Diameter Multiplexer (MUX) Dictionary

Table of Contents

The Dictionary is part of the MUX package. Its purpose is to provide unified access to information regarding AVP structure, content and definition. It is configured with an XML file: *dictionary.xml*.

Dictionary logic is contained in the `org.mobicents.diameter.dictionary.AvpDictionary` class. It exposes the following methods:

`public AvpRepresentation getAvp(int code)`
: Return an `AvpRepresentation` object representing the AVP with the given code (assuming vendor ID as 0 (zero)). If there is no AVP defined, it returns `null`.

`public AvpRepresentation getAvp(int code, long vendorId)`
: Returns an `AvpRepresentation` object representing the AVP with the given code and vendor ID. If there is no AVP defined, it returns `null`.

`public AvpRepresentation getAvp(String avpName)`
: Returns an `AvpRepresentation` object representing the AVP with the given name. If there is no AVP defined, it returns `null`.

Dictionary uses a POJO class to provide access to stored information: `org.mobicents.diameter.dictionary.AvpRepresentation`. It exposes the following methods:

`public int getCode()`
: Returns the code assigned to the represented AVP.

`public long getVendorId()`
: Returns the vendor ID assigned to the represented AVP.

`public String getName()`
: Returns name assigned to the represented AVP. If no name is defined, it returns `null`.

`public boolean isGrouped()`
: Returns `true` if the AVP is of grouped type.

`public String getType()`
: Returns a `String` with the name of the represented AVP type. Return value is equal to one of defined types. For example, `OctetString` or `Unsiged32`.

`public boolean isMayEncrypt()`
: Returns `true` if the AVP can be encrypted.

`public boolean isProtected()`
: Returns `true` if the AVP *must* be encrypted. This occurs if `public String getRuleProtected()` returns `must`.

`public boolean isMandatory()`
: Returns `true` if the AVP must be supported by an agent to properly consume the message. It only returns `true` if `public String getRuleMandatory()` returns `must`.

`public String getRuleMandatory()`
: Returns the mandatory rule value. It can return one of the following values: `may`, `must` or `mustnot`.

`public String getRuleProtected()`

Returns the protected rule value. It can have one of the following values: `may`, `must` or `mustnot`.

`public String getRuleVendorBit()`

Returns the vendor rule value. It can have one of the following values: `must` or `mustnot`.

The Diameter MUX Dictionary can be used as follows:

```
public static void addAvp(Message msg, int avpCode, long vendorId, AvpSet set, Object
avp) {
    AvpRepresentation avpRep = AvpDictionary.INSTANCE.getAvp(avpCode, vendorId);

    if(avpRep != null) {
        DiameterAvpType avpType = DiameterAvpType.fromString(avpRep.getType());

        boolean isMandatoryAvp = avpRep.isMandatory();
        boolean isProtectedAvp = avpRep.isProtected();

        if(avp instanceof byte[]) {
            setAvpAsRaw(msg, avpCode, vendorId, set, isMandatoryAvp, isProtectedAvp,
(byte[]) avp);
        }
        else
        {
            switch (avpType.getType()) {
            case DiameterAvpType._ADDRESS:
            case DiameterAvpType._DIAMETER_IDENTITY:
            case DiameterAvpType._DIAMETER_URI:
            case DiameterAvpType._IP_FILTER_RULE:
            case DiameterAvpType._OCTET_STRING:
            case DiameterAvpType._QOS_FILTER_RULE:
                setAvpAsOctetString(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    avp.toString());
                break;

            case DiameterAvpType._ENUMERATED:
            case DiameterAvpType._INTEGER_32:
                setAvpAsInteger32(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Integer) avp);
                break;

            case DiameterAvpType._FLOAT_32:
                setAvpAsFloat32(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Float) avp);
                break;

            case DiameterAvpType._FLOAT_64:
                setAvpAsFloat64(msg, avpCode, vendorId, set, isMandatoryAvp,
```

```
isProtectedAvp,
                    (Float) avp);
                break;

            case DiameterAvpType._GROUPED:
                setAvpAsGrouped(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (DiameterAvp[]) avp);
                break;

            case DiameterAvpType._INTEGER_64:
                setAvpAsInteger64(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Long) avp);
                break;

            case DiameterAvpType._TIME:
                setAvpAsTime(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Date) avp);
                break;

            case DiameterAvpType._UNSIGNED_32:
                setAvpAsUnsigned32(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Long) avp);
                break;

            case DiameterAvpType._UNSIGNED_64:
                setAvpAsUnsigned64(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (Long) avp);
                break;

            case DiameterAvpType._UTF8_STRING:
                setAvpAsUTF8String(msg, avpCode, vendorId, set, isMandatoryAvp,
isProtectedAvp,
                    (String) avp);
                break;
            }
        }
    }
}
```