DataPort – IoT Platform

Developers’ & Users’ Guide

Abid S. Khan
1 Table of Contents

1.0 Introduction.................................................................................................................................................. 3
1.1 Data Service Creation/Publishing .................................................................................................................. 3
1.2 Service Mediation/Orchestration .................................................................................................................... 3
1.3 Message Enrichment ...................................................................................................................................... 3
1.4 Message Routing ........................................................................................................................................... 3
1.5 Sync./A sync. Communication ....................................................................................................................... 4
1.6 Connector ...................................................................................................................................................... 4
1.7 Monitoring ................................................................................................................................................... 4
1.8 Database ...................................................................................................................................................... 4
1.9 SQL to XML/JSON conversation .................................................................................................................. 4

2.0 User interface .................................................................................................................................................. 5
2.1 Process ........................................................................................................................................................ 5
2.2 Queue ......................................................................................................................................................... 5
2.3 Connector ................................................................................................................................................... 6
2.4 Database Connection (Pool) ......................................................................................................................... 6
2.5 Timed Process ............................................................................................................................................ 7

3.0 Registration ................................................................................................................................................... 8

4.0 Example ....................................................................................................................................................... 11
4.1 Configuring a Database connection Pool ..................................................................................................... 11
4.2 Configuring a Http Connector ...................................................................................................................... 13
4.3 Connector to save data ............................................................................................................................... 15
4.4 Configuring a Process .................................................................................................................................. 17
4.5 Configuring a Timed Process ....................................................................................................................... 24
4.6 Configure a process to fetch data from the database ................................................................................... 25
4.7 How to Use a configured process in a javascript page .............................................................................. 28

5.0 Implementing a Connector ........................................................................................................................... 30
5.1 Implementing first method .......................................................................................................................... 31
5.2 Implementing second method .................................................................................................................... 32
5.3 Writing the deployment descriptor .............................................................................................................. 32
5.4 Deploying a Connector ............................................................................................................................... 33

6.0 DataPort Agent ............................................................................................................................................ 34
6.1 DataPort Agent Installation ........................................................................................................ 34
  6.1.1 Hardware minimum requirements ......................................................................................... 34
1.0 Introduction

DataPort is a cloud based software application development, integration, service orchestration and IOT platform that has built in connector to connect to SOAP, ReST, HTTP/S and database services, enabling connectivity to legacy application or SaaS application. This platform also facilitates of data storage to a replicated, hosted, managed and scalable relational database.

DataPort platform facilitates the development of software applications, Data Services, SOA enablement, and their integration (EAI) in shorter time and with lesser effort compared to traditional software development or even other IOT platforms and PAAS solutions.

With following main features:

1.1 Data Service Creation/Publishing

This software provides a very simple approach to construct new services utilizing a comprehensive set of connector and thru a pluggable, hot deployable Connector framework enabling complete software application development.

1.2 Service Mediation/Orchestration

The service mediator aids communication between system-level service consumers and service providers. As consumers request services, the mediator negotiates between the consumer and the provider to manage the service request and its delivery.

Workflow (Process) supports integrating two or more applications and/or services together to automate communication between them to fulfil a bigger task, or synchronize data in near real-time.

Such communication is often, point-to-point integration; but can be easily managed, controlled and monitored from the cloud hosted DataPort platform to make it centrally managed communication.

1.3 Message Enrichment

Consider an incoming message from source “a” with two letter state code, but the consuming application needs full state name; the message enrichment mechanism can seamlessly do that.

1.4 Message Routing

Content based routing is fully support. That is same message could be sent to different targets based on some value in the message. The same mechanism can also be used for decisions making in side flow of a process.
1.5 Sync. / A sync. Communication

Synchronous and asynchronous mode of communication is supported by Processes and Queues respectively those could be configured by using a simple web based interface. Queue can be used to launch a Process or can be externally used by any application, utilizing provided java based API.

1.6 Connector

Connector is simple mechanism to add functionality or legacy application connectivity to already a feature rich platform. A java based “application” can be developed using a very simple Connector interface and can be hot deployed as a connector.

1.7 Monitoring

Built in metrics and alerting on failures. Internal auditing logs all steps in a process for all message or request that are processed. Helps in improving process or connector that might be taking long time or failing more often.

1.8 Database

The platform offers fully managed, replicated and scalable database, that can be used by built in database connector to store data that may come from sensors, mobile phones and other sources. That stored data may later be used for analytics or other BI needs.

1.9 SQL to XML/JSON conversation

A mechanism of delivering XML or JSON as a result from SQL select statement, that is independent of database vendor. Or data in XML or JSON format coming in from an external source could be stored into Database and retrieved from database.
2.0 User interface

The landing page of the development platform has five tabs. Those tabs help to configure five major components of the platform.

The landing page of the web-based user interface of DataPort platform:

![DataPort Landing Page](image)

Figure 1 DataPort Landing Page

2.1 Process

A process is kind of a choreographing mechanism, that it is a series of steps and decisions involved in the way work is completed. We may not realize it, but processes are everywhere and in every aspect of our leisure and work lives. In DataPort this is used to create new services, transform data from existing services and to implement business process. Processes can be configured to do workflow or BPM type of activities. Each process may be configured to perform one or more tasks; each task may invoke more than one connector.

2.2 Queue

A queue is a particular kind of an abstract data type or collection in which the message/request kept in order and the principal (or only) operations on the collection are the addition of messages to the rear terminal position, known as en-queue, and removal of messages from the front terminal position, known as de-queue. In DataPort queues are used...
to facilitate asynchronous communication. Message/requests received in a particular queues are can be processed using already configured Process, or can be processed externally by other applications, using provided queue system API.

2.3 Connector

Connectors represent communication channel among components. They provide the glue for architectural designs. From the run time perspective, connectors mediate the communication among other software application or may perform some application specific task. That’s precisely what they do in DataPort. There are built in connectors that can perform most of the standard communication e.g. SOAP, HTTP/S, and Database.

Connector is one of the major components in DataPort, and provides different way to interact with external or internal application; also a new connector can be developed and deployed as a pluggable component to facilitate connectivity to any out-of-the-box unsupported interface. A Connector does not necessarily be connecting to another application, it can just do some calculation on the data that is already fetched and is in the process contexts. The connectors are divided into following for types:

1) HTTP/S, this connector can be used to interact with application that can communicate using SOAP, ReST based, XML /JSON over HTTP interfaces.

2) HTTP/S Basic Auth. connector type can also be used to interact with applications that expose SOAP, ReST based interface, but also use authentication mechanism of HTTP Basic Auth.

3) Database, this connector type can be used to interact with RDBMSs. This connector supports complete set of database operation, like Insert, Select, Update and Delete. These operations are called “Create”, “Read”, “Update”, and “Delete” and are known as “CRUD” operations.

4) Other, This type allows DataPort user to configure an externally developed connector.

2.4 Database Connection (Pool)

A database connection is the means by which a database server and its client software applications communicate with each other. This tab helps us configure database connections.

DataPort provides one connection pool to the hosted database this connection pool can be used with database connectors. However if the developer want to use some other database; this tab helps to configure a connection pool for that.
2.5  Timed Process

*Timed Process* is simply a process configured to run on scheduled time. Different processes can be chronically started at scheduled time to batch processing of some data or send certain notifications to business partners or other application in an enterprise.

Processes and Queues are the only components that are publically accessible, surely *not* without authentication. The read-only attribute called “Link” on the configuration page for Process is used to invoke a process, if the process expects an input that input can be also posted to that link, where as in the case of Queue the URL given in with Link can be used to queue a message and or retrieve a message from the top of the Queue. To see the value of the Link attribute a particular Queue or Process must be selected.

In the next chapter we will detail the steps for registration.
3.0 Registration
To use DataPort cloud application platform, registration is required. The registration process creates domain that is linked to your e-mail address. Please also note that if you use an e-mail address that from public e-mail provider like yahoo.com, outlook.com and or gmail.com, you would be sharing your account with other users of that domain. In other words if your e-mail address is “joe.smith@yahoo.com” DataPort creates domain with name “yahoo”, that means other person with e-mail address from “yahoo.com” domain will share their configuration with you. In this practice I have created an e-mail account with mail.com, and the user name is dataport@mail.com.

Here are the steps to register:

1) Open Firefox or Chrome, as IE is not fully supported
2) Go to https://datasharelink.com/
3) Click on “Sign Up (free)” button in the black bar at the top of the page, see the picture below the sign up button is circled with orange ink.

4) This will take you to the registration page where you can select PlanType, Evaluation, Small Business, Medium Business and Enterprise. Every plan has a monthly subscription fee other than the Evaluation plan.
Note: - Please use correct e-mail address as that is used to send link for account activation. Also if you are using Evaluation plan there is a 3 month evaluation period and your applications (Processes, Queues etc.) will be deactivated after that period.

5) Enter your full name, e-mail address, phone number and choose a password for your account with DataPort, and click sign up. This click will take you to following page:

As mentioned in the green text on the image above, an e-mail will be sent you; open your e-mail, and either click on that link or copy and paste the activation link to Firefox or Chrome address bar and then press enter. This will activate your account.

7) Use the e-mail address as user name and password that you entered on the registration page, logon to the DataPort Web IDE to start developing applications.

8) If you used “Evaluation” plan use the following link logon:

https://www.datasharelink.com/demo/
9) If you used any of “Small Business”, “Medium Business” or “Enterprise” plan use the following link to logon to our premium environment:

https://www.datasharelink.com/dataport/

In next chapter we will go thru an example to illustrate how this platform can be used in any organization to develop their business applications.
4.0 Example
In this example we are going to implement a small application. The target of this exercise is not to create an application but to show how most of the component work.

We will take on simple IoT related example. Let’s say you have a low power Data Service Device (DSD) sitting in some location where some sponsors’ are producing data. Those sensors could be detecting motion, heat or just distance. Sensor sends the motion signal, timestamp and location of the sensor to the DSD. That device’s disk or memory will be full in say 10 minutes of gathering data from all the sensors in that location. We need to schedule a service on this IoT Platform (DataPort) that will poll that device at every 5 minutes. Fetch all the data from the device and save that data into a table that we have created in the database. There is definitely other way to implement which would be easier, for example instead of polling the DSD, the DSD can post the data to the DataPort as well. The latter will be used in real deployments. Our API this is part of the DSD deployment is capable of sending data on regular intervals to the cloud hosted platform.

A connection pool is provided pre-configured for the hosted database platform. Alternatively developer can configure a connection pool to the database of their choice. To complete this example we will configure two connectors; first http connection that will fetch data from the DSD, second database connector that will save data to the table that we have created in the database.

Then we will configure a Process with two task, first task will fetch the data from the DSD and second Task will save the data to the database table. To poll the DSD at a fix frequency at every 5 minutes; that process will be timed using the Timed Process configuration page.

*Note:* For all components on the DataPort Web IDE, please first just enter the value for name and description fields and press the `save` button.

Before we start to develop this example application on the DataPort platform; if you have not already have registered please go and register. (see chapter 3 for registration). Let’s proceed with the development of our example application. We will configure database the connection pool first.

*Note:* This step is not really require if you plan to use the hosted database that is provided with DataPort.

4.1 Configuring a Database connection Pool
For pretty much every application some data is persisted and retrieved from a Relational Database Management System (RDBMS). To facilitate this in most performant and reliable way DataPort provides this mechanism to configure a connection pool so the connections can be shared between different user requests. One connection pool is provided with each domain to the hosted replicated and managed database, but to connect to an external database a connection pool can be configured. To configure/create a new Database Connection Pool you need:
Figure 2: Database Connection Pool Configuration

As shown in the screen above following values are required:

- **Connection Name**: Name for the connection pool to identify which Database it is connected to.

- **Connection Desc**: Description of the of the connection pool

Press the “Save Connector” button, and after this connector shows up in the left pane select it and then enter the other information.

- **User Name**: Database user’s login name to authenticate to establish a connection.

- **User Password**: Database user’s password to authenticate to establish a connection.

- **Driver Class**: JDBC Driver class

- **Primary Conn. String**: JDBC connection string e.g. `jdbc:oracle:thin:@myhost:1521:orcl`

- **Secondary Conn. String**: JDBC connection String to a failover database.
**Max. Connection:** Number of connections this pool will maintain.

Now let us say we need to our marketing database that is on the machine with ip address 10.102.2.101, this is an Oracle database, with a user name marketDBUser and same as password, and we named it MarketDB with max connection 5 and 3 number of tries to establish a connection. By entering all that values and clicking on **“Save Connection”** button:

![Configured Database Connection Pool](image)

**Figure 3: Configured Database Connection Pool**

Here you have it, a database connection pool with MarketDB name is now configured and ready to be used by the “database connector”.

### 4.2 Configuring a Http Connector

Now let’s define a HTTP connector to pull data from an external web service. As described before HTTP connector can be configured to fetch data from a WebService, SOAP, and/or ReST services. Let us say that we need to poll data the URL for ReST service is [http://172.17.0.203/demo/incident/data](http://172.17.0.203/demo/incident/data) with other values like “Max Tries” as if the fetch fails how many tries will be made, how long to delay between failed tries that is represented in “Retry Delay”. “Connection Timeout” is how long to wait to establish a connection, and “Response Timeout” is how long to wait for the response. Please note that the name of this connector is “FetchIncidentData” and that is what was used for “Process Context” for the variables and “Input Node” values. What
this represents that you can use any connector’s input or output as an input to any connector in that
articular process.

**Note:** For using an output of a connector that particular connection must be
configured in a task prior to the task that is using the data.

**Note:** if there was some XPath added to the context the value for “Input Node”
could look like “{FetchIncidentData}.root.event”. 
4.3 Connector to save data

Now let’s configure a database connector. First click on the tab connector, and then press button labeled “New Connector” and then select “Database” from smaller tabs. Now enter “SaveIncidentData” in the “Connector name” field and “Connector to save incident data” in the “Connector Desc.” field. And press the “Save Connector” button.

After this new connector shows up in the left pane select it.

This connector takes data from the xml message that will be inserted into the managed database. Let’s say the data that we are getting from some external service looks like this XML document:

```xml
<root>
  <event>
    <time>2015-07-19T19:56:05-0600</time>
    <name>Application Outage</name>
    <body>Check Network</body>
  </event>
  <event>
    <time>2015-07-19T19:59:19-0600</time>
    <name>Application Outage</name>
    <body>CPU usage above 80%</body>
  </event>
</root>
```

In the message above to make it readable in this document, I have used small object called “event” and have only displayed two records, the object can be any sized, and any number of them can be in a message. To configure a connector that takes can insert all record in the hosted database the configuration would look like following:

Other than then the name and description everything else is worth noting. The SQL statement looks like this:

```
INSERT INTO EVENT_LOG (start_time, event_type, event_desc) values (:start_time,:event_type,:event_desc)
```

There are three variable in this SQL statement and are exactly named as column names in the database table called “EVENT_LOG”, the only difference is each variable has a character “:” (Colon) in front of them. Each variable is also defined in the lower pane named “SQL Variables”. In this pane the variable names do not have colon “:”. And there are other three properties for each variable, we will come back to “Process Context” but XPath is the name of the node in the message with a character “.” (dot) before them. “Data Type” is to tell the DataPort what type of the data is defined in the database. Default value is used if the when there is no value for that variable in the incoming message.
There another value on this screen shot “XPath to Entity” this holds the XPath to the record node. Please note this also has a character “.” in front of it, this how it look like:

![DataPort screenshot](image)

**Note:** In DataPort XPath is defined as “root.event.name” would lead to values “Application Outage” in the first event.

Like “Process Context” the value for “Input Node” are in this example are the name of the connector fetching the data from the external service, i.e. “FetchIncidentData”. And here are two things that needs to be remembered 1) you would see in the 4.4. section that we will configure the initial task that is the task which actually invoke “FetchIncidentData” connector; this is significant because if a connector is reference in the “Input Node” or in “process context” that must be configured to have performed before the that is using it reference.
4.4 Configuring a Process

Process consists of Tasks and Transitions, Each Task in a process can have one or more connectors configured. If there is more than one connector configured they will be started in parallel. To run connector one after another (in serial) one connector per Task could be configured.

In the example that we are working with; we are going to configure a process that uses “FetchIncidentData” connector to fetch data in the “initial” Task and then uses connector “SaveIncidentData” to save the data into database, in the second task.

Each Process must have a task that is configured as first task. For that purpose I will detail each step.

1) Click on the “Process” tab in the bigger tabs, your page should look like this:

Since this is going to be our first process, the screen is empty.

2) Enter “ProcessIncidentData” in the “Process Name” field, and “Process to save incident data” in the “Process Desc” field, and click on button “Save Process” you page should look like this, note there is now a process in the left pane:
3) Select the process and change the “Max Process Time” from -1 to 10000 (this value is milliseconds). Since we are planning to use this process “Timed Process”. We are not going to see the output of this process we don’t need to select the input and output formats.

4) Click on “Save Process” and re-select it from the left pane. It would look like this:
5) Click on “New Task”, enter “FetchDataTask” in the “Task Name” field and “Task to fetch data using FetchIncedentData connector” in the “Task Desc” field.

6) Click on “Save Task” button, and re-select the task when it appears in the left pane. You would see the configured connectors will show up in the “Available Connectors” pane. The page should look like this:
7) Select the connector “FetchIncidentData: FetchIncidentData” and press the arrow that is pointing toward right, and press the “Save Task” button.

8) When the task re-appears in the left pane, select it, and your Task configuration should look like this:
9) Since we have one the task that we need this process to start as soon as this process would be started. Select the top node in the left pane, that is the “Process” we are configuring.

10) Select the “FetchDataTask” in the “Initial Task Key” drop down list and press “Save Process”,

11) Select the “ProcessIncidentData: Process to save ……” from the left pane. You would notice the value you selected in the “Inittial Task Key” field is there. Your page would look like this:
Let's create the so far this process will fetch data from the connector “FetchIncidentData” via FetchDataTask of the process and that data will be returned to the caller if one puts the following URL with authentication token, the URL is displayed in front of the label named “Link :”.

https://www.datasharelink.com/demo/MessageService/mail/process/ProcessIncidentData

Click on “New Task” button because we need to save the data that we are getting from the FetchDataTask. Enter “SaveDataTask” in the “Task Name” field and “Task to invoke save data Connector” in the “Task Desc” field, and click on “Save Task”.

Select “SaveDataTask” in the left pane your process configuration page should like this, some user have complained that this page does not look like this, if that is what you experience too, select the process node and reselect the “SaveDataTask:...” node again from the left pane.
15) Select the “SaveIncidentData” from the “Available Connectors” list and press the button with arrow pointing to right, then press “Save Task” button. Select the “SaveDataTask....” button from left pane and your page should look like the following:
16) Now select the “FetchDataTask”, and in the “Default Next Task” drop down list, select “SaveDataTask”, and click on the “Save Task” button, this will tell the DataPort that as soon as connector(s) configured in the “FetchDataTask” complete their work, the which next “SaveDataTask” will started.

### 4.5 Configuring a Timed Process

A “Timed Process” is nothing but a scheduled process, this scheduling system is very important for IOT environments where a processing of data from sensors or other system may be batch processed on a fixed frequency or data be pushed to another system for analytical processing.

Configuring a process to run on any specific frequency is very easy. Click on the “Timed Process” tab. In this case again you would see an empty left pane, enter “TimedFetchIncidentData” in the “Timed Process Name” field, and enter “Scheduled processing of incident data” in the “Timed Process Desc” field. And press “Save Timed Process” button.

Select the “TimedFetchIncidentData” timed process from the left pane, and select 05 in the “Minute” drop down list. Also select “ProcessIncidentData” from the “Process” drop down list and press “Save Timed Process” button.

That will configure this application to run “ProcessIncidentData” at every five minutes.
4.6 Configure a process to fetch data from the database

Since we know the scheduled process “Timed Process” is going to run every five minutes and save data to table named even_log, we can just configure one more connector that fetch data from that tables, and we will configure another process to fetch data just to see how this works.

Click on “Connector” tab and then select “Database” tab in the smaller tabs. And Click “New Connector” button. Enter “ShowEventData” in the “Connector Name” field and enter “Show the saved data” in the “Connector Desc” field and press “Save Connector” button. Select the connector we just save and enter following sql

Select count(*), event_desc from event_log group by event_desc

The “SQL statement” field, select the “Database Connection” that we selected in the connector “SaveIncidentData” and press “Save Connector” button. After we select the “ShowEventData” in the left pane the page should look like following:

To use the connector that we just configure to see saved data, we need to configure a new process.

- Click on the Process tab, and click “New Process” button.
- Enter “ViewData” in the “Process Name” field and “Process to view data” in the “Process Desc” field and press “Save Process” button.
- Select “ViewData” in the left pane then press “New Task” button.
Enter “ViewDataTask” in “Task Name” field, and enter “Task to view data” in the “Task Desc” field and press “Save Task” button, and then select this task from left pane.

You would see three connectors in the “Available Connector” list select “ShowEventData:…” connector and press arrow that is pointing to right.

After the “ShowEventData” connector appears in the “Selected Connectors” pane, press the “Save Task” button.

If you select the “ViewDataTask” from the left pane you page would like like:

Now select the “ViewData” process from the left pane and select “ViewDataTask” from “Initial Task Key” drop down list.

Enter 10000 in the “Max Process Time” field.

If you want response from this process (rest service) as XML, select the “Output format” accordingly. I am selecting JSON

Press “Save Process” button.

The process now should look like this:
4.7 How to Use a configured process in a javascript page

The processes configured are rest service and can be used in any webpage or other application as such. And since these are protected services on a valid user can see. First we would need to configure a login page, the java script to validate login credentials would look like this:

```javascript
function logon1()
  
  $('#logon_form').submit(function(event)
    $.get("https://datasharelink.com/demo/ss/gt", $(this).serialize(),
    function(xml) {
      var tkn = $(xml).find('token').first().text();
      if((tkn === null) || (tkn === 'undefined') || (tkn.length === 0)) {
        var failed1 = $(xml).find("login-failed").first().text();
        if(failed1 === 'true') {
          alert('logon failed');
        }
      } else {
        $.cookie('external_token', tkn);
        window.open("dataoutput.html", '_self');
        retResult = true;
      }
    }, 'xml');
  });
</script>
```

Please note we are setting a cookie in above code.
To use any of the process (rest) configured following javascript code can be used, please not the yellowish box around the URL for the process that we configured to fetch data, also note that we are setting an header named “api_ax_token” and setting a value of the token that we got from the login javascript function:

```javascript
function drawchart() {
  var tok = $.cookie("external_token");
  $.ajaxSetup({
    headers: { "api_ax_token": tok } 
  });
  $.ajax({
    url: "https://www.datasharelink.com/demo/MessageService/mail/process/ViewData",
    data: {},
    dataType: "json",
    success: function (jsonData) {
      //Your code to display jsondata goes here
    },
    error: function (jqXHR, textStatus, errorThrown) {
      alert(textStatus + "\n" + errorThrown);
    }
  });
}
```

This example should give a lot of user a good starting point.
5.0 Implementing a Connector

Implementing a connector is very simple; following very simple java language interface needs to be implemented:

```java
public interface Connector extends Serializable {

    /**
     * Output type represents file that contain XML
     */
    public static final int OutputTypeFileNameWithXMLContents = 1;

    /**
     * Output type represents file that contain plain text
     */
    public static final int OutputTypeFileNameWithTextContents = 2;

    /**
     * Output type represents String with XML contents
     */
    public static final int OutputTypeXMLString = 3;

    /**
     * Output type represents String with TEXT contents
     */
    public static final int OutputTypeTextString = 4;

    /**
     * Output type represents internal message constructs.
     */
    public static final int OutputTypeInternalMessage = 5;

    /**
     * This method is called by transaction system
     * @param cc -- Contains all the entries made using UI.
     * @param objIn -- may be String or Message
     * @param aTransactional -- Transaction Context
     * @return may return String or Message based on the implementation
     */
    Object get(ConnectorConfiguration cc, Object objIn, Transactional aTransactional);

    /**
     * @return one of the five output types defined above.
     */
    int getOutputType();
}
```

Figure 4 Connector Interface
The above connector interface is provided in the Connector API jar that can be downloaded from DataShareLink.com web site. As you can see there are two methods that needs to be implemented.

### 5.1 Implementing first method
The first method is:

```java
Object get(
    ConnectorConfiguration cc,
    Object objIn,
    Transactional aTransactional
);
```

This method is called by the DataPort run-time. And an instance of class ConnectorConfiguration is passed in by the DataPort run-time. This instance that has everything that the user configured thru the web interface.

For example to get values properties configured in the “Other” tab, can be obtained by using using this code

```java
cc.getParameter("PropertyName");
```

The second parameter of type Object “objIn” can be instance of java.lang.String class or the instance of Message this class is provided in the API jar, and represents an internal way of representing a XML document. This instance of class Message can be used to traverse thru the hierarchal data structure of the XML data nodes.

If the objIn is an instance of Message you can get to the child node by using:

```java
Message m = ((Message)o).getMessage("root.event");
```

And if the child node is a leaf node, you can get the value of that node by using something like this:

```java
Operand o = ((Message)o).getMessage("root.event.name");
```

The third parameter is the instance of implementation of interface Transactional, that will also be passed in by the DataPort run-time. This can be used to get to the memory scopes. Those memory scopes are maintained by the DataPort. And can be used to get input to the message or tasks, and can be used to get to the output of any connector.
If the objIn is an instance of Message you can get the post parameters from the web browser values by using something like:

```java
Operand o = aTransactional.getValue("request.data.symbol");
```
(in this example the input variable name was “symbol” on the html page)

5.2 Implementing second method
The second method that needs to be implemented:

```java
int getOutputType();
```

This method is very simple to implement you just have to return one of the constants shown in the picture labeled “Connector Interface” in the last page and this method is used by the DataPort run-time to understand the output of the connector.

5.3 Writing the deployment descriptor
The developer is also produce XML document describing the connector so that if the connector is uploaded thru the web UI. And the name of this XML must be “Connector.xml”. That XML should look like as below:

```xml
<Connector>
  <Name>CSVConnector</Name>
  <Class>com.ask.data.port.connector.HttpCSVConnector</Class>
  <Description>
    This connector receives CSV data from given URL and converts that into XML. This connector returns file name that contains XML output
  </Description>
</Connector>
```

Now the structure of the jar file must look following:

```
|com
|lib
  |ccore.jar
|META-INF
  |Connector.xml
  |MANIFEST.MF
```

In the “com” folder there could be nested folders that contain usual java packages and classes that the developer has developed.
The folder “lib” folder must have the connector API jar, as named “ccore.jar” above. And also can contain other jars that were used to develop this connector.

The META-INF folder is self-explanatory; it has a MANIFEST.MF file that is created by the java jar utility. And the other file “Connector.xml” is explained in this section earlier. A jar containing the above folder structure can be created using the java jar utility. Please name the jar appropriately so that its easy for you to recall what function that particular connector in case you have multiple connector deployed.

### 5.4 Deploying a Connector

To deploy a Connector, select the tab “Connector” from the bigger tab. And click on the “Upload Connector” that is highlighted with an orange box in the picture below:

![Connector Deployment](image)

After the “Upload Connector” button is pressed you would see a pop up window with title “Connector Archive upload” that would look like the following:
Select the “Choose file” button and browse to the jar file on your system and select the jar file, and press button “Open” on the file the selection window. After the control comes back to the window above, you should see the file name right next to be “Choose file” button; where it says “No file chosen”. Then press the button labeled “Upload”. Now this new connector is available for you to use.

6.0 DataPort Agent
DataPort agent is a configurable tool to schedule data delivery from data sources that a business uses for its day to day business. This agent gathers data from various sources i.e. web services, Relation Database(s), CSV or flat files, and securely transfer data to the cloud DataPort.

6.1 DataPort Agent Installation
DataPort agent can be installed on a windows, unix or linux computers.

6.1.1 Hardware minimum requirements
- 16 GB Memory
- 500GB Hard Disk free space.
- GHz or faster quad core processor
Download tomcat and install it on a computer this computer does not have to be any high end server and entry level server should be enough. And this computer may have windows operating system or Linux, Linux (CentOS) is preferred.

Following six files are included in the distribution zip file.

1. ds.war
2. ds-keystore.jks
3. connection-pools.xml
4. data-services.properties
5. data-services.xml
6. prod-key.txt

Let us assume you tomcat is installed in /var/opt/tomcat.

Set an environmental variable CATALINA_HOME to /var/opt/tomcat

Copy file “ds.war” (numbered 1 above) into folder /var/opt/tomcat/webapps

Copy file “ds-keystore.jks” (numbered 2 above) into folder /var/opt/tomcat/bin

Copy all remaining files (connection-pools.xml, data-services.properties, data-services.xml) into folder /var/opt/tomcat/lib.

The file “prod-key.txt” (numbered 6 above) contains product key data and client identity data that is periodically sent over to the DataPort for verification, and for security purpose.

**Note:** The reason these files were not bundled inside the war so they can be modified without un-assembling and re-assembling the war file. Files “ds.war” and “ds-keystore.jks” should never need to be modified, unless an update is provided by DataShareLink Ltd.

Set following –D java propertie in the either the startup.sh/startup.bat or catalina.sh/catalina.bat:

1. keyFileLocationWithAbsolutePath
2. javax.net.ssl.trustStore
3. javax.net.ssl.trustStorePassword

The first property to set the location of 6th file in the list with 6 items above, that file is used to authenticate the client.

The 2nd and 3rd properties are Java’s properties to configure the security keys and password, that the data that is being sent back and forth to and from cloud service is encrypted.
6.1.2 Instruction to modify configuration files

6.1.2.1 Modifying connection pools file

To add a connection configuration to any database in the copy the section “pool-info” in the picture shown below and change the values. As explained in the file

```xml
<connection-pools>
  <pool-info>
    <!-- Name of the pool that uniquely identify this pool in this file -->
    <pool-name>FinancelSystemDatabase</pool-name>

    <!-- Database user name that has at least read access to the database -->
    <db-user-name>data_analysis_user</db-user-name>

    <!-- Password for the user -->
    <db-user-password>database user's password</db-user-password>

    <!-- Driver class for the given database
    for Oracle: oracle.jdbc.OracleDriver
    for MySql : com.mysql.jdbc.Driver
    for Postgresql: org.postgresql.Driver
    -->
    <driver-class>oracle.jdbc.OracleDriver</driver-class>

    <!-- jdbc connection url
    Example for Oracle :
    jdbc:oracle:thin:@/10.10.5.202:1521/financeldb
    Example for MySql :
    jdbc:mysql://10.10.5.202:3306/financeldb
    Example for Postgresql:
    jdbc:postgresql://10.10.5.202:5740/financeldb
    Please use correct values for host name/ip address, port and database name above values are for example only.
    -->
    <connection-url-db>jdbc:oracle:thin:@/10.10.5.202:1521/financeldb</connection-url-db>

    <!-- how many connection this pool should be initialized with -->
    <initial-pool-size>3</initial-pool-size>

    <!-- Please make sure this following number is equal to how many parallel/concurrent queries would be running against this database, if this number is lower than that the system will still run but those queries will be serialized. -->
    <pool-size-limit>5</pool-size-limit>
  </pool-info>
</connection-pools>

6.1.2.2 Modifying data services file

To add new data polling service from database to post to the cloud service, copy the section “service-info” of the file as shown below and modify the nodes as explained
<data-services>
  <service-info>
    <!-- None of the service can be used to test the service, URL semantics should be followed--> 
    <service-name>SensorData</service-name> 
    <!-- URL to post to -->
    <dataport-url>https://www.doshorelink.com/demo/MessageService/doshorelink/process/EventHistory</dataport-url> 
    <!-- None of the pool configured in connection-pools.xml -->
    <pool-name>SensorDatabase</pool-name> 
    <!-- SQL select statement, that return data this service is supposed to provide -->
    <sql-select-statement>select start_time, event_type, event_desc from event_log</sql-select-statement> 
    <!-- SQL to xml connector if you are not implementing action for this service -->
    <service-class>com.ssk.data.engine.DataSender</service-class> 
    <!-- recurring tag and its children tags are optional -->
    <recurring>
      <!-- Minutes tag is optional, if provided it must have a value between 0 and 59 inclusive -->
      <minutes>58</minutes>
    </recurring>
    <!-- hours tag is optional, if provided it must have a value between 0 and 23 inclusive -->
    <hours></hours>
    <!-- day-of-week tag is optional, if provided must have a value between 1 and 7 inclusive, Sunday being 1 -->
    <day-of-week></day-of-week>
    <!-- day-of-month tag is optional, if provided must have a value between 1 and 29 or 30 or 31 -->
    <day-of-month></day-of-month>
    <!-- month tag is optional, if provided must have a value between 0 and 11, January being 0 -->
    <month></month>
  </service-info>
</data-services>