Java Compute Node API From ESQL Perspective

Dr.Srinivasa Babu Purushothaman

psrinivasababu@gmail.com

Agenda

- Overview
- Configuration and Deployment
- Processing messages
- Working with Databases
- Debugging JCN Code

Disclaimer

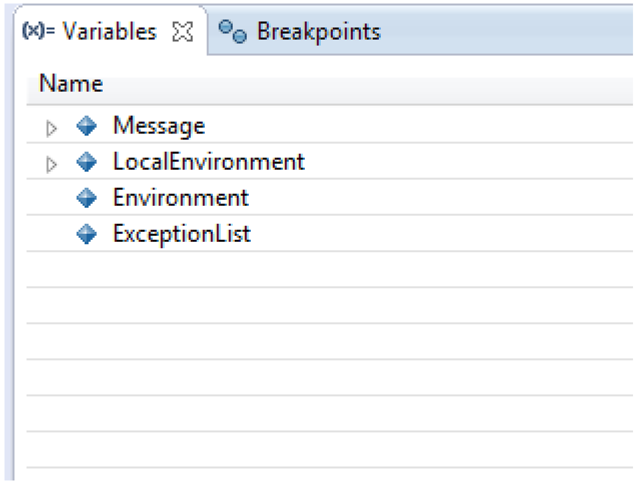
- The java statements given here are not 100% direct statements of ESQL. They are either equivalent or serve similar purpose.

Why Java Compute Node?

- Popular language
- Ready availability of Java resources
- Rich third party API support
- Full IDE support of Eclipse
- Support for both JDBC and ODBC
- Xpath support
- Global Cache support
- Flexible to create User Defined Node
- Integration API support for custom solutions

Message Assembly

- (Input/Output) Message
- LocalEnvironment
- (Global)Environment
- ExceptionList



(x)= Variables ⌧ | Breakpoints

Name
- Message
- LocalEnvironment
- Environment
- ExceptionList

- Input/Output Message
- LocalEnvironment
- (Global)Environment
- ExceptionList

$\}$ **MbMessage Objects**

ESQL to Java Correlation Name Mapping

- **Mapping**

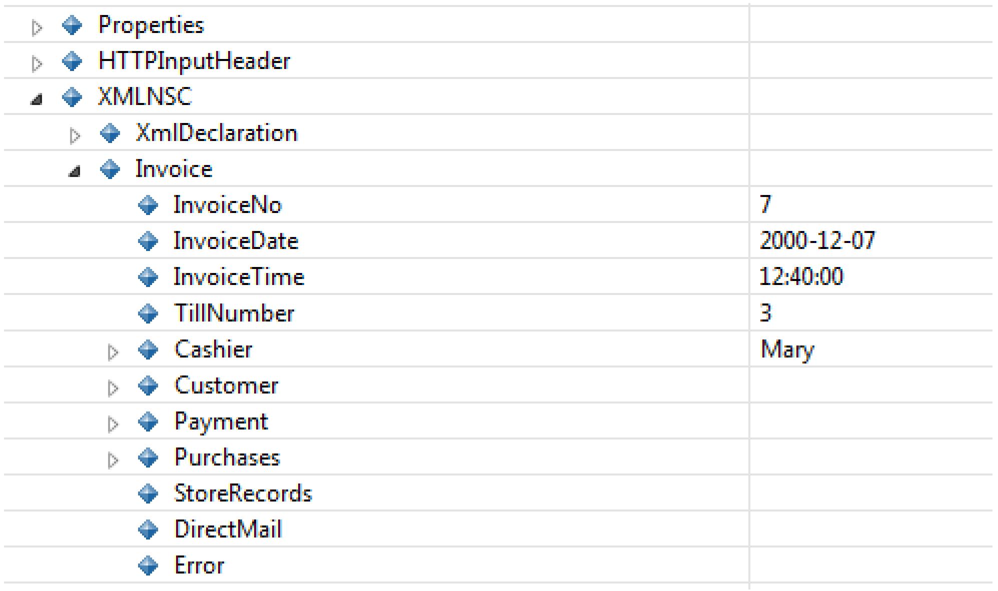
ESQL correlation name	Java accessor from MbMessageAssembly
InputRoot	getMessage().getRootElement()
InputBody	getMessage().getRootElement().getLastChild()
InputLocalEnvironment	getLocalEnvironment().getRootElement()
Environment	getGlobalEnvironment().getRootElement()
InputExceptionList	getExceptionList().getRootElement()

- **Sample Java code**

```
MbMessage inMessage = inAssembly.getMessage();
MbElement inputRoot = inMessage.getRootElement();
MbElement inputbody = inMessage.getRootElement().getLastChild();
MbElement inputLocalEnvironment =
inAssembly.getLocalEnvironment().getRootElement();
MbElement environment = inAssembly.getGlobalEnvironment().getRootElement();
MbElement inputExceptionList = inAssembly.getExceptionList().getRootElement();
```

Reference…Reference…Reference…

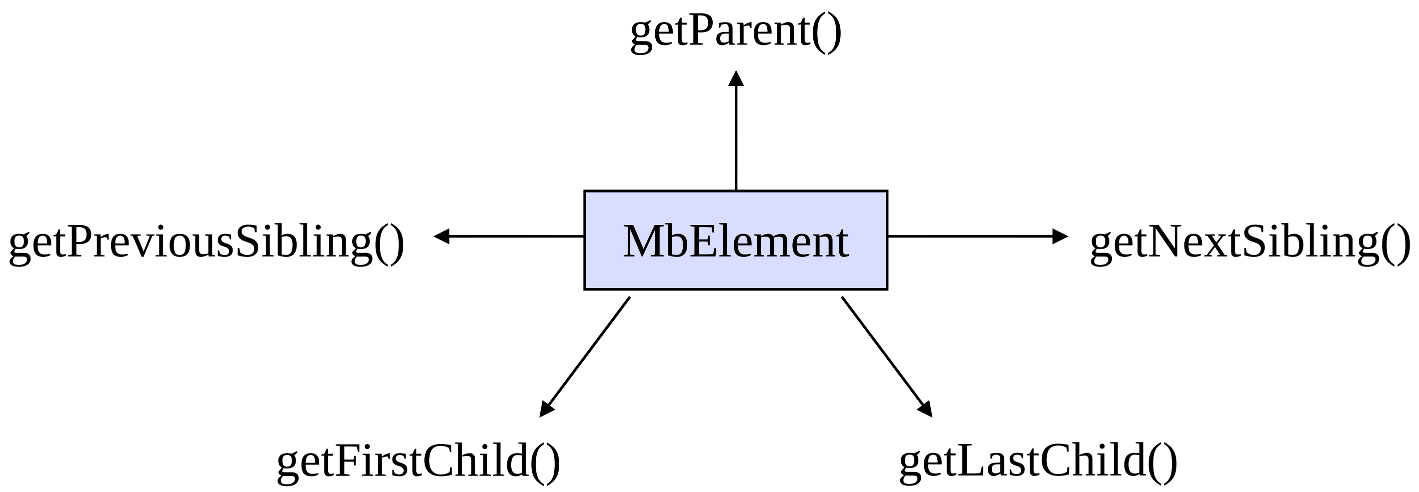
- **Each element in the tree is represented by MbElement object**
- **Java MbElement object = ESQL Reference Variable**



Properties	
HTTPInputHeader	
XMLNSC	
XmlDeclaration	
Invoice	
InvoiceNo	7
InvoiceDate	2000-12-07
InvoiceTime	12:40:00
TillNumber	3
Cashier	Mary
Customer	
Payment	
Purchases	
StoreRecords	
DirectMail	
Error	

ESQL	`DECLARE ipRef REFERENCE TO InputRoot.XMLNSC.Invoice;`
Java	`MbElement ipRef = inAssembly.getMessage().getRootElement().getFirstElementByPath("XMLNSC/Invoice");`

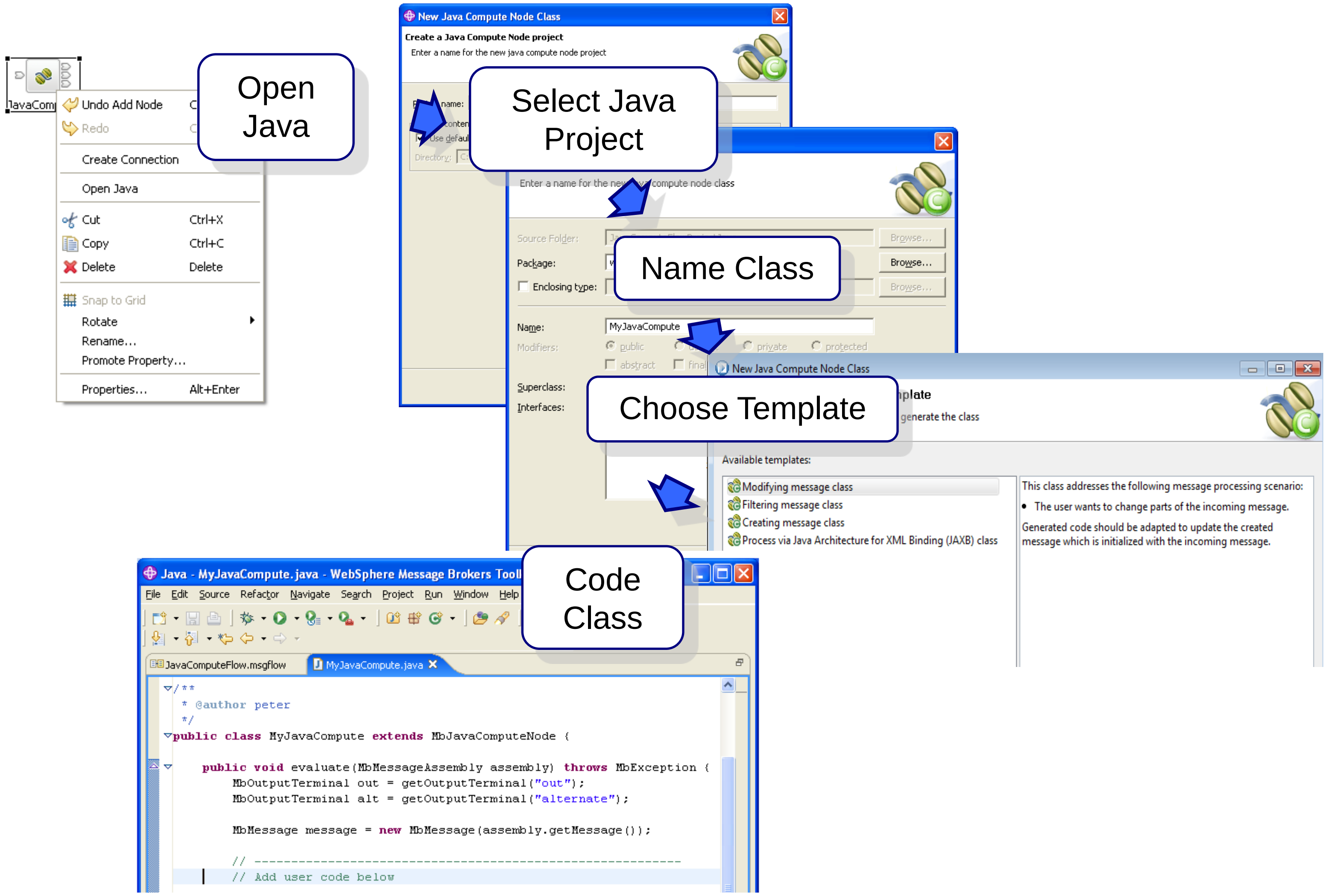
Traversing Message tree using MbElement



ESQL	`MOVE ipRef NEXTSIBLING;`
	`MOVE ipRef PREVIOUSSIBLING;`

Java	`ipRef = ipRef.getNextSibling();`
	
	`ipRef = ipRef.getPreviousSibling();`

Steps from JavaCompute Node to Code



Create a Java Compute Node project

Enter a name for the new java compute node project

Enter a name for the new java compute node class

Name: MyJavaCompute

This class addresses the following message processing scenario:

- The user wants to change parts of the incoming message.

Generated code should be adapted to update the created message which is initialized with the incoming message.

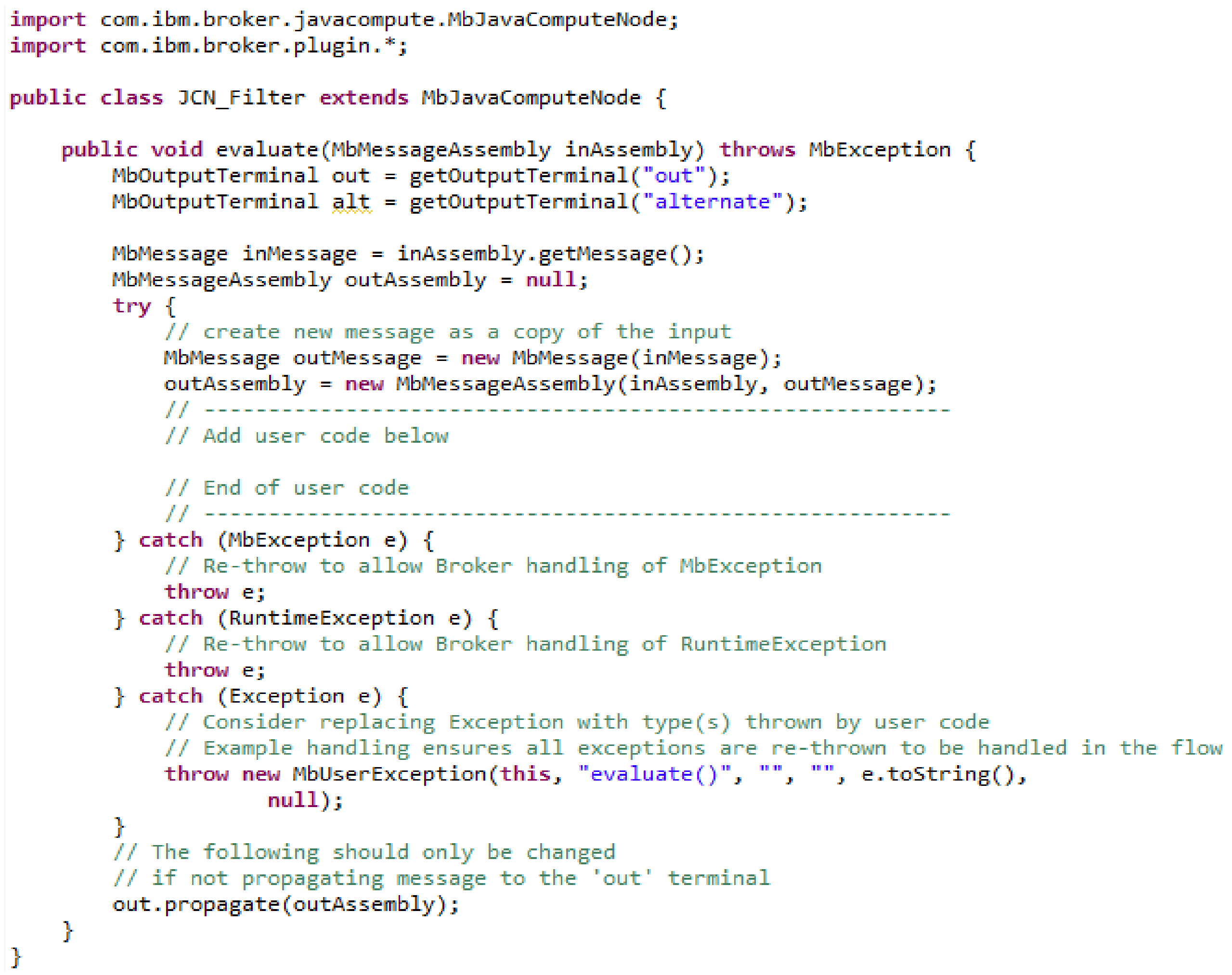
```
/**
 * @author peter
 */
public class MyJavaCompute extends MbJavaComputeNode {

    public void evaluate(MbMessageAssembly assembly) throws MbException {
        MbOutputTerminal out = getOutputTerminal("out");
        MbOutputTerminal alt = getOutputTerminal("alternate");

        MbMessage message = new MbMessage(assembly.getMessage());

        // ----------------------------------------------------------
        // Add user code below
```

Sample Auto generated code



```
import com.ibm.broker.javacompute.MbJavaComputeNode;
import com.ibm.broker.plugin.*;

public class JCN_Filter extends MbJavaComputeNode {

    public void evaluate(MbMessageAssembly inAssembly) throws MbException {
        MbOutputTerminal out = getOutputTerminal("out");
        MbOutputTerminal alt = getOutputTerminal("alternate");

        MbMessage inMessage = inAssembly.getMessage();
        MbMessageAssembly outAssembly = null;
        try {
            // create new message as a copy of the input
            MbMessage outMessage = new MbMessage(inMessage);
            outAssembly = new MbMessageAssembly(inAssembly, outMessage);
            // ----------------------------------------------------------
            // Add user code below

            // End of user code
            // ----------------------------------------------------------
        } catch (MbException e) {
            // Re-throw to allow Broker handling of MbException
            throw e;
        } catch (RuntimeException e) {
            // Re-throw to allow Broker handling of RuntimeException
            throw e;
        } catch (Exception e) {
            // Consider replacing Exception with type(s) thrown by user code
            // Example handling ensures all exceptions are re-thrown to be handled in the flow
            throw new MbUserException(this, "evaluate()", "", "", e.toString(),
                    null);
        }
        // The following should only be changed
        // if not propagating message to the 'out' terminal
        out.propagate(outAssembly);
    }
}
```

Deployment

- **JAR files (including external third party jar files) are added automatically to the BAR file; No explicit selection is required.**

- **JAR files are searched from the following path while deployment**
  - Java project
  - Workspace
  - Local File system

- **If JAR file is too large and is creating concerns in deployment, move those jar files to shared classes directory.**

ESQL Module Vs Java Node

- Every node class must extend **MbJavaComputeNode** class
- Must implement **evaluate()** method
- Names are case sensitive

ESQL	`CREATE COMPUTE MODULE JSONMockService_Compute1`
	`CREATE FUNCTION Main() RETURNS BOOLEAN`
	`BEGIN`
	
	`RETURN TRUE;`
	`END;`
	
	`END MODULE;`

Java	`public class JSONMockService_Compute1 extends MbJavaComputeNode {`
	
	`public void evaluate(MbMessageAssembly inAssembly) throws MbException {`
	
	`}`
	
	`}`

Variable Declaration and Assignment

- **Declaration**

ESQL	`DECLARE I INTEGER 1;`
	`DECLARE status CHARACTER;`
	`DECLARE isFound BOOLEAN;`

Java	`int i = 1;`
	`String status;`
	`boolean isFound;`

- **Assignment**

ESQL	`SET I = 10;`
	`SET status = 'Success';`
	`SET isFound = FALSE;`

Java	`i = 10;`
	`status = "Success";`
	`isFound = false;`

External Variable aka UDP

- No Declaration required in Java
- Use method `getUserDefinedAttribute()` directly to get the value
- Pass UDP name as parameter to `getUserDefinedAttribute()` method
- Cast the values as per the definition

ESQL	`DECLARE WaitTime EXTERNAL INTEGER -1;`
	`DECLARE SchemaName EXTERNAL CHARACTER NULL;`
	`DECLARE SendEmail EXTERNAL BOOLEAN FALSE;`

Java	`int waitTime = (Integer) getUserDefinedAttribute("WaitTime");`
	`String schemaName = (String) getUserDefinedAttribute("SchemaName");`
	`boolean sendEmail = (Boolean) getUserDefinedAttribute("SendEmail");`

Shared variables

- No shared variables in JCN ; Use Globalcache instead
- Need additional configuration to setup and enable GlobalCache
- Use appropriate method to insert, update, select and remove values from Global Cache
- Ideal for sharing data across multiple integration servers or nodes
- Lifetime of the cache data is configurable using MbSessionPolicy

Java

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");

MbGlobalMapSessionPolicy policy = new MbGlobalMapSessionPolicy(3600);
MbGlobalMap globalMap1 = MbGlobalMap.getGlobalMap("MyMapWithSessionPolicy",policy);

if(!globalMap.containsKey("Key1")) {
 globalMap.put("key1", "value1");
 }

if(globalMap.containsKey("Key1")) {
 globalMap.get("key1");
 }

if(globalMap.containsKey("Key1")) {
 globalMap.update("Key1", "25000");
 }

if (globalMap.containsKey("Key1")) {
globalMap.remove("Key1");
}
```

Operators

- **Arithmetic Operators**

ESQL	Java
+ (Addition)	+ (Addition)
- (Subtraction)	- (Subtraction)
* (Multiplication)	* (Multiplication)
/ (Division)	/ (Division)
MOD (Modulus)	% (Modulus)

- **Relational Operators**

ESQL	Java
= (equal to)	== (equal to)
<> (not equal to)	!= (not equal to)

- ## Logical Operators

ESQL	Java
AND (logical and)	&& (logical and)
OR (logical or)	|| (logical or)
NOT (logical not)	! (logical not)

- ## Assignment Operators

ESQL	Java
=	=

`SET I = I + 10;`	`i = i + 10;`
	`i += 1;`

- ## Concatenation Operators

ESQL	Java
||	+

Conditional Statements

ESQL	Java

`IF I = 10 THEN`
 	
`END IF;`
	`if (i == 10) {`
	
	`}`

`IF I = 10 THEN`
 	
`ELSE`
 	
`END IF;`
	`if (i == 10) {`
	
	`}else {`
	
	`}`

`IF I = 10 THEN`
 	
`ELSEIF I > 10 THEN`
 	
`ELSE`
 	
`END IF;`
	`if (i == 10) {`
	
	`}else if (i > 10) {`
	
	`} else {`
	
	`}`

- **Only simple case is supported**

ESQL	Java

```
CASE
UPPER(FIELDVALUE(ipRef))
WHEN 'A' THEN
WHEN 'B' THEN

ELSE

END CASE;
```
	```
	switch (ipRef.getValueAsString().toUpperCase()) {
	case "A":

	break;
	case "B":

	break;

	default:
	break;
	}
	```

Looping Statements

Loop	ESQL	Java

while Loop	WHILE LASTMOVE(ipRef) DO
	
	MOVE ipRef NEXTSIBLING;
	DELETE PREVIOUSSIBLING OF ipRef;
	END WHILE;
		while (ipRef != null) {
		
		ipRef = ipRef.getNextSibling();
		ipRef.getPreviousSibling().delete();
		}

Repeat… until	REPEAT
	MOVE ipRef NEXTSIBLING;
	
	UNTIL FIELDNAME(ipRef) =
	'REPEATING_ELEMENT'
	END REPEAT;
		do {
		ipRef = ipRef.getNextSibling();
		} **while (ipRef != null**
		&&
		!ipRef.getName().equals("REPEATING_ELEMENT"));

For loop	FOR payRef AS ipRef.Customer[] DO
	
	SET name = payRef.FirstName ||' '
	|| payRef.LastName;
	
	END FOR;
		for (MbElement payRef =
		ipRef.getFirstElementByPath("Customer");
		(payRef != null && payRef
		.getName().equals("Customer")); payRef = payRef
		.getNextSibling()) {
		
		name =
		payRef.getFirstElementByPath("FirstName") + " "
		+ payRef.getFirstElementByPath("LastName");
		}

Labelled Loop

	ESQL	Java

while Loop	X:WHILE LASTMOVE(ipRef) DO
	
	IF ipRef.Id = 100 THEN
	MOVE ipRef NEXTSIBLING;
	ITERATE X;
	ELSEIF ipRef.Id = 200 THEN
	LEAVE X;
	END IF;
	
	MOVE ipRef NEXTSIBLING;
	END WHILE;
		X:while (ipRef != null) {
		
		if ((int)ipRef.getFirstElementByPath("Id").getValue()
		    == 100) {
		ipRef = ipRef.getNextSibling();
		continue X;
		} else
		    if((int)ipRef.getFirstElementByPath("Id").getValu
		    e() == 200) {
		break X;
		}
		
		ipRef = ipRef.getNextSibling();
		}

Repeat… Until	Allowed	Allowed. Similar to above
For loop	Not Allowed	Allowed. Similar to above

Reading the input message

- Use **MbElement** to point to the "parsed" logical part of the tree
- Cast values to appropriate type
- Parser specific fields like **XMLNSC.Attribute, XMLNSC.Folder** are retrieved using **getSpecificType()** method

ESQL	Java
`FIELDNAME(ipRef)`	`ipRef.getName()`
`FIELDVALUE(ipRef)`	`ipRef.getValue()`
`FIELDNAMESPACE(ipRef)`	`ipRef.getNamespace()`
`FIELDTYPE(ipRef)`	`ipRef.getType()`
	`ipRef.getSpecificType()`
	`ipRef.getValueAsString()`

Creating the output message

ESQL

```
CREATE LASTCHILD OF OutputRoot DOMAIN('XMLNSC') NAME 'XMLNSC';
CREATE FIELD OutputRoot.XMLNSC.InvoiceResponse;

DECLARE opRef REFERENCE TO OutputRoot.XMLNSC.InvoiceResponse;

DECLARE ipRef REFERENCE TO InputRoot.XMLNSC.InvoiceRequest;

SET opRef.Customer.Name = ipRef.FirstName || ' ' || ipRef.LastName;
```

Java

```
MbMessage outMessage = new MbMessage();
MbElement outputRoot = outMessage.getRootElement();
MbElement inputRoot = inAssembly.getMessage().getRootElement();

MbElement opRef =
outputRoot.createElementAsLastChild(MbXMLNSC.PARSER_NAME).createElementAsFirstChild(Mb
Element.TYPE_NAME, "InvoiceResponse", null);

MbElement ipRef = inputRoot.getFirstElementByPath("XMLNSC/InvoiceRequest");

opRef.evaluateXPath("./?Customer/?Name[set-
value('"+ipRef.getFirstElementByPath("FirstName")+" "+
ipRef.getFirstElementByPath("LastName")+"')]");
outAssembly = new MbMessageAssembly(inAssembly, outMessage);
```

Examples

ESQL	Java

SET OutputRoot.JSON.Data.Message =
    'Hello World';
	MbElement outRoot = outMessage.getRootElement();
	MbElement outJsonRoot = outRoot
	.createElementAsLastChild(MbJSON.*PARSER_NAME*);
	MbElement outJsonData =
	    outJsonRoot.createElementAsLastChild(
	MbElement.*TYPE_NAME*, *MbJSON.DATA_ELEMENT_NAME*, ***null***);
	MbElement outJsonTest =
	    outJsonData.createElementAsLastChild(
	MbElement.*TYPE_NAME_VALUE*, *"Message"*, *"Hello World"*);

CREATE FIELD OutputRoot.JSON.Data
IDENTITY (JSON.Array)Data;
CREATE LASTCHILD OF
OutputRoot.JSON.Data TYPE
NameValue NAME 'Item' VALUE
'valueA';
CREATE LASTCHILD OF
OutputRoot.JSON.Data TYPE
NameValue NAME 'Item' VALUE
'valueB';
	MbElement outRoot = outMessage.getRootElement();
	MbElement outJsonData =
	outRoot.createElementAsLastChild(
	MbJSON.*ARRAY*, *"Data"*, ***null***);
	outJsonData.createElementAsLastChild(MbElement.*TYPE_NAM*
	E_VALUE,
	"Item", "valueA");
	outJsonData.createElementAsLastChild(MbElement.*TYPE_NAM*
	E_VALUE,
	"Item", "valueB");

Examples

ESQL	Java
`SET OutputRoot = InputRoot;`	`MbMessage outMessage = new MbMessage(inMessage);`

`SET OutputLocalEnvironment =`
`    InputLocalEnvironment;`
	`MbMessage outLocalMessage = new`
	`    MbMessage(inAssembly.getLocalEnvironment());`

`SET OutputRoot.Properties =`
`    InputRoot.Properties;`
	`MbMessage outMessage = new MbMessage();`
	`MbElement outputRoot = outMessage.getRootElement();`
	
	`MbElement inputRoot =`
	`    inAssembly.getMessage().getRootElement();`
	
	`outputRoot.addAsFirstChild(inputRoot.getFirstChild().copy`
	`    ());`

`SET OutputRoot.XMLNSC =`
`    InputRoot.DFDL;`
	`outputRoot.createElementAsLastChild(MbXMLNSC.PARSER_NAME)`
	`    .copyElementTree(inputRoot.getFirstElementByPath("DFD`
	`    L"));`

`SET OutputRoot.BLOB.BLOB =`
`    CAST('abc' AS BLOB);`
	`outputRoot.createElementAsLastChild(MbBLOB.PARSER_NAME).c`
	`    reateElementAsFirstChild(MbElement.TYPE_NAME_VALUE,`
	`    MbBLOB.ROOT_ELEMENT_NAME, "abc".getBytes());`

Modifying the message

- **Methods for setting element name/value/type**

ESQL	Java
`NAME`	`setName()`
`VALUE`	`setValue()`
`NAMESPACE`	`setNamespace()`
`TYPE`	`setSpecificType()`

- **Methods for creating elements**

ESQL	Java
`CREATE FIRSTCHILD OF`	`createElementAsFirstChild()`
`CREATE LASTCHILD OF`	`createElementAsLastChild()`
`CREATE PREVIOUSSIBLING OF`	`createElementBefore()`
`CREATE NEXTSIBLING OF`	`createElementAfter()`

Propagating the Messages

- JCN has only two out terminals
  - `Out`
  - `Alternate`

- No compute mode to select what needs to be propagated

- MbMessageAssembly constructor defines what is propagated
  - `MbMessageAssembly(MbMessageAssembly assembly, MbMessage message)`
  - `MbMessageAssembly(MbMessageAssembly assembly, MbMessage localEnvironment, MbMessage exceptionList, MbMessage message)`

ESQL	Java
`PROPAGATE TO TERMINAL 'out';`	`getOutputTerminal("out").propagate(outAssembly,true);`
`PROPAGATE TO LABEL 'abc';`	`getRoute("abc").propagate(outAssembly);`
`PROPAGATE TO TERMINAL 'out' DELETE NONE;`	`getOutputTerminal("out").propagate(outAssembly);`
`PROPAGATE TO TERMINAL 'out' DELETE DEFAULT;`	`out.propagate(outAssembly,true);`

Procedures and Functions

- **Procedures and functions are called as methods.**
- **ESQL procedure = Java method with return type void**
- **ESQL function = Java method with return type int, String etc**
- **No parameter directions**
- **Method name is case sensitive**
- **Method overloading is allowed(same method name but different parameters)**

Ways to overload a method	Example

`Number of parameters`	`add(int, int)`
	`add(int, int, int)`

`Data type of parameters`	`add(int, int)`
	`add(int, float)`

`Sequence of Data type of parameters`	`add(int, float)`
	`add(float, int)`

ESQL	Java

CREATE PROCEDURE validateAccountId (IN accountId INTEGER)
BEGIN
 	
END;
	private void validateAccountId(int accountId) {
	
	}

CREATE FUNCTION validateAccountId (IN accountId INTEGER) RETURNS BOOLEAN
BEGIN
 	
RETURN FALSE;
 	
END;
	private boolean validateAccountId(int accountId) {
	return false;
	}

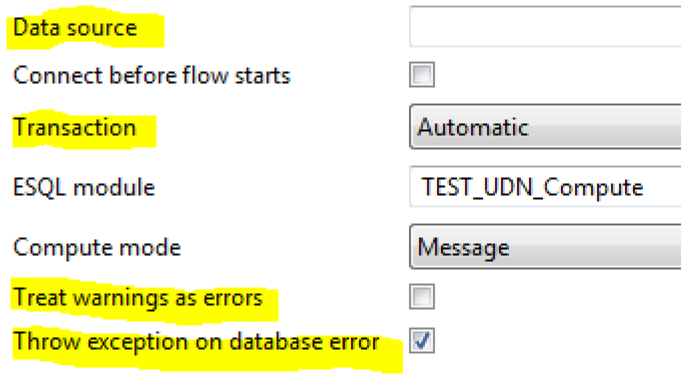
CREATE PROCEDURE CopyMessageHeaders() BEGIN
DECLARE I INTEGER 1;
DECLARE J INTEGER;
SET J = CARDINALITY(InputRoot.*[]);
WHILE I < J DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I = I + 1;
END WHILE;
END;
	public static void copyMessageHeaders(MbMessage inMessage, MbMessage outMessage)
	throws MbException {
	MbElement outRoot = outMessage.getRootElement();
	MbElement header = inMessage.getRootElement().getFirstChild();
	while (header != null && header.getNextSibling() != null)
	{
	outRoot.addAsLastChild(header.copy());
	header = header.getNextSibling();
	}
	}

Working with Databases - ODBC

- `MbSQLStatement` provides support for accessing external ODBC Database
- Ability to set Transaction Type
  - `SQL_TRANSACTION_COMMIT`
  - `SQL_TRANSACTION_AUTO`
- Call `select()` method to return the results of the query(e.g., select statement)
- Call `execute()` method when no results are returned(e.g., creating a table, deleting rows etc)
- Option to throw and handle database exceptions and warnings
- Use database state values to capture database operation result
  - `getSQLCode(), getSQLState(), getSQLNativeError(), and getSQLErrorText()`

ESQL

```
SET Environment.Rows[] = PASSTHRU('SELECT * FROM table');
```



Data source	
Connect before flow starts	☐
Transaction	Automatic
ESQL module	TEST_UDN_Compute
Compute mode	Message
Treat warnings as errors	☐
Throw exception on database error	✓

Java

```
MbMessageAssembly newAssembly = new MbMessageAssembly(inAssembly,
inAssembly.getGlobalEnvironment());
String table = "dbTable";
MbSQLStatement state = createSQLStatement(
(String)getUserDefinedAttribute("DatatSourceName"),
                 "SET Environment.Rows[] = PASSTHRU('SELECT * FROM " + table +
"');" );
state.setThrowExceptionOnDatabaseError(false);
    state.setTreatWarningsAsErrors(true);

    state.select( inAssembly, newAssembly );

    int sqlCode = state.getSQLCode();
    if(sqlCode != 0)
      {
        // Do error handling here

      }
```

Working with Databases-JDBC

- Broker supports type 4 drivers
- Create a configurable service of type JDBCProviders
- Set security settings using `mqsisetdbparms`
- Use broker Java API `getJDBCType4Connection()` to initiate the connection
- Do not close the connection. Broker manages the connection, connection pooling and lifecycle.
- Max Connection pool size is configurable
- If connection is idle for 1 minute or if the message flow completes, the broker closes the connection

Sample Code

```
public class MyJavaCompute extends MbJavaComputeNode {
    public void evaluate(MbMessageAssembly inAssembly) throws MbException {
      Statement stmt = null;
      ResultSet rs = null;

      try {
        // Obtain a java.sql.Connection using a JDBC Type4 datasource - in this example for a
        // JDBC broker configurable service called "MyDB2"

        Connection conn = getJDBCType4Connection("MyDB2", // MyDB2 is the configurable service name
                     JDBC_TransactionType.MB_TRANSACTION_AUTO);

        // Example of using the Connection to create a java.sql.Statement
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                     ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery("SELECT NAME, CITY FROM MySchema.MyTable");

        stmt.executeUpdate("UPDATE MySchema.MyTable SET CITY = \"Springfield\" WHERE Name = \"Bart\"");
        // Perform other database updates

      } catch (SQLException sqx ){
        sqx.printStackTrace();
      } finally {
        // Close the artifacts
try {
if (stmt != null)
stmt.close();
if (rs != null) rs.close();
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
//No need to close the connection. It is handled by the JDBC configurable service
      }
    }
  }
```

Which statement to use?

- Use `PreparedStatement` when
  - optional parameters are to be specified
  - values that do not convert easily to strings, for example BLOBs

```
// the mysql insert statement
        String query = " insert into accountInfo (first_name, last_name,
date_created, is_admin, num_points)"
          + " values (?, ?, ?, ?, ?)";

        // create the mysql insert preparedstatement
        PreparedStatement preparedStmt = conn.prepareStatement(query);
        preparedStmt.setString (1,
payload.getFirstElementByPath("FirstName").getValueAsString());
        preparedStmt.setString (2,
payload.getFirstElementByPath("LastName").getValueAsString());
        preparedStmt.setDate   (3, new java.sql.Date(System.currentTimeMillis()));
        preparedStmt.setBoolean(4,
Boolean.valueOf(payload.getFirstElementByPath("IsAdmin").getValueAsString()));
        preparedStmt.setInt    (5, 0);

        // execute the preparedstatement
        preparedStmt.execute();
```

- **Use `CallableStatement`**
  - to call the stored procedures and functions
- **IN parameters are specified using `setXXX()` method**
- **OUT parameters are specified using `registerOutParameter()` method**

Stroed Procedure	`CallableStatement callableStatement = conn .prepareCall("{call calculateAccountBalance(?, ?)}");`
	
	`callableStatement.setString(1, payload.getFirstElementByPath("AccountId").getValueAsString());`
	`callableStatement.registerOutParameter(2, java.sql.Types.DOUBLE);`
	
	`callableStatement.execute();`
	`Double balance = callableStatement.getDouble(2);`

Function	`CallableStatement callableStmt = conn.prepareCall("{ ? = call MYFUNCTION(?)}");`
	
	`callableStmt.registerOutParameter(1, java.sql.Types.NUMERIC);`
	
	`callableStmt.setInt(2, 100);`
	
	`callableStmt.executeUpdate();`

Exception Handling

- Can capture and handle right exception
- Can create user defined exception

ESQL	Java

`THROW USER EXCEPTION`	`throw new MbUserException(this, "evaluate()",`
	`    "", "", e.toString(),`
	`null);`

`DECLARE CONTINUE HANDLER FOR`
`    SQLSTATE LIKE '%' BEGIN END;`
	`try {`
	
	`} catch (Exception e) {`
	`}`

`DECLARE EXIT HANDLER FOR SQLSTATE`
`    LIKE '%' BEGIN END;`
	`X:{`
	`        try {`
	
	`} catch (Exception e) {`
	`break X;`
	`}`
	`  }`

`RESIGNAL`	`throw e;`

Xpath -Overview

N O T E S

- XPath stands for XML Path Language
- Works for all message types having logical tree
- Used to navigate the tree
- Can search, extract, filter and read from any part of the logical tree
- Path is separated by /
- Broker extension allows set and modify element values
- Supports Xpath 1.0 in Java

Broker extensions for Xpath 1.0

Broker specific Xpath functions	Description
`set-local-name(object)`	sets the name of the node
`set-namespace-uri(object)`	sets the namespace URI
`set-value(object)`	sets the string-value of the context node

Broker specific Xpath axes	Description
`?name`	select children called 'name'. Create one (as last child) if none exist, then select it.
`?$name`	create 'name' as last child, then select it
`?^name`	create 'name' as first child, then select it.
`?>name`	create 'name' as next sibling, then select it
`?<name`	create 'name' as previous sibling, then select it
`@name`	select attribute called 'name'. Create one if none exist

Sample code

1

```
MbXPath setMQDestinationXPath = new
MbXPath("?Destination/?MQ/?DestinationData/?queueName[set-value($queueName)]");

            setMQDestinationXPath.assignVariable("queueName",
(String)getUserDefinedAttribute("QueueName"));

outAssembly.getLocalEnvironment().getRootElement().evaluateXPath(setMQDestinationXPa
th);
```

2

```
MbXPath xpath = new MbXPath("//Item/Quantity | //Item/Author");
List<MbElement> arrayList = (List<MbElement>)inputRoot.evaluateXPath(xpath);

for (MbElement mbElement : arrayList) {

if (mbElement.getName().equals("Quantity")) {
opRef.createElementAsLastChild(MbElement.TYPE_NAME,"Qty",mbElement.getValue());
}else {
opRef.addAsLastChild(mbElement.copy());
}
}
```

3

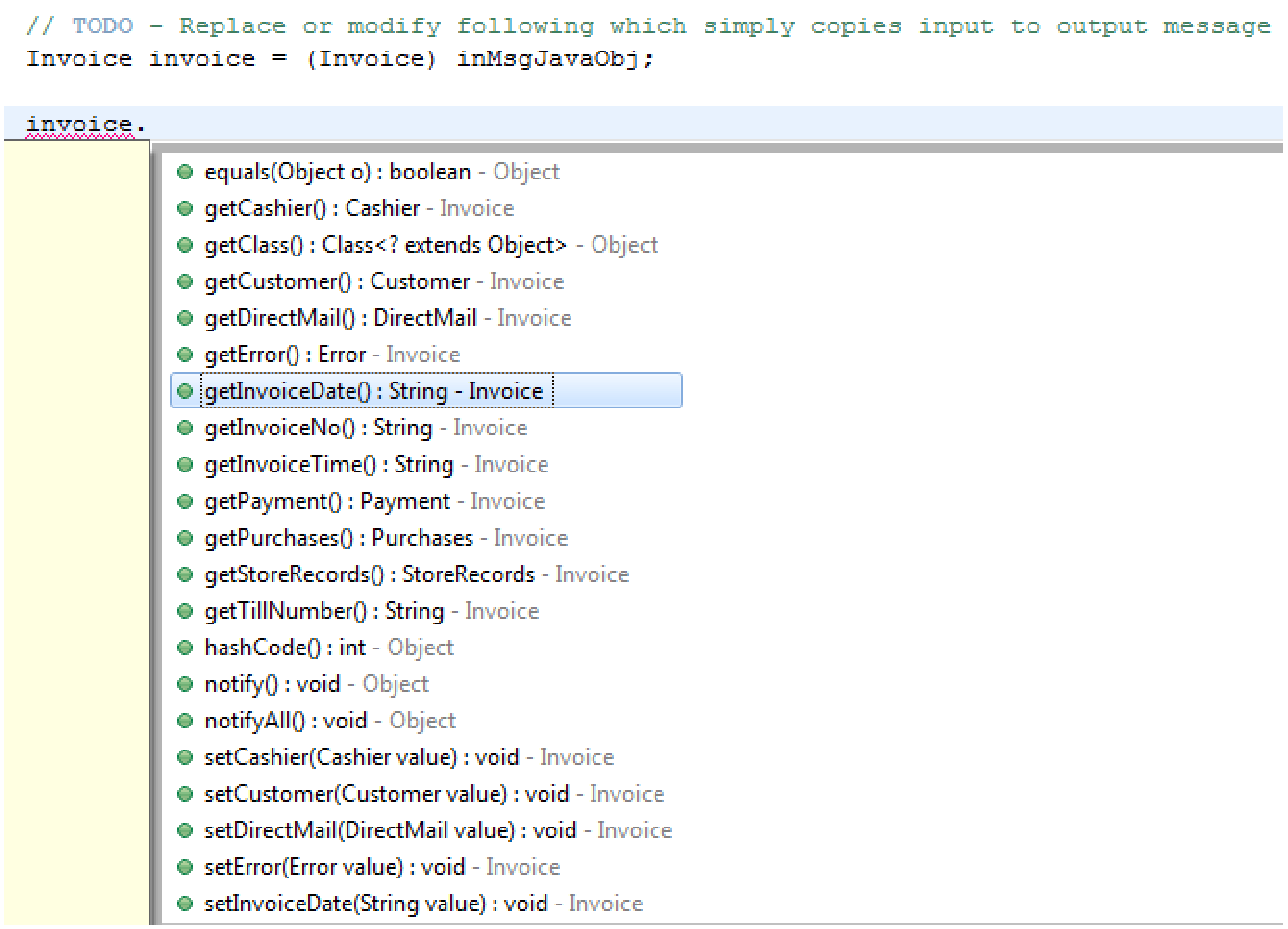
```
// the following returns a list of all chapters in the document using an XPath
// expression.
List<MbElement> chapters=
(List<MbElement>)inputRoot.evaluateXPath("/document/chapter");
MbElement chapter = (MbElement)chapters.get(0); // returns the first chapter
```

Xpath functions defined by the standard

- last()
- position()
- count()
- id()
- local-name()
- namepsace-uri()
- name()
- string()
- concat()
- starts-with()
- contains()
- substring-before()
- substring-after()
- substring()
- string-length()
- normalize-space()
- translate()
- boolean()
- not()
- true()
- false()
- lang()
- number()
- sum()
- floor()
- ceiling()
- round()

JAXB support

- **Provides an alternative to IIB Java plugin API**
- **Content Assistance available for all fields**
- **Fields are accessed using get and set methods**
- **Tree is created as per the schema and not as per the code**
- **Works for all message types**
- **Steps to create JAXB classes**
  - Choose JAXB template class when you create a JCN class file
  - Choose the XSD file which has input and output message definition
  - Select the Java project to store the JAXB object classes
  - Add necessary logic in the generated class file's evaluate() method

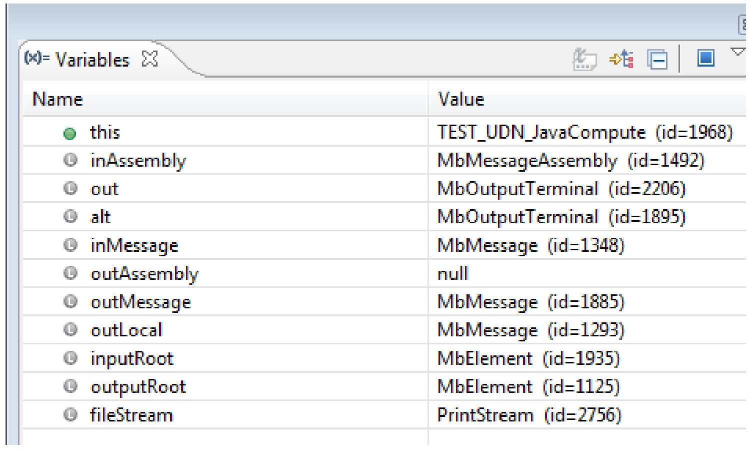


```
// TODO - Replace or modify following which simply copies input to output message
Invoice invoice = (Invoice) inMsgJavaObj;

invoice.
```

```
equals(Object o) : boolean - Object
getCashier() : Cashier - Invoice
getClass() : Class<? extends Object> - Object
getCustomer() : Customer - Invoice
getDirectMail() : DirectMail - Invoice
getError() : Error - Invoice
getInvoiceDate() : String - Invoice
getInvoiceNo() : String - Invoice
getInvoiceTime() : String - Invoice
getPayment() : Payment - Invoice
getPurchases() : Purchases - Invoice
getStoreRecords() : StoreRecords - Invoice
getTillNumber() : String - Invoice
hashCode() : int - Object
notify() : void - Object
notifyAll() : void - Object
setCashier(Cashier value) : void - Invoice
setCustomer(Customer value) : void - Invoice
setDirectMail(DirectMail value) : void - Invoice
setError(Error value) : void - Invoice
setInvoiceDate(String value) : void - Invoice
```

Debugging JCN code



Name	Value
● this	TEST_UDN_JavaCompute (id=1968)
● inAssembly	MbMessageAssembly (id=1492)
● out	MbOutputTerminal (id=2206)
● alt	MbOutputTerminal (id=1895)
● inMessage	MbMessage (id=1348)
● outAssembly	null
● outMessage	MbMessage (id=1885)
● outLocal	MbMessage (id=1293)
● inputRoot	MbElement (id=1935)
● outputRoot	MbElement (id=1125)
● fileStream	PrintStream (id=2756)

- **Use log4j or System.out.print method to redirect the values to a file**

```
//Set the FileStram to the intended logFile
            PrintStream fileStream = new PrintStream (new FileOutputStream
("c:\\trace\\testSysOut.txt"));
            //Set the System Out
            System.setOut(fileStream);

            System.out.println((String)getUserDefinedAttribute("QueueName"));
```

- **Use `MbService` class to write information to the System logs**
  - Methods available to specify Information, Warning and Error messges

Datetime manipulations

- `MbDate, MbTime and MbTimeStamp` classes are representation of the broker's ESQL date, time and timestamp types respectively

Java

```
Calendar calendar = MbDate.getInstance();
           int year        = calendar.get(Calendar.YEAR);
           int month       = calendar.get(Calendar.MONTH); // Jan = 0, dec = 11
           int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
           int dayOfWeek  = calendar.get(Calendar.DAY_OF_WEEK);
           int weekOfYear = calendar.get(Calendar.WEEK_OF_YEAR);
           int weekOfMonth= calendar.get(Calendar.WEEK_OF_MONTH);

           int hour        = calendar.get(Calendar.HOUR);        // 12 hour clock
           int hourOfDay  = calendar.get(Calendar.HOUR_OF_DAY); // 24 hour clock
           int minute      = calendar.get(Calendar.MINUTE);
           int second      = calendar.get(Calendar.SECOND);
           int millisecond= calendar.get(Calendar.MILLISECOND);
          //add one month
           calendar.add(Calendar.MONTH, 1);
         //subtract 10 days
           calendar.add(Calendar.DAY_OF_MONTH, -10);
```

Java

```
//Convert Date to String
            SimpleDateFormat sdf1 = new SimpleDateFormat("dd/M/yyyy");
            String date = sdf1.format(new Date());

            //Convert String to Date
            SimpleDateFormat sdf2 = new SimpleDateFormat("dd-M-yyyy hh:mm:ss");
            String dateInString = "26-09-2017 10:20:44";
            Date date1 = sdf2.parse(dateInString);

            //Convert Calendar to Date
            Date date2 = calendar.getTime();

            //Date comparison
            if (date1.compareTo(date2) > 0) {
                System.out.println("Date1 is after Date2");
            } else if (date1.compareTo(date2) < 0) {
                System.out.println("Date1 is before Date2");
            } else if (date1.compareTo(date2) == 0) {
                System.out.println("Date1 is equal to Date2");
            }
```

Accessing Broker Properties from JCN

ESQL	Java
BrokerName	getBroker().*getName()*
QueueManagerName	getBroker().getQueueManagerName()
ExecutionGroupLabel	getExecutionGroup().getName()
MessageFlowLabel	getMessageFlow().getName()
ApplicationLabel	getMessageFlow().getApplicationName()
LibraryLabel	getMessageFlow().getLibraryName()
NodeLabel	getName()
BrokerUserId	System.getProperty("user.name")
Family	System.getProperty("os.name")

String functions

ESQL	Java
`CONTAINS`	`string1.contains(string2)`
`ENDSWITH`	`string1.endsWith(string2)`
`LENGTH`	`string1.length()`
`LOWER`	`string1.toLowerCase()`
`LEFT`	Use Apache commons language API
`LTRIM`	Use Apache commons language API
`OVERLAY`	Use Apache commons language API
`POSITION`	`string1.indexOf()`
`REPLACE`	`string1.replace()`
`REPLICATE`	Use Apache commons language API
`RIGHT`	Use Apache commons language API
`RTRIM`	Use Apache commons language API
`SUBSTRING`	`string1.substring()`

Miscellaneous statements

ESQL	Java
BROKER SCHEMA	package com.test.jcn;
PATH	import com.test.jcn;
ATTACH	copy()
DETACH	detach()
SQLCODE	getSQLCode()
SQLERRORTEXT	getSQLErrorText()
SQLNATIVEERROR	getSQLNativeError()
SQLSTATE	getSQLState()
SAMEFIELD	is(MbElement comparisonElement)
UUIDASCHAR	UUID.*randomUUID*().toString()
UUIDASBLOB	UUID.*randomUUID*().toString().getBytes()
CARDINALITY	Use count() xpath

ESQL	Java
EXISTS	Use boolean() xpath function
LASTMOVE	Check MbElement != null
CAST	XXX.Parse()/e.g., Integer.parseInt(“100”)
ASBITSTREAM	toBitstream(String messageType, String messageSet, String messageFormat, int encoding, int ccsid, int options)
PARSE	createElementAsLastChildFromBitstream(byte[] bitstream, String parserName, String messageType, String messageSet, String messageFormat, int encoding, int ccsid, int options)

Summary

- Why Java Compute Node
- ESQL to Java Correlation Name mapping
- Deploying jar files
- Reading messages
- Writing Messages
- Xpath support
- Working with databases ODBC & JDBC
- Debugging Java Code
- String and other ESQL statement’s Java equivalent