMethodFactory.createQueryMethod(
        new URI("http://example.com/operation/1"), new URI(
                "http://www.nec.com/ivcp/example/query/processflow"));
```

(2) Write

The sample code shown below is the code used when transmission is done by Write Operation (in case URI of ProcessFlow property is not specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Write. */
ORIENTMethod method = orientMethodFactory
        .createWriteMethod(new URI("http://example.com/operation/1"));
```

The sample code shown below is the code used when transmission is done by Write Operation (in case URI of ProcessFlow property is specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Write. */
ORIENTMethod method = orientMethodFactory.createQueryMethod(
        new URI("http://example.com/operation/1"), new URI(
                "http://www.nec.com/ivcp/example/write/processflow"));
```

(3) Notify

The sample code shown below is the code used when transmission is done by Notify Operation (in case URI of ProcessFlow property is not specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Notify. */
ORIENTMethod method = orientMethodFactory
        .createNotifyMethod(new URI("http://example.com/operation/1"),
            new URI[] { new URI("http://example.com/notify_receiver") });
```

The sample code shown below is the code used when transmission is done by Notify Operation (in case URI of ProcessFlow property is specified).

```
/** 1) Create ORIENTMethodFactory. */
ORIENTMethodFactory orientMethodFactory = new ORIENTMethodFactoryImpl();

/** 2) Create ORIENTMethod of Notify. */
ORIENTMethod method = orientMethodFactory
        .createNotifyMethod(
                new URI("http://example.com/operation/1"),
                new URI("http://orient_example.com/example/notify/processflow"),
                new URI[] { new URI("http://example.com/notify_receiver") });
```

5. Creation procedure details of various ORIENT resources of ORIENT Specifications

Resource-wise explanation is given below regarding creation method of ORIENT resources, which are defined in ABF (ARMOR Base Framework), AOF (ARMOR Operation Framework) and ARF (ARMOR Restriction Framework), defined in ORIENT Specifications.

5.1 ABF (ARMOR Base Framework)

5.1.1 Creation procedure of Operation resource

(1) Processing method

Operation is automatically created by URI of Operation resource, which is specified in argument at the time of creating ORIENT Method (Refer to 5.1.2 Sample code using API). Owing to this Operation is hidden from API user and it is not directly visible to him.

URI of processFlow properties is specified in argument at the time of ORIENT Method creation, in case it is necessary to specify processFlow properties of Operation resource. However, ProcessFlow resource is not created only by configuring URI in processFlow properties at this level (Refer to "7.4 Usage method of ProcessFlow, ProcessPhase and EngineOperation" for creation of ProcessFlow resource).

5.1.2 Creation procedure of Restriction resource

(1) Processing method

Restriction is created automatically. Owing to this, Restriction is hidden from API user and it is not directly visible to him.

In case of Query Operation, Restriction resource is not created if generation and configuration of Condition or Scope, which are necessary resources, is not carried out (Refer to "7.5 Usage method of Condition" for creation and configuration of Condition. Refer to "7.6 Usage method of Scope" for creation and configuration of Scope). Therefore, error occurs if transmission is done without creating and configuring Condition or Scope.

In case of Notify Operation, URI of notification address is required to be specified in monitoredBy property of Restriction resource. Therefore, URI of notification address is specified in an array in argument at the time of ORIENT Method creation (Refer to (3) Notify of Section 5.1.2).

5.1.3 Creation procedure of Vocabulary resource

(1) Processing method

Vocabulary is automatically created when DSM to be used is configured in ORIENT Method by the following procedure. Owing to this, Vocabulary is hidden from API user and it is not directly visible to him.

```
/** Set DSM to be used in ORIENT Method */
method.useDSM(personDSM);
```

5.1.4 Creation procedure of Entities and Entity resource

(1) Creation procedure of Entities and Entity

(1-1) Processing method

Entities and Entity are created by the following method and are configured in ORIENT Method.

An error occurs if URI, which is specified at the time of Entity creation and creation of resource configured in Entity, has already been used.

1) Entities are created from ORIENT Method.

2) Entity is created from Entities.

3) DataProperty is configured in Entity.

4) Resource is configured in Entity. Refer to “7.4.2 Usage method of Resource configured in Entity” for creation of resource configured in Entity.

5) Entity is configured in Entities.

6) Entities are configured in ORIENT Method.

(1-2) Sample code using API

The sample code shown below is the code used for creation and configuration of Entities.

```
/** 1)Create Entities from ORIENT Method.*/
Entities entities = method.createEntities(RDF.SEQ);

/** 2)Create Entity from Entities. */
Entity person1 = entities.createEntity(
        PersonDSMResourceTypes.PERSON, new URI(
                "mailto:nippon-denki@dummy.nec.com"));

/** 3)Set DataProperty in Entity.*/
person1.setDataProperty(PersonDSMProperties.VCARD_FN, "NEC Tarou");


/** 4)Set Resource in Entity.*/
DSMResourceFactory dsmResourceFactory = method
        .getDSMResourceFactory();

Resource organization1 = dsmResourceFactory.createResource(
        PersonDSMResourceTypes.ORGANIZATION, new URI(
                "http://www.nec.com/ivcp/orient/org/nec/kmg"));
organization1.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
        "ABC Labs");
person1.setResource(PersonDSMProperties.OBJ_VCARD_ORG,
        organization1);

/** 5) Set Entity in Entities.*/
entities.addEntity(person1);

/** 6) Set Entities in ORIENT Method. */
method.addEntities(entities);
```

(2) Creation procedure of DSM Resource configured in Entity resource

(2-1) Processing method

In case of configuring Resource in Entity, setResource is used. The Resource to be configured in Entity is created by the following method.

1) DSMResourceFactory is created from ORIENT Method.

2) Resource is created from DSMResourceFactory.

3) DataProperty is configured in created Resource.

(2-2) Sample code using API

The sample code shown below is the code used for creation of Resource, which is to be configured in Entity.

```
/** 1) Create DSMResourceFactory from ORIENT Method.*/

DSMResourceFactory dsmResourceFactory = method
        .getDSMResourceFactory();

/** 2) Create Resource from DSMResourceFactory.*/
Resource organization1 = dsmResourceFactory.createResource(
        PersonDSMResourceTypes.ORGANIZATION, new URI(
            "http://www.nec.com/ivcp/orient/org/nec/kmg"));

/** 3)Set DataProperty in created Resource.
organization1.setDataProperty(PersonDSMProperties.VCARD_ORG_NAME,
        "ABC Labs");
```

5.2 Details of creation procedure of various ORIENT resources of AOF (ARMOR Operation Framework)

5.2.1 Creation procedure of ProcessFlow, ProcessPhase and EngineOperation

(1) Processing method

An error occurs if Resource URI, which is to be specified in Argument, has already been used for ProcessFlow, ProcessPhase and EngineOperation.

ProcessFlow, ProcessPhase and EngineOperation are created by the following method and are configured in ORIENT Method.

1) ProcessFlow is created from ORIENT Method.

2) ProcessPhase is created from ProcessFlow generated in 1).

3) EngineOperation is created from ProcessPhase generated in 2).

4) EngineOperation is configured in ProcessPhase.

5) ProcessPhase is configured in ProcessFlow. At that time, an error occurs in case EngineOperation is not configured in ProcessPhase.

6) ProcessFlow is configured in ORIENT Method. At that time, an error occurs in case ProcessPhase and EngineOperation are not configured in ProcessFlow.

(2) Sample code using API

The sample code shown below is the code used for creation and configuration of ProcessFlow, ProcessPhase and EngineOperation.

```
/** 1) Create ProcessFlow from ORIENT Method. */
ProcessFlow processFlow = orientMethod.createProcessFlow(new URI(
        "http://www.nec.com/ivcp/example/query/processflow"));

/** 2) Create ProcessPhase from ProcessFlow. */
ProcessPhase processPhase = processFlow.createPhase(new URI(
        "http://www.nec.com/ivcp/example/query/processphase1"));

/** 3) Create EngineOperation from ProcessPhase. */
EngineOperation engineOperation = processPhase
        .createEngineOperation(
                new URI("http://orient_example.com/person_search"),
                new URI(
                        "http://orient_example.com/person_search/api"),
                10000, 1);

/** 4) Set EngineOperation in ProcessPhase. */
processPhase.addEngineOperation(engineOperation);

/** 5) Set ProcessPhase in ProcessFlow. */
processFlow.addPhase(processPhase);

/** 6) Set ProcessFlow in ORIENT Method. */
orientMethod.setProcessFlow(processFlow);
```

5.3 ARF (ARMOR Restriction Framework)

5.3.1 Creation procedure of Condition

(1) Processing method

Condition is created by following method and is configured in ORIENT Method.

1) Condition is created from ORIENT Method.

2) Filter is created from Condition (Refer to 7.8 Filter usage methods for Filter).

3) Filter is configured in Condition.

4) Condition is configured in ORIENTMethod.

(2) Sample code using API

The sample code shown below is the code used for creation and configuration of Condition.

```
 /** 1) Create Condition from ORIENT Method. */
 Condition condition = method.createCondition();

/** 2) Create Filter (EqualsResourceFilter) from Condition. */
EqualsResourceFilter equalsResourceFilter = condition
        .createEqualsResourceFilter(new URI(
                "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

/* Set Entity in EqualsResourceFilter. */
equalsResourceFilter.setValue(new URI("mailto:nippon-denki@dummy.nec.com"));

/** 3) Set Filter in Condition. */
condition.setFilter(equalsResourceFilter);

/** 4) Set Condition in ORIENT Method. */
method.setCondition(condition);
```

5.3.2 Creation procedure of Scope

(1) Processing method

Scope is created by the following method and is configured in ORIENT Method.

1) Scope is created from ORIENT Method.

2) The information of properties, which you want to obtain as a search result, is configured in Scope.

3) Filter is configured in Scope (Refer to 7.8 Filter usage methods for Filter).

4) Scope is configured in ORIENTMethod.

(2) Sample code using API

The sample code shown below is the code used for creation and configuration of Scope.

```
/* OrderByFilter Creation*/
OrderByFilter orderByFilter = condition
        .createOrderByFilter(
                new URI(
                        "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday"),
                1, 2, ORDER.ASCENDING);

/** 1) Create Scope from ORIENT Method. */
Scope scope = method.createScope();

/** 2) Set information of properties, which you want to obtain as a search result, in Scope. */
scope.setEntityType(PersonDSMResourceTypes.PERSON);
String baseURI = "http://www.nec.com/ivcp/orient/da/general/ri/";
URI[] SCOPE_RESOURCES = {
        new URI(baseURI + "ds.person"),
        new URI(baseURI + "ds.person/vCard.fn"),
        new URI(baseURI + "ds.person/vCard.org/ds.Organization"),
        new URI(baseURI + "ds.person/vCard.org/ds.Organization/vCard.orgunit") };
scope.addScopeReferenceModelNodeURIs(SCOPE_RESOURCES);

/** 3) Set Filter in Scope. */
scope.setFilter(orderByFilter);

/** 4) Set Scope in ORIENTMethod.*/
method.addScope(scope);
```

5.3.3 Creation procedure of Filter

(1) Creation procedure of DataFilter

Processing method, in which EqualsFilter, RangesFilter, ContainsFilter, SimilarToFilter and MatchesFilter are used, is explained below.

(1-1) Equals

(1-a) Processing method

EqualsFilter is created and configured by the following processing method.

1) EqualsFilter is created from Condition. The value, which is to be compared, is configured at the time of creation.

2) The created EqualsFilter is configured in Condition.

(1-b) Sample code using API

The sample code shown below is the code used for creation and configuration of EqualsFilter.

This sample code is to search resources, which match with strings of “PC Incharge”, in literal (http://www.nec.com/ivcp/orient/da/general/ri/ds.person/vCard.note) of evaluation target.

```
/** 1) Create EqualsFilter from Condition*/
EqualsFilter equalsFilter = condition.createEqualsFilter(new URI(
      "http://www.nec.com/ivcp/orient/da/general/ri/ds.person/vCard.note"), new LiteralValueImpl(
      XSDDataType.STRING, "PC Incharge"));

/** 2) Set EqualsFilter in Condition*/
condition.setFilter(equalsFilter);
```

(1-2) Ranges

(1-a) Processing method

RangesFilter is created and configured by the following processing method.

1) RangesFilter is created from Condition.

2) Begin value of comparison range is configured in RangesFilter.

3) Whether begin value is to be included in target of comparison or not is configured.

4) End value of comparison range is configured in RangesFilter.

5) Whether end value is to be included in target of comparison or not is configured.

6) The created RangesFilter is configured in Condition.

(1-b) Sample code using API

The sample code shown below is the code used for creation and configuration of RangesFilter.

This sample code is to search resources corresponding to range of “1980-1-1” ~ “1985-1-1” in literal (http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday), which is the target of evaluation.

```
/** 1) Create RangesFilter from Condition*/
RangesFilter rangesFilter = condition.createRangesFilter(new URI(
      "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday"));

/** 2) Set begin value of comparison range in RangesFilter.*/
DateTime beginValue = new DateTime(1980, 1, 1, 0, 0, 0, 0,
      DateTimeZone.forID("America/New_York"));
rangesFilter.setBegin(new LiteralValueImpl(XSDDataType.DATE,
      beginValue));

/** 3) Set whether begin value is to be included in target of comparison or not.*/
rangesFilter.setIncludeBegin(true);

/** 4) Set end value of comparison range in RangesFilter.*/
DateTime endValue = new DateTime(1985, 1, 1, 0, 0, 0, 0,
      DateTimeZone.forID("America/New_York"));
rangesFilter
      .setEnd(new LiteralValueImpl(XSDDataType.DATE, endValue));

/ ** 5) Set whether end value is to be included in target of comparison or not.*/
rangesFilter.setIncludeEnd(true);

/** 6) Set RangesFilter in Condition.*/
condition.setFilter(rangesFilter);
```

(1-3) Contains

(1-a) Processing method

ContainsFilter is created and configured by the following processing method.

1) ContainsFilter is created from Condition. The value, which is to be compared, is configured at the time of creation.

2) The created ContainsFilter is configured in Condition.

(1-b) Sample code using API

The sample code shown below is the code used for creation and configuration of ContainsFilter.

This sample code is to search resources, which include string of “Intellectual property”, in literal (http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.org/ds.Organization/vCard.orgunit), which is the target of evaluation.

```
/** 1) Create ContainsFilter from Condition. Value to be compared is configured during creation.*/
ContainsFilter equalsFilter = condition
        .createContainsFilter(
                new URI(

                "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.org/ds.Organization/vCa
                rd.orgunit"),
                new LiteralValueImpl(XSDDataType.STRING, "Intellectual property"));

/** 2) Set the created ContainsFilter in Condition.*/
condition.setFilter(equalsFilter);
```

(1-4) SimilarTo

(1-a) Processing method

SimilarToFilter is created and configured by the following processing method.

1) SimilarToFilter is created from Condition. The value, which is to be compared, is configured at the time of creation.

2) The created SimilarToFilter is configured in Condition.

(1-b) Sample code using API

The sample code shown below is the code used for creation and configuration of SimilarToFilter.

This sample code is a code, which is used to search value closer to “Intellectual property and R&D Unit” in literal (http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.org/ds.Organization/vCard.orgunit), which is target of evaluation.

```
 /** 1) Create SimlarToFilter from Condition*/
 SimilarToFilter similarToFilter = condition.createSimilarToFilter(
        newURI(“http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.org/ds.Organization/vCard.orgunit”),
        new LiteralValueImpl(XSDDataType.STRING, "Intellectual property and R&D unit"), 0.2);

/** 2) Set SimilarToFilter in Condition*/
condition.setFilter(similarToFilter);
```

(1-5) Matches

(1-a) Processing method

MatchesFilter is created and configured by the following processing method.

Expression to be configured in MatchesFilter is defined by the person in DSM.

1) MatchesFilter is created from Condition. The value, which is to be compared, is configured at the time of creation.

2) Expression is created from MatchesFilter.

3) Comparison content is configured in Expression.

4) Expression is configured in MatchesFilter.

5) MatchesFilter is configured in Condition.

(1-b) Sample code using API

The sample code shown below is the code used for creation and configuration of MatchesFilter.

This sample code is the code in which search is carried out by defining evaluation formula in Expression whether resources that match with string of “Researcher” exists or not in Literal (http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.note), which is target of evaluation.

```
/** 1)Create MatchesFilter from Condition. The value to be compared is configured during creation.*/
MatchesFilter matchesFilter = condition
        .createMatchesFilter(new URI(
                "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.note"));

/** 2)Create Expression.*/
Expression expression = matchesFilter.createExpression(
        PersonDSMResourceTypes.ORGANIZATION, new URI(
                "http://example.com/expression/"));


/** 3)Set comparison content in Expression.*/
expression.setDataProperty(PersonDSMProperties.VCARD_NOTE, "Researcher");

/** 4)Set Expression in MatchesFilter.*/
matchesFilter.setExpression(expression);

/** 5)Set MatchesFilter in Condition..*/
condition.setFilter(matchesFilter);
```

(2) Creation procedure of ObjectFilter

Processing method, in which EqualsResourceFilter, ContainsResourceFilter, ExistFilter, SimilarToResourceFilter and RangesResourceFilter are used, is explained below.

(2-1) EqualsResource

(2-a) Processing method

EqualsResource is created and configured by the following processing method.

1) EqualsResourceFilter is created from Condition.

2) Resource URI, which is to be compared, is configured in EqualsResourceFilter.

3) The created EqualsResourceFilter is configured in Condition.

(2-b) Sample code using API

The sample code shown below is the code used for creation and configuration of EqualsResource.

This sample code is to search resources matching with “mailto:nippon-denki@dummy.nec.com” in resource (“http://www.nec.com/ivcp/orient/da/general/ri/ds.person”), which is the target of evaluation.

```
/** 1) Create EqualsResource from Condition.*/
EqualsResourceFilter equalsResourceFilter = condition
        .createEqualsResourceFilter(new URI(
                "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

/** 2) Set Resource URI of target for comparison in EqualsResource */
equalsResourceFilter.setValue(new URI("mailto:nippon-denki@dummy.nec.com"));

/** 3) Set the created EqualsResource in Condition. */
condition.setFilter(equalsResourceFilter);
```

(2-2) ContainsResource

(2-a) Processing method

ContainsResource is created and configured by following processing method.

1) ContainsResourceFilter is created from Condition.

2) Resource URI, which is to be compared, is configured in ContainsResourceFilter.

3) The created ContainsResourceFilter is configured in Condition.

(2-b) Sample code using API

The sample code shown below is the code for creation and configuration of ContainsResource.

This sample code is to search resources, containing property that resource “mailto:nippon-denki@dummy.nec.com” possesses, its Literal value and Resource, in Resource (“http://www.nec.com/ivcp/orient/da/general/ri/ds.person”), which is the target of evaluation.

```
/** 1) Create ContainsResourceFilter from Condition.*/
ContainsResourceFilter containsResourceFilter = condition
            .createContainsResourceFilter(new URI(
                        "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

/** 2) Set Resource URI of target for comparison in ContainsResourceFilter. */
containsResourceFilter.setValue(new URI(
          " mailto:nippon-denki@dummy.nec.com"));

/** 3) Set the created ContainsResourceFilter in Condition. */
condition.setFilter(containsResourceFilter);
```

(2-3) Exists

(2-a) Processing method

Exists is created and configured by the following processing method.

1) ExistsFilter is created from Condition.

2) The created ExistsFilter is configured in Condition.

(2-b) Sample code using API

This sample code is to determine whether resource (“http://www.nec.com/ivcp/orient/da/general/ri/ds.person”), which is the target of evaluation, exists in Entity instance or not.

```
/** 1) Create Exists from Condition.*/
      ExistsFilter existsFilter = condition.createExistsFilter(new URI(
                  "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

/** 2) Set the created Exists in Condition. */
condition.setFilter(existsFilter);
```

(2-4) SimilarToResource

(2-a) Processing method

1) SimilarToResourceFilter is created from Condition.

2) URIReference, which performs similarity comparison, is configured in SimilarToResourceFilter.

3) The created SimilarToFilter is created in Condition.

(2-b) Sample code using API

This sample code is to search whether resource specified in “mailto:nippon-denki@dummy.nec.com” is similar to resource (“http://www.nec.com/ivcp/orient/da/general/ri/ds.person”), which is the target of evaluation.

```
/** 1) Create SimilarToFilter from Condition.*/
      SimilarToResourceFilter similarToResourceFilter = condition
                  .createSimilarToResourceFilter(new URI(
                              "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

/* 2) Set ResourceURI of resource, for which similarity comparison is to be carried out, as a value.*/
      similarToResourceFilter.setValue(new URI(
                  " mailto:nippon-denki@dummy.nec.com"));

/** 3) Set the created SimilarToFilter in Condition. */
condition.setFilter(similarToResourceFilter);
```

(2-5) RangesResource

(2-a) Processing method

1) RangesResourceFilter is created from Condition.

2) The begin value of Resource range is configured in RangesResourceFilter.

3) Whether the begin value is to be included or not in Resource range is configured in RangesResourceFilter.

4) The end value of Resource range is configured in RangesResourceFilter.

5) Whether the end value is to be included or not in Resource range is configured in RangesResourceFilter.

6) The created RangesResourceFilter is configured in Condition.

(2-b) Sample code using API

This sample code is to search whether resources specified in beginResource and endResource are included or not in scope in resource (“http://www.nec.com/ivcp/orient/da/general/ri/ds.person”), which is the target of evaluation.

```
/** 1) Create RangesResourceFilter from Condition.*/
    SimilarToResourceFilter similarToResourceFilter = condition
    RangesResourceFilter rangesResourceFilter = condition
            .createRangesResourceFilter(new URI(
                    "http://www.nec.com/ivcp/orient/da/general/ri/ds.person"));

    /* 2) Set begin value of Resource range in RangesResourceFilter.*/
    rangesResourceFilter
            .setBegin(new URI(
                    "http://www.nec.com/ivcp/orient/da/general/ri/ds.person/score/100"));

    /* 3) Set whether begin value is to be included or not in Resource range in RangesResourceFilter.*/
    rangesResourceFilter.setIncludeBegin(true);

    /* 4) Set end value of Resource range in RangesResourceFilter.*/
    rangesResourceFilter
            .setEnd(new URI(
                    "http://www.nec.com/ivcp/orient/da/general/ri/ds.person/score/150"));

    /* 5) Set whether end value is to be included or not in Resource range in RangesResourceFilter.*/
    rangesResourceFilter.setIncludeEnd(true);

   /** 6) Set the created RangesResourceFilter in Condition. */
   condition.setFilter(rangesResourceFilter);
```

(3) Creation procedure of Logic

Processing method, in which ANDFilter, NotFilter and OrFilter are used, is explained below.

(3-1) And

(3-a) Processing method

And is created and configured by the following processing method.

1) AndFilter is created from Condition.

2) Filter to be specified as AND condition is configured in AndFilter.

3) The created AndFilter is configured in Condition.

(3-b) Sample code using API

The sample code shown below is the code for creation and configuration of And.

This sample code is to search such that conditions of both EqualFilter and RangesFilter are fulfilled.

```
/* Create EqualsFilter*/
EqualsFilter equalsFilter = condition.createEqualsFilter(new URI(
        “http://www.nec.com/ivcp/orient/da/general/ri/ds.person/vCard.note”), new LiteralValueImpl(
        XSDDataType.STRING, "PC Incharge"));

/* Create RangesFilter*/
RangesFilter rangesFilter = condition.createRangesFilter(new URI(
        “http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday”));
DateTime beginValue = new DateTime(1980, 1, 1, 0, 0, 0, 0,
        DateTimeZone.forID("America/New_York"));
rangesFilter.setBegin(new LiteralValueImpl(XSDDataType.DATE,
        beginValue));
rangesFilter.setIncludeBegin(true);
DateTime endValue = new DateTime(1985, 1, 1, 0, 0, 0, 0,
        DateTimeZone.forID("America/New_York"));
rangesFilter
        .setEnd(new LiteralValueImpl(XSDDataType.DATE, endValue));
rangesFilter.setIncludeEnd(true);

/** 1) Create AndFilter from Condition.*/
ANDFilter andFilter = condition.createANDFilter();

/** 2) Set Filter to be specified as AND condition in AndFilter. */
andFilter.addFilter(equalsFilter);
andFilter.addFilter(rangesFilter);

/** 3) Set the created AndFilter in Condition.*/
condition.setFilter(andFilter);
```

(3-2) Not

(3-a) Processing method

Not is created and configured by the following processing method.

1) NotFilter is created from Condition.

2) Filter to be specified as NOT condition is configured in NotFilter.

3) The created NotFilter is configured in Condition.

(3-b) Sample code using API

The sample code shown below is the code for creation and configuration of Not.

This sample code is to search resources, which do not fulfill conditions of EqualsFilter.

```
/* Create EqualsFilter*/
EqualsFilter equalsFilter = condition.createEqualsFilter(new URI(
        “http://www.nec.com/ivcp/orient/da/general/ri/ds.person/vCard.note”), new LiteralValueImpl(
        XSDDataType.STRING, "Researcher"));

/**  1)Create NotFilter from Condition. */
NOTFilter notFilter = condition.createNOTFilter();

/**  2) Set Filter to be specified as NOT condition in NotFilter. */
notFilter.setFilter(equalsFilter);

/**  3) Set the created NotFilter in Condition.*/
condition.setFilter(notFilter);
```

(3-3) OR

(3-a) Processing method

OR is created and configured by the following processing method.

1) OrFilter is created from Condition.

2) Filter to be specified as OR condition is configured in OrFilter.

3) The created OrFilter is configured in Condition.

(3-b) Sample code using API

The sample code shown below is the code for creation and configuration of Or.

This sample code is to search resources, which fulfill either of the two EqualsFilter conditions.

```
/* Create EqualsFilter*/
EqualsFilter equalsFilterForServ = condition.createEqualsFilter(
        new URI(ORGUNIT_NAMESPACE), new LiteralValueImpl(
                XSDDataType.STRING, "Central labs"));

EqualsFilter equalsFilterForInfo = condition.createEqualsFilter(
        new URI(ORGUNIT_NAMESPACE), new LiteralValueImpl(
                XSDDataType.STRING, "ABC Labs"));

/** 1) Create OrFilter from Condition. */
ORFilter orFilter = condition.createORFilter();

/** 2) Set Filter to be specified as OR condition in OrFilter. */
orFilter.addFilter(equalsFilterForServ);
orFilter.addFilter(equalsFilterForInfo);

/** 3) Set the created OrFilter in Condition.*/
condition.setFilter(orFilter);
```

(4) Creation procedure of OrderBy

(4-1) Processing method

OrderBy is created and configured by the following processing method.

1) OrderByFilter is created from Condition. Sorting conditions are configured at the time of creation.

2) The created OrderBy Filter is configured in Scope.

(4-2) Sample code using API

The sample code shown below is the code used for creation and configuration of OrderBy.

This sample code is to search top two names in ascending order in

Literal (“http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday”), which is the target of evaluation.

```
/** 1) Create OrderByFilter from Condition */
OrderByFilter orderByFilter = condition
        .createOrderByFilter(
                new URI(
                        "http://www.nec.com/ivcp/orient/ds/general/ri/ds.person/vCard.bday"),
                1, 2, ORDER.ASCENDING);

/* Create property condition to be obtained */
Scope scope = method.createScope();

/** 2) Set the created OrderBy Filter in Scope.*/
scope.setFilter(orderByFilter);
```

5.3.4 Creation procedure of Status

(1) Creation procedure of Operation Status

(1-1) Processing method

Operation Status is created and configured by the following processing method.

1) Status is created from ORIENT Method. The information of Status is configured at the time of creation.

2) Status is configured in ORIENTMethod.

(1-2) Sample code using API

The sample code shown below is the code used for creation and configuration of Operation Status

```
/** 1) Create Status from ORIENT Method. */
Status status = orientMethodRequest.createOperationStatus(200, "OK");

/** 2) Set Status in ORIENTMethod. */
orientMethodRequest.setOperationStatus(status);
```

(2) Creation procedure of EngineOperationStatus

(2-1) Processing method

EngineOperation Status is created and configured by the following processing method.

1) Status is created from ORIENT Method. The information of Status is configured at the time of creation.

2) Status is configured in ORIENTMethod. URI of EngineOperation, which is the target of Status, has to be specified in argument at the time of configuring Status. An error occurs if EngineOperation corresponding to URI specified in argument does not exist.

(2-2) Sample code using API

The sample code shown below is the code used for creation and configuration of EngineOperation Status

```
/**    1) Set Status of EngineOperation */
Status EngineOperationStatus = orientMethodRequest.createEngineOperationStatus(200, "OK");

 /** 2) Set Status in ORIENTMethod. */
orientMethodRequest.setEngineOperationStatus(new URI(
        "http://orient_example.com/person_search"),
        EngineOperationStatus);
```

6. Creation procedure of DSM

Refer to “OSDK-GUIDE-DSMResourceCreation-APIUsageGuide” for creation procedure of DSM.

7. Transmission procedure of ORIENT message

Transmission method of ORIENT Method, which is ORIENT message, is explained below.

Query Operation and Write Operation perform synchronous communication.

Notify Operation performs non-synchronous communication.

7.1 Transmission procedure of Query Operation

7.1.1 Analysis data exchange protocol in case of transmission to ARMOR

Analysis data exchange protocol, when transmission is done to ARMOR in Query Operation, is explained below.

(1) Processing method

Transmission is carried out by following processing method.

1) ORIENTMethodSenderFactory is created. At that time, ORIENTMethodSenderFactoryImpl is specified.

2) OrientMethodSender is created.

3) Request is sent by using sendRequestSync method. ORIENT Method, which is responded as return value, is obtained.

4) Operation Status is obtained from ORIENT Method and it is checked if code is 200 (successful completion).

5) Information of Entities is obtained from ORIENT Method in case of successful completion in 4).

6) Information of Entity is obtained from Entites.

7) Error handling is carried out if it is not a successful completion in 4).

(2) Sample code using API

Given below is an example of code when transmission is done to ARMOR.

```
/** 1) Create ORIENTMethodSenderFactory. */
ORIENTMethodSenderFactory senderFactory = new ORIENTMethodSenderFactoryImpl ();

/** 2) Create OrientMethodSender. */
OrientMethodSender sender = senderFactory.create();

/** 3) Send request by using sendRequestSync method. */
/**      Obtain ORIENT Method, which is received as response for return value. */
ORIENTMethod response = sender.sendRequestSync(new URI(
            "http://ARMOR_ENDPOINT"), method,
            ORIENTConstants.RDF_XML_MESSAGE_TYPE);

/** 4) Obtain Operation Status from ORIENT Method and */
 /** check if code is 200 (successful completion). */
 if (response.getOperationStatus().getCode() == 200) {
      /**5) Obtain information of Entities from ORIENT Method in case of successful completion.*/
      Set<Entities> entitiesSet = response.getAllEntities();
      for (Entities entities : entitiesSet) {
            /** 6) Obtain information of Entity from Entites. */
            List<Entity> entityList = entities.getAllEntities();
            Iterator<Entity> entityIterator = entityList.iterator();
            while (entityIterator.hasNext()) {
                  Entity entity = entityIterator.next();
            }
      }
 } else {
      /** 7) Perform error handling if it is not a successful completion. */
 }
```

7.2 Analysis data exchange protocol of Write Operation

7.2.1 Analysis data exchange protocol in case of transmission to ARMOR

Analysis data exchange protocol, when transmission is done to ARMOR in Write Operation, is explained below.

(1) Processing method

Send and receive are performed by following processing method.

1) ORIENTMethodSenderFactory is created. ORIENTMethodSenderFactoryImpl is specified at that time.

2) OrientMethodSender is created.

3) Request is sent by using sendRequestSync method. ORIENT Method, which is responded as return value, is obtained.

4) Operation Status is obtained from ORIENT Method and it is checked if code is 200 (successful completion).

5) There is no response in case of successful completion in 4).

6) Error handling is carried out if it is not a successful completion in 4).

(2) Sample code using API

Given below is an example of code used when transmission to ARMOR is carried out.

```
/** 1) Create ORIENTMethodSenderFactory. */
ORIENTMethodSenderFactory senderFactory = new ORIENTMethodSenderFactoryImpl ();

/** 2) Create OrientMethodSender. */
OrientMethodSender sender = senderFactory.create();

/** 3) Send request by using sendRequestSync method. */
/**      Obtain ORIENT Method, which is received as response for return value. */
ORIENTMethod response = sender.sendRequestSync(new URI(
            "http://ARMOR_ENDPOINT"), method,
            ORIENTConstants.RDF_XML_MESSAGE_TYPE);

/** 4) Obtain Operation Status from ORIENT Method and */
/** check if code is 200 (successful completion). */
if (response.getOperationStatus().getCode() == 200) {
      /** 5) No response in case of successful completion. */s
} else {
      /** 6) Perform error handing if it is not a successful completion. */
}
```

7.3 Transmission procedure of Notify Operation

(1) Processing method

Transmission procedure, when transmission is performed by Notify Operation, is explained below.

1) ORIENTMethodSenderFactory is created. Either of ORIENTMethodSenderFactoryImpl or ArmorORIENTMethodSenderFactoryImpl can be specified as Notify does not return response.

2) OrientMethodSender is created.

3) Request is sent by using sendRequestAsync method.

(2) Sample code using API

Given below is an example of code used when transmission is performed by Notify Operation.

```
/** 1) Create ORIENTMethodSenderFactory. */
ORIENTMethodSenderFactory senderFactory = new ORIENTMethodSenderFactoryImpl ();

/** 2) Create OrientMethodSender. */
OrientMethodSender sender = senderFactory.create();

/** 3) Send request by using sendRequestAsync method. */
sender.sendRequestAsync(new URI("http://AP_ENDPOINT"
), method,
        ORIENTConstants.RDF_XML_MESSAGE_TYPE);
```

8. Serialization and deserialization of ORIENTMethod

Serialization and deserialization method of ORIENTMethod is explained below. AP developer is not required to use this API directly. However, it can be used in case of generating serialization format of message in debugging, etc.

8.1 Method to use serialized API

ORIENTMethod can be serialized into XML by using serialize method, which is the interface of ORIENTMethod. However, encoding of string will be UTF-8.

```
sLog.info(orientMethod.serialize());
```

8.2 Method to use deserialized API

ORIENTMethodLoader is an interface which reads ORIENTMethod from InputStream, etc. and creates ORIENTMethod. User can perform deserialization by reading ORIENTMethod after creating instance of ORIENTMethodLoaderImpl class, which is implementation of this interface.

```
ORIENTMethodLoader loader = new ORIENTMethodLoaderImpl();
```