Proxy of DisplayText Format Type

Status
This document is a request for a specification change for review.

Summary
The ability for an OSID Consumer to request a specific DisplayText format Type from OSID Providers through an OsidProxyManager requires an out of band agreement around a Proxy Record and ProxyConditionRecord Type. This is a request to proxy a format Type using core methods and have visibility into the format Type through OsidSessions.

Table of Contents

1. Current Specification........................................................................................................2
2. Problem ........................................................................................................................3
3. Proposed Change ...........................................................................................................4
4. Impacts ........................................................................................................................5
   4.1. Specification ...........................................................................................................5
   4.2. OSID Consumers .................................................................................................5
   4.3. OSID Providers ......................................................................................................5
5. Interoperability Considerations ..................................................................................5
6. Proposed Interfaces ....................................................................................................6
   6.1. osid.OsidSession .................................................................................................6
   6.2. osid.proxy.ProxyCondition ...............................................................................8
   6.3. osid.proxy.Proxy .................................................................................................9
7. Copyright Statement ..................................................................................................10
1. **Current Specification**

DisplayText is an OsidPrimitive to bundle a display string with a format Type. The format Type is an indicator to an OSID Consumer that is able to perform format conversions.

Alignment between of the DisplayText format between the application and the OSID Provider is an integration issue. For OSID Providers that may support more than one DisplayText format, the alignment may be performed by:

1. **different service endpoints into the same OSID Provider implementation**

   ```
   getManager(osid, provider1)
   ```

   ```
   getManager(osid, provider2)
   ```

   ```
   Osid Runtime
   ```

   ```
   Format OSID Adapter
   ```

   ```
   Base OSID Provider
   ```

   ```
   DisplayText format=NROFF
   ```

   ```
   DisplayText format=HTML5
   ```

   ```
   OSID Provider 1
   ```

   ```
   OSID Provider 2
   ```

   ```
   OSID Provider 1 & 2
   ```
2. configuration of the OSID Provider

getManager(osid, provider)

Osid Runtime

“context”

OsidManager

formatType

Configuration OSID

DisplayText format=HTML5

DisplayText format=NROFF

OSID Provider

Proxy format=HTML5

OsidProxyManager

DisplayText format=HTML5

DisplayText format=NROFF

OSID Provider

3. in the case of a proxy scenario, a ProxyRecord defined to pass a format Type

2. Problem

1. While there is a core method to proxy locale and authentication information, there is no core method to proxy format to implement scenario 3 above. Currently, the format Type needs to be defined in a ProxyRecord for an OSID Provider that is able to support multiple formats. The definition of a format type in the OsidPrimitive DisplayText indicates that the issue of text formats is of general concern. Therefore, it is reasonable to include its explicit definition in the Proxy OSID where the Proxy is passed into OsidProxyManagers.
2. OsidSessions expose the Locale, effective date, and the effective agent. While this information may have been proxied by an OSID Consumer, the OSID Provider may or may not have honored it. The OsidSession provides the means to communicate what information actually in use by an OSID Provider. Similarly, the same should be true for the DisplayText format. Like the calendar and time Types, the OsidSession communicates the mode of the OSID Provider. It is conceivable that DisplayTexts of other format Types are seen as is the case with DateTime. The OSID specifications make no statement about such discrepancies.

3. **Proposed Change**

Add methods to osid.proxy.ProxyCondition and osid.proxy.Proxy to specify and access a format Type.

```java
osid.proxy.ProxyCondition {
    ...

    /**
     * Specifies the preferred DisplayText format Type.
     *
     * @param formatType a DisplayText format Type
     * @throws osid.NullArgumentException formatType is null
     */
    void setFormatType(osid.type.Type formatType);
}

osid.proxy.Proxy {
   ...

    /**
     * Tests if a DisplayText format Type is available.
     *
     * @return true if a format Type is available, false otherwise
     */
    boolean hasFormatType();

    /**
     * Gets the preferred DisplayText format Type.
     *
     * @return the preferred DisplayText format Type
     * @throws osid.IllegalStateException hasFormatType() is false
     */
    osid.type.Type getFormatType();
}
```
Add a method to osid.OsidSession for a DisplayText format Type.

```java
osid.OsidSession {
    ...
    /**
     * Gets the DisplayText format Type preference in effect for
     * this session.
     *
     * @return the effective DisplayText format Type
     */
    osid.type.Type getFormatType();
}
```

## 4. Impacts

### 4.1. Specification

Changes are restricted to the addition of four new methods among osid.Proxy, osid.ProxyCondition, and osid.OsidSession.

### 4.2. OSID Consumers

No impacts on OSID Consumers since no interface methods are changed or removed. OSID Consumers may optionally make use of the format specifier in ProxyCondition or the format hint provided from OsidSessions.

### 4.3. OSID Providers

OSID Proxy Providers would be expected to minimally implement:

- ProxyCondition.setFormatType()
- Proxy.hasFormatType()
- OsidSession.getFormatType()

An OSID Provider supporting an OsidProxyManager would not required to use the DisplayText format Type in which case format level disagreements would be resolved at the provider level.

OsidSessions would be required to return a default format Type. Ideally, an OSID Provider should not flip back and forth among multiple DisplayText formats although there is nothing preventing it from doing so. It is also assumed that there is some default DisplayText format the OsidSession is delivering.

## 5. Interoperability Considerations

This change clarifies the specification for those OSID Providers supporting OsidProxyManagers how to switch between DisplayText formats if the preferred format
Type is present in the Proxy. The change eliminates the interoperability hurdle of a record Type agreement in the Proxy.

The presence or absence of the format Type in the Proxy is indicated by the Proxy.hasMoreFormatType() method. This allows OSID Proxy Providers to easily support it or not support it.

The decision rests with an OSID Provider to honor the format Type, ignore the format Type, or supply a best effort match.

The only behavioral change requested is that OSID Providers seed OsidSessions with a default format Type.

6. Proposed Interfaces

6.1. osid.OsidSession

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Return</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLocale</td>
<td>Gets the locale indicating the localization preferences in effect for this session.</td>
<td>osid.locale.Locale</td>
<td>mandatory</td>
</tr>
<tr>
<td>isAuthenticated</td>
<td>Tests if an agent is authenticated to this session.</td>
<td>boolean</td>
<td>mandatory</td>
</tr>
<tr>
<td>getAuthenticatedAgentId</td>
<td>Gets the Id of the agent authenticated to this session. This is the agent for which credentials are used either acquired natively or via an OsidProxyManager.</td>
<td>osid.id.Id</td>
<td>mandatory</td>
</tr>
<tr>
<td>getAuthenticatedAgent</td>
<td>Gets the agent authenticated to this session. This is the agent for which credentials are used either acquired natively or via an OsidProxyManager.</td>
<td>osid.authentication.Agent</td>
<td>mandatory</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Return</td>
<td>Compliance</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>getEffectiveAgentId</td>
<td>Gets the Id of the effective agent in use by this session. If isAuthenticated() is true, then the effective agent may be the same as the agent returned by getAuthenticatedAgent(). If isAuthenticated() is false, then the effective agent may be a default agent used for authorization by an unknown or anonymous user.</td>
<td>osid.id.Id</td>
<td>mandatory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the effective agent</td>
<td></td>
</tr>
<tr>
<td>getEffectiveAgent</td>
<td>Gets the effective agent in use by this session. If isAuthenticated() is true, then the effective agent may be the same as the agent returned by getAuthenticatedAgent(). If isAuthenticated() is false, then the effective agent may be a default agent used for authorization by an unknown or anonymous user.</td>
<td>osid.authentication.Agent</td>
<td>the effective agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPERATION_FAILED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unable to complete request</td>
<td></td>
</tr>
<tr>
<td>getDate</td>
<td>Gets the service date which may be the current date or the effective date in which this session exists.</td>
<td>timestamp</td>
<td>the service date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mandatory</td>
<td>This method must be implemented.</td>
</tr>
<tr>
<td>getClockRate</td>
<td>Gets the rate of the service clock.</td>
<td>decimal</td>
<td>the clock rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mandatory</td>
<td>This method must be implemented.</td>
</tr>
<tr>
<td>getFormatType</td>
<td>Gets the DisplayText format Type preference in effect for this session.</td>
<td>osid.type.Type</td>
<td>the effective DisplayText format Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mandatory</td>
<td>This method must be implemented.</td>
</tr>
<tr>
<td>supportsTransactions</td>
<td>Tests for the availability of transactions.</td>
<td>boolean</td>
<td>true if transaction methods are available, false otherwise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mandatory</td>
<td>This method must be implemented.</td>
</tr>
<tr>
<td>startTransaction</td>
<td>Starts a new transaction for this session. Transactions are a means for an OSID to provide an all-or-nothing set of operations within a session and may be used to coordinate this service from an external transaction manager. A session supports one transaction at a time. Starting a second transaction before the previous has been committed or aborted results in an ILLEGAL_STATE error.</td>
<td>osid.transaction.Transaction</td>
<td>a new transaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ILLEGAL_STATE</td>
<td>a transaction is already open</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPERATION_FAILED</td>
<td>unable to complete request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNSUPPORTED</td>
<td>transactions not supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optional</td>
<td>This method must be implemented if supportsTransactions() is true.</td>
</tr>
<tr>
<td>Provider Notes</td>
<td>Ideally, a provider that supports transactions should guarantee atomicity, consistency, isolation and durability in a 2 phase commit process. This is not always possible in distributed systems and a transaction provider may simply allow for a means of processing bulk updates.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To maximize interoperability, providers should honor the one-transaction-at-a-time rule.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 6.2. osid.proxy.ProxyCondition

<table>
<thead>
<tr>
<th>Interface</th>
<th>osid.proxy.ProxyCondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements</td>
<td>osid.OsidCondition</td>
</tr>
<tr>
<td>Description</td>
<td>A ProxyCondition is used to transfer external information into a proxy.</td>
</tr>
</tbody>
</table>

#### Method
- **setEffectiveAgentId**
  - Description: Sets the effective agent Id to indicate acting on behalf of.
  - Parameters:
    - `osid.id.Id` agentId: an agent Id
  - Errors:
    - `NULL_ARGUMENT` agentId is null
  - Compliance: This method must be implemented.

#### Method
- **setEffectiveDate**
  - Description: Sets the effective date.
  - Parameters:
    - `timestamp` date: a date
    - `decimal` rate: the rate at which the clock should tick from the given effective date. 0 is a clock that is fixed
  - Errors:
    - `NULL_ARGUMENT` date is null
  - Compliance: This method must be implemented.

#### Method
- **setLanguageType**
  - Description: Sets the language type.
  - Parameters:
    - `osid.type.Type` languageType: the language type
  - Errors:
    - `NULL_ARGUMENT` languageType is null
  - Compliance: This method must be implemented.

#### Method
- **setScriptType**
  - Description: Sets the script type.
  - Parameters:
    - `osid.type.Type` scriptType: the script type
  - Errors:
    - `NULL_ARGUMENT` scriptType is null
  - Compliance: This method must be implemented.

#### Method
- **setCalendarType**
  - Description: Sets the calendar type.
  - Parameters:
    - `osid.type.Type` calendarType: the calendar type
  - Errors:
    - `NULL_ARGUMENT` calendarType is null
  - Compliance: This method must be implemented.

#### Method
- **setTimeType**
  - Description: Sets the time type.
  - Parameters:
    - `osid.type.Type` timeType: the time type
  - Errors:
    - `NULL_ARGUMENT` timeType is null
  - Compliance: This method must be implemented.

#### Method
- **setCurrencyType**
  - Description: Sets the currency type.
  - Parameters:
    - `osid.type.Type` currencyType: the currency type
  - Errors:
    - `NULL_ARGUMENT` currencyType is null
  - Compliance: This method must be implemented.

#### Method
- **setUnitSystemType**
  - Description: Sets the unit system type.
  - Parameters:
    - `osid.type.Type` unitSystemType: the unit system type
  - Errors:
    - `NULL_ARGUMENT` unitSystemType is null
  - Compliance: This method must be implemented.

#### Method
- **setFormatType**
  - Description: Sets the DisplayText format type.
  - Parameters:
    - `osid.type.Type` formatType: the format type
  - Errors:
    - `NULL_ARGUMENT` formatType is null
### 6.3. osid.proxy.Proxy

<table>
<thead>
<tr>
<th>Interface</th>
<th>osid.proxy.Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements</td>
<td>osid.OsidResult</td>
</tr>
<tr>
<td>Description</td>
<td>A Proxy is used to transfer external information from an application server into an OSID Provider.</td>
</tr>
</tbody>
</table>

#### hasAuthentication

<table>
<thead>
<tr>
<th>Method</th>
<th>hasAuthentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Tests if an authentication is available.</td>
</tr>
<tr>
<td>Return</td>
<td>boolean</td>
</tr>
<tr>
<td>Compliance</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

#### getAuthentication

<table>
<thead>
<tr>
<th>Method</th>
<th>getAuthentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gets the Authentication for this proxy.</td>
</tr>
<tr>
<td>Return</td>
<td>osid.authentication.process.Authentication</td>
</tr>
<tr>
<td>Errors</td>
<td>ILLEGAL_STATE</td>
</tr>
<tr>
<td>Compliance</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

#### hasEffectiveAgent

<table>
<thead>
<tr>
<th>Method</th>
<th>hasEffectiveAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Tests if an effective agent is available.</td>
</tr>
<tr>
<td>Return</td>
<td>boolean</td>
</tr>
<tr>
<td>Compliance</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

#### getEffectiveAgentId

<table>
<thead>
<tr>
<th>Method</th>
<th>getEffectiveAgentId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gets the effective Agent Id for this proxy.</td>
</tr>
<tr>
<td>Return</td>
<td>osid.id.Id</td>
</tr>
<tr>
<td>Errors</td>
<td>ILLEGAL_STATE</td>
</tr>
<tr>
<td>Compliance</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

#### hasEffectiveDate

<table>
<thead>
<tr>
<th>Method</th>
<th>hasEffectiveDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Tests if an effective date is available.</td>
</tr>
<tr>
<td>Return</td>
<td>boolean</td>
</tr>
<tr>
<td>Compliance</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

#### getEffectiveDate

<table>
<thead>
<tr>
<th>Method</th>
<th>getEffectiveDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gets the effective date.</td>
</tr>
<tr>
<td>Return</td>
<td>timestamp</td>
</tr>
<tr>
<td>Errors</td>
<td>ILLEGAL_STATE</td>
</tr>
</tbody>
</table>
## Compliance

### Method `getEffectiveClockRate`
- **Description**: Gets the rate of the clock.
- **Return**: `decimal` the rate
- **Errors**: `ILLEGAL_STATE` hasEffectiveDate() is false
- **Compliance**: mandatory This method must be implemented.

### Method `getLocale`
- **Description**: Gets the locale.
- **Return**: `osid.locale.Locale` a locale
- **Compliance**: mandatory This method must be implemented.

### Method `hasFormatType`
- **Description**: Tests if a DisplayText format Type is available.
- **Return**: `boolean` true if a format type is available, false otherwise
- **Compliance**: mandatory This method must be implemented.

### Method `getFormatType`
- **Description**: Gets the DisplayText format Type.
- **Return**: `osid.type.Type` the format Type
- **Errors**: `ILLEGAL_STATE` hasFormatType() is false
- **Compliance**: mandatory This method must be implemented.

### Method `getProxyRecord`
- **Description**: Gets the proxy record corresponding to the given Proxy record Type. This method is used to retrieve an object implementing the requested record. The proxyRecordType may be the Type returned in getRecordTypes() or any of its parents in a Type hierarchy where hasRecordType(proxyRecordType) is true.
- **Parameters**: `osid.type.Type` proxyRecordType the type of proxy record to retrieve
- **Return**: `osid.proxy.records.ProxyRecord` the proxy record
- **Errors**: `NULL_ARGUMENT` proxyRecordType is null
- `OPERATION_FAILED` unable to complete request
- `UN_SUPPORTED` hasRecordType(proxyRecordType) is false
- **Compliance**: mandatory This method must be implemented.

7. **Copyright Statement**

Copyright (C) Ingenescus (2013). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the authors, Ingenescus, or other organizations, except as required to translate it into languages other than English.

This document and the information contained herein is provided on an "AS IS" basis and Ingenescus and the authors DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.