LISTEN.
THINK.
SOLVE.®

DCS ADAPTER

RELEASE 2.1

TECHNICAL MANUAL

PUBLICATION DCTMAD-GR001C-EN-E-DECEMBER-2017
Supersedes publication DCTMAD-GR001B-EN-E

Allen-Bradley · Rockwell Software

Rockwell Automation

Contents

Chapter 1 Introduction 1

Typographical Conventions 1

List of Abbreviations 2

Integration Touchpoints 3

Chapter 2 Architecture of an MES Integrated with the DCS Adapter 5

Responsibility of the Components 6

Error Handling 6

Chapter 3 Integration into an MES 9

Prerequisites Checklist 9

Installing the DCS Adapter 9

Setting up the DCS Adapter Configuration 10

Making a Message Broker Available 10

Connecting an MES Client to a DCS 11

Implementing Listeners 11

Configuring the DCS Adapter 13

Simulating a DCS with a Java Mock 14

Testing the MES Client without a DCS 15

Chapter 4 Integration of a DCS 19

Channels for Integration Touchpoints 20

XML Schema of the Interface 20

- General Structure of the XML Requests and Replies 21
- Request: ProcessCreateDCSBatch 21
- Reply: AcknowledgeCreateDCSBatch 23
- Request: GetDCSAlarmEvents 24

Reply: ShowDCSAlarmEvents.. 24
Request: GetDCSBatchValues.. 26
Reply: ShowDCSBatchValues.. 27
Request: ProcessOrderContext 28
Reply: AcknowledgeOrderContext 29
Request: ProcessConsumedMaterial 30
Reply: AcknowledgeConsumedMaterial 31
Request: ProcessProducedMaterial 32
Reply: AcknowledgeProducedMaterial 32
Request: GetBatchInformation 33
Reply: ShowBatchInformation 34
Example of an MSB Implementation Based on EIHub 36
Testing the DCS Integration without an MES Client.............................. 38

Chapter 5 Extending the Standard 39
Extending Standard Implementation of Listeners in an MES Client 40

Chapter 6 Reference Documents 41

Chapter 7 Revision History..43

Index 47

Figures

Figure 1: Architecture 5
Figure 2: TestDCSAdapter tool 15
Figure 3: Adding alarm events to the mock 16
Figure 4: Adding batch values to the mock 17
Figure 5: Stopping and starting listener mocks 18
Figure 6: Example of an MSB flow 36
Figure 7: Example of an MSB mock flow 37
Figure 8: Create DCS batch tab of the TestDCSAdapter tool 38

Chapter 1

Introduction

This manual is intended for system integrators who connect a Manufacturing Execution System (MES) with one or more Distributed Controls Systems (DCS) by means of the DCS Adapter.

The DCS Adapter provides the possibility to send standardized requests between the MES and any DCS and to receive standardized replies.

For ease of handling, typically a middleware component (Manufacturing Service Bus (MSB)/Enterprise Service Bus (ESB)) is used to transform the MES requests into the data structure and technology of a specific DCS.

This chapter details the supported integration touchpoints (page 3); the following chapters describe the architecture of an MES integrated with the DCS Adapter (page 5), the integration into an MES (page 9), the integration of a DCS (page 19), and how to extend the standard (page 39).

Typographical Conventions

This documentation uses typographical conventions to enhance the readability of the information it presents. The following kinds of formatting indicate specific information:

Bold typeface Designates user interface texts, such as

- window and dialog titles
- menu functions
- panel, tab, and button names
- box labels
- object properties and their values (e.g. status).

Italic typeface Designates technical background information, such as

- path, folder, and file names
- methods
- classes.

CAPITALS	Designate keyboard-related information, such as ■ key names ■ keyboard shortcuts.
`Monospaced typeface`	Designates code examples.

List of Abbreviations

In this document, the following abbreviations are used:

Abbreviation	Definition
B2MML	Business to Manufacturing Markup Language
DCOM	Distributed Component Object Model
DCS	Distributed Control System
ESB	Enterprise Service Bus
JAR	Java ARchive
JMS	Java Message Service
MES	Manufacturing Execution System
MSB	Manufacturing Service Bus
S88	ANSI/ISA-88 standard
S95	ANSI/ISA-95 standard
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition

Integration Touchpoints

The DCS Adapter supports the following touchpoints between an MES and the DCS:

- Create DCS Batch
  Create a new batch in the DCS for a particular DCS master recipe and a set of parameters.
  This is typically triggered from the MES when a specific part of the MES recipe shall be executed automatically by a DCS.
- Get DCS Alarms
  Receive GMP-relevant alarm events from the DCS.
  This is typically done to document the alarms as MES exceptions, hence it uses the review-by-exception features of the MES.
- Get DCS Batch Values
  Read values of the batch report values from the DCS.
  This is typically done to re-use the values in MES processing/calculations or to add them to the MES batch report.
- Set Order Context
  Send information about an MES order including the list of material parameters (input, output, and transfer materials).
  This is typically done from the MES when a specific part of the MES recipe shall be executed automatically by a DCS.
- Process Consumed Material
  Receive information about material consumptions (aka goods receipts) performed on the DCS as part of a specific order.
  This is typically done so that the MES can apply the corresponding changes on the inventory and document the consumed materials in the MES batch report.
- Process Produced Material
  Receive information about material production (aka goods issues) performed on the DCS as part of a specific order.
  This is typically done so that the MES can apply the corresponding changes on the inventory and document the produced materials in the MES batch report.
- Get Batch Information
  Read all relevant information of a batch from the MES.
  This is typically done from the DCS to retrieve the batch status and other information relevant for automatic execution.

Chapter 2

Architecture of an MES Integrated with the DCS Adapter

The figure below illustrates the architecture of all involved components.

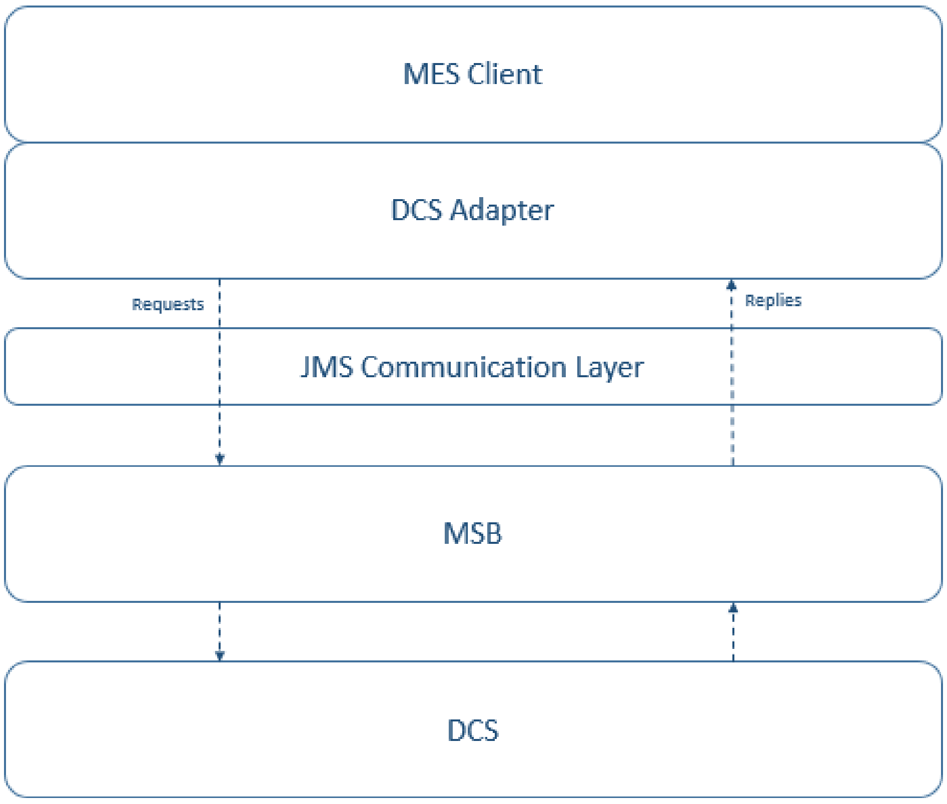


Figure 1: Architecture

The DCS Adapter typically runs within the MES layer. The MES client calls the DCS Adapter to communicate with the DCS. The DCS Adapter creates and sends a request via the Java Message Service (JMS) communication layer.

The Manufacturing Service Bus (MSB) is a middleware component that communicates with the MES via JMS and with a DCS via the DCS-specific technology. The MSB adopts, translates, and routes the defined MES requests to the specific DCS according to their interface specification. For the communication, a DCS can use different technologies, e.g. DCOM or web services.

The communication is bidirectional, i.e. a DCS can also send requests to an MES and receive replies. Again, the MSB adopts, translates, and routes the DCS requests to the DCS Adapter.

The usage of an MSB is not mandatory. For simple integration scenarios, it is sufficient to implement a specific JMS message handler to integrate a particular DCS.

Responsibility of the Components

Each component has its specific responsibility when an MES is connected with a DCS by means of the DCS Adapter.

MES client:

- Provides the parameters as needed by the DCS to execute a request.
- Calls the DCS Adapter.
- Evaluates the reply from the DCS and records the data as needed.
- Defines listeners that process requests coming from a DCS on a high level: they apply the corresponding business logic and return a result.

DCS Adapter:

- Provides an interface to a DCS that is independent of a specific DCS.
- Creates the requests and sends them via JMS.
- Waits for the defined reply and returns the results of the DCS to the MES client.
- Defines listeners that process requests coming from a DCS on a low level: they convert the requests to objects that can be handled by the MES, call the corresponding high-level listeners, and then send the result as a reply to the DCS via JMS.

Manufacturing Service Bus:

- Listens to the requests of all systems connected to it. The sender of a request could be the DCS Adapter or a DCS.
- Translates the received requests into the language of the receiver (a specific DCS or the DCS Adapter) and communicates with the receiver according to the corresponding interface.
- Translates the replies of the receiver into the defined reply message of the sender (a specific DCS or the DCS Adapter) and sends the replies.

Error Handling

The following types of errors can occur:

- JMS broker not available:
  In this case, not even the JMS request can be sent to the MSB.
- Request timeout, typically because the MSB is not available:
  If no component listens to the message sent by the DCS Adapter, the DCS

Adapter runs into a timeout.
This can also happen if the configured timeout is too short. That means the timeout is smaller than the time the DCS or the MES require to process the request.

- An error occurred in the DCS or the MES:
  In this case the MSB replies that the request has been rejected. The reply contains the corresponding error message.

A *DCSException* contains a localizable error message and a detailed error message:

- The error message contains general information about the issue.
- The detailed error message contains information about the root cause of the issue.

The component using the DSC Adapter (e.g. an MES client) must handle the exceptions. Usually it displays an error dialog with the error messages.
The DCS Adapter provides the *DCS<SpecificName>Exception* subclasses for each exception (e.g. *DCSCommunicationException*, *DCSJMSNotAvailableException*). Refer to the subclasses if the client shall distinguish between the different error cases.

Integration into an MES

The integration of the DCS Adapter into an MES requires several steps:

1. Install the DCS Adapter (page 9).
2. Set up the DCS Adapter configuration (page 10).
3. Make a message broker available (page 10).
4. Connect an MES client to a DCS (page 11).
5. Implement the high-level message listeners for all relevant requests coming from a DCS (page 11).
6. Configure the DCS Adapter (page 13).

We recommend to perform the following tasks before using the DCS Adapter in a productive environment:

- Simulate a DCS with a Java mock (page 14).
- Test the MES client without a DCS (page 15).

Prerequisites Checklist

Before you start the installation, check the prerequisites:

	Prerequisite	Your Notes	Done?
1	ActiveMQ is available in version 5.15.0 as required by the DCS Adapter.		

Installing the DCS Adapter

TIP

Starting with PharmaSuite 8.4, the DCS Adapter is included in PharmaSuite and will be installed along with PharmaSuite. However, a DCS Adapter-specific license is required. In case the DCS Adapter is not installed, proceed as described below.

The DCS Adapter is deployed with following JAR files:

- *mes-dcs-interface.jar*
- *mes-dcs-interface-xml.jar*

To install the DCS Adapter, proceed as follows:

1. Open Internet Explorer and navigate to the Rockwell Automation Download Site.
2. Navigate to the DCS Adapter and download the package.
3. On the Windows machine, expand the file that you have downloaded to extract the JAR files to a directory of your choice.
4. Add the JAR files to your development environment.
5. Make sure that the ActiveMQ JAR files are available in your development environment.
6. Extend the classpath accordingly.
7. Optional:
   For FactoryTalk ProductionCentre-based applications, add the JAR files as **Library** objects in Process Designer.

Setting up the DCS Adapter Configuration

The *mes-dcs-interface.jar* file provides properties files that can be used to configure and customize the DCS Adapter:

- *ServiceImplementations.properties* allows to assign a custom implementation class to services and bean interfaces used by the DCS Adapter.
- *log4j.properties* contains the log4j configuration of the DCS Adapter. You can change logging levels and other settings.
- *DCSConfig.properties* contains basic configuration properties of the DCS Adapter. Currently, the file contains only one property: **SystemName**. This is the name of the MES system used by the DCS Adapter. The name is used to automatically fill the sender field for all requests sent by the DCS Adapter and should be used by a DCS to fill the receiver field when sending requests to the MES. By default, **SystemName** is **MES**, but you can change it.
- *DCSAdapterMsgPack.properties* contains all error messages used by the DCS Adapter.

Making a Message Broker Available

In order to use JMS, a message broker is needed. The DCS Adapter supports the following message brokers:

- ActiveMQ broker of PharmaSuite
  You can use the message broker of your PharmaSuite installation. The broker is installed as an MS Windows service and named **PharmaSuite ActiveMQ**

Broker. For details, see "PharmaSuite Technical Manual Installation - Enterprise Edition" [A1] (page 41).

- ActiveMQ broker of FactoryTalk ProductionCentre
  You can start the broker of your FactoryTalk ProductionCentre installation. The broker is installed as an MS Windows service and named **ActiveMQ**. For details, see "FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced" [A2] (page 41).
- A separate ActiveMQ broker.

Connecting an MES Client to a DCS

The interface of the DCS Adapter is provided as a Java service with methods for each integration touchpoint (page 3) to the DCS. For details, see the Java documentation of a service.

The tasks of the MES client are:

- call the service methods and provide the parameters as needed,
- evaluate the reply to the service call and record the data as needed.

The *IDCSService* is responsible for the tasks. It will be instantiated by using a factory:

```
import com.rockwell.mes.dcs.ifc.IDCSService;
...
IDCSService service = com.rockwell.mes.dcs.ifc.DCSServiceFactory.getDCSService();
```

Example for the create DCS batch touchpoint

```
import com.rockwell.mes.dcs.ifc.IDCSService;
...
IDCSService service = com.rockwell.mes.dcs.ifc.DCSServiceFactory.getDCSService();

IDCSBatchCreationParameter params = new DCSBatchCreationParameter(“MyRecipeID”);
params.setCampaignID(“MyCampaignID”).setFormulaID(“MyFormulaID”) ...

IDCSAdapterConfiguration config = new DCSAdapterConfiguration(brokerURL, timeout);

service.createDCSBatch(config, “MyDCS”, “NewbatchID”, batchCreationParameter);
```

For details about the configuration, see "Configuring the DCS Adapter" (page 13).

Implementing Listeners

The DCS Adapter supports a bidirectional communication between a DCS and an MES: either an MES sends the request and a DCS replies or a DCS sends the request and an MES replies.

Please refer to "Connecting an MES Client to a DCS" (page 11) for details how to invoke an **MES sender** communication. To invoke a **DCS sender** communication, implement a listener for each relevant integration touchpoint to the DCS (page 3).

The DCS Adapter provides the low-level listeners for all integration touchpoints as abstract classes that extend the *AbstractDCSMessageListener* class. The low-level listeners convert the JMS message coming from a DCS to a bean object that holds all relevant information of the request. Additionally, the low-level listeners are generic in regards to the sent/received message types and the bean objects created by them. They are based on the requests and provide a single abstract method that should be implemented on the higher level:

Abstract method to be implemented on a higher level

```
/**
   * Process the request.
   *
   * @param sender the name of the DCS system that is sending the request
   * @param dcsObject the DCS object containing the request information
   * @param otherInformations the other informations that came with the request as a map
   * of key/value pairs. Supported Objects of the value are String, BigDecimal, Integer,
   * Calendar and Boolean
   * @return the response
   */
  protected abstract IDCSResponse processRequest(final String sender, final ReceivedType
            dcsObject, final Map<String, Object> otherInformations);
```

The returned response should wrap an **IResultContainer** object that contains the result required by this listener. Each listener contains a *createExtendedResult(Map<String, Object>)* method that can be used to create a result object of the correct type. For details, see the Java documentation of the listeners.

Example implementation of a listener that handles batch information requests coming from a DCS

```
public class MyGetBatchInformationMessageListener extends
             GetBatchInformationMessageListener {
  /**
   * @param dcsAdapterConfig the DCS adapter configuration
   */
  public MyGetBatchInformationMessageListener(IDCSAdapterConfiguration
         dcsAdapterConfig) {
    super(dcsAdapterConfig,
          DCSConfiguration.INSTANCE.getStringValue(DCSConfiguration.SYSTEM_NAME));
  }

  @Override
  protected IDCSResponse processRequest(final String sender, IBatchInformationParameter
            dcsObject, final Map<String, Object> otherInformations) {
    IDCSResponse<IExtendedBatchInformationResult> response =
                               newDCSResponse<IExtendedBatchInformationResult>();
    IBatchInformationResult specificResult =
                DCSServiceFactory.createBean(IBatchInformationResult.class);
    String batchID = dcsObject.getBatchID();
    Batch batch = PCContext.getFunctions().getBatchByName(batchID);
    if (batch == null) {
      response.setReplySuccessful(false);
      response.setReplyErrorMessage("Batch does not exist.");
      specificResult = null;
    } else {
     specificResult.setBatchID(batch.getName());
     specificResult.setMaterialID(batch.getPart());
     ... // set all other fields of the result here
     response.setReplySuccessful(true);
    }
    // if you want to send other information in the reply,
    // add it in the map of the extended result
    IExtendedBatchInformationResult result = createExtendedResult(specificResult,
                  Collections.EMPTY_MAP);
    response.setResult(result);
    return response;
  }
```

After implementing the listener, you must create an instance of it and start it using the *startReceiving()* method. For each request from a DCS, the *processRequest* method is invoked.

TIP

When you have finished the usage of the listener, do not forget to call its *shutdown()* method to clean up the resources that were used by the listener.

You may have noticed that the Java service of the DCS Adapter contains also methods that allow to send requests for the integration touchpoints where it is usually expected that a DCS sends requests to an MES. This is done for the following reasons: it allows you to test your listeners without using a real DCS and it allows an MES to simulate requests being sent by a DCS, e.g. to add additional consumptions

Configuring the DCS Adapter

The DCS Adapter expects its configuration as a parameter at each service method. The configuration is bundled in a class implementing the *IDCSAdapterConfiguration* interface. The following parameters are needed:

Parameter	Description	Example
MessageBrokerURL	URL of the message broker.	tcp://localhost:61646
ReplyTimeoutInSeconds	Time in seconds needed to send the request, process the request in the DCS or MES, and return the reply message.	10

Simulating a DCS with a Java Mock

To test the connection of an MES client to a DCS or to test requests that are normally sent by a DCS and processed by MES, a mock for the DCS can be used instead of a real DCS. The DCS Adapter provides a corresponding Java mock that can be started with a test form (see section "Testing the MES Client without a DCS" (page 15)).

To test the connection of an MES to a DCS, the mock acts as a DCS. The response of the mock is controlled by specific input of the parameters.

Request	Reply
Create DCS Batch	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.
Get DCS Alarms	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful ■ Filter criteria are implemented on a fixed set of alarms.
Get DCS Batch Values	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful. ■ For each requested batch report value, a value is replied. The type depends on the specification.
Set Order Context	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.

For testing the connection of a simulated DCS to the MES two options are available: either testing is performed with a real MES or testing with the DCS Adapter includes also the simulation of the MES.

The mock provides listeners that act as an MES. This is useful in case an MES is simulated as well.
To test a real MES, the listeners are not needed since the real MES provides the listeners. However, the listeners of the mock must be stopped.
The response of the mock listener (aka the simulated MES) is controlled by specific input of the parameters.

Request	Reply
Process Consumed Material	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.
Process Produced Material	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.

Request	Reply
Get Batch Information	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful. ■ The returned batch status depends on the *BatchID*: if it starts with *Released*, *Quarantined*, or *Blocked*, the corresponding status is returned. If it starts with anything else, *OTHER* is returned as batch status. The other batch information fields are filled with fixed values.

The mock can be parameterized with the following parameters:

Parameter	Description	Example
MessageBrokerURL	URL of the message broker.	tcp://localhost:61646
Receiver	The mock handles all requests sent to the given receiver/DCS.	JavaDCSMock

TIP

Running the Java mock within the **TestDCSAdapter** tool (page 15) uses the default value for the receiver.

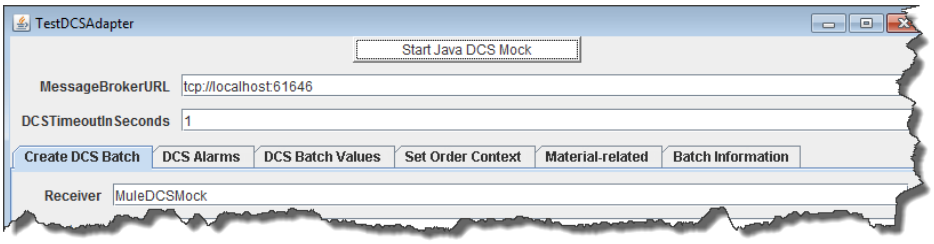
The listeners of the *Process Consumed Material*, *Process Produced Material*, and *Get Batch Information* requests use MES by default as a receiver name, because they are intended to simulate the high-level listeners implemented by an MES.

Testing the MES Client without a DCS

To test the DCS integration without a real DCS, the **TestDCSAdapter** tool provided for this purpose can be used.

To run the **TestDCSAdapter** tool, proceed as follows:

1. Navigate to the *mes-dcs-interface.jar* file.
2. Start the *com.rockwell.mes.dcs.impl.ui.TestDCSAdapter* Java main as an **Application**.



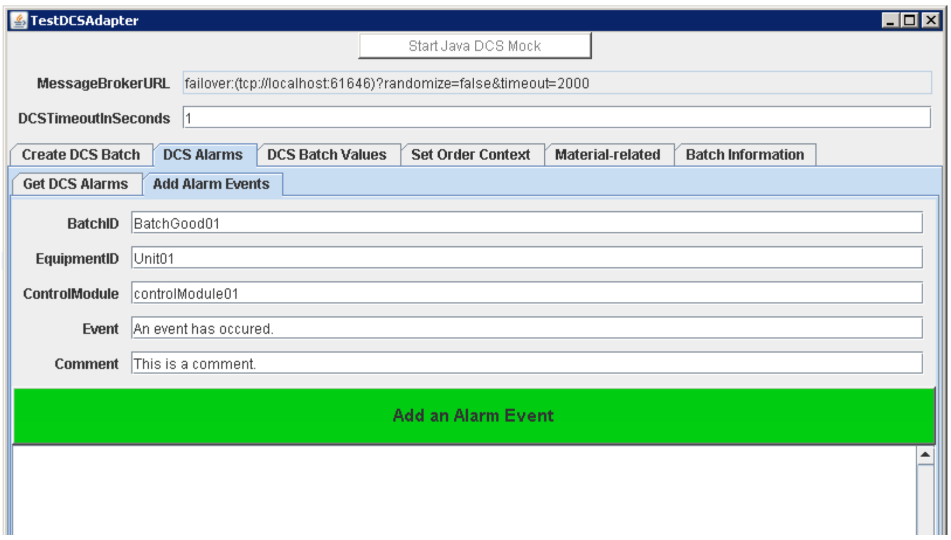
MessageBrokerURL tcp://localhost:61646

DCSTimeoutInSeconds 1

Receiver MuleDCSMock

Figure 2: TestDCSAdapter tool

3. The upper part of the form contains the configuration of the DCS Adapter. Here you can configure the message broker URL and start the Java DCS simulation mock (page 14).
   The mock handles messages sent to the **JavaDCSMock** receiver. If your MES client uses another broker URL, adapt it here before you start the mock.
4. To start the mock, click the **Start Java DCS Mock** button. After the mock has been started, it can send replies to the MES client.
   - In this context only, the *Add Alarm Events* tab is useful. Here you can dynamically add alarm events to the running mock on the fly. To define the alarm events, use the input boxes. The alarm event will get the current timestamp. The next time a request to get the alarm events is replied to, the newly added alarm events are included (depending on the configured batch, unit, modules).

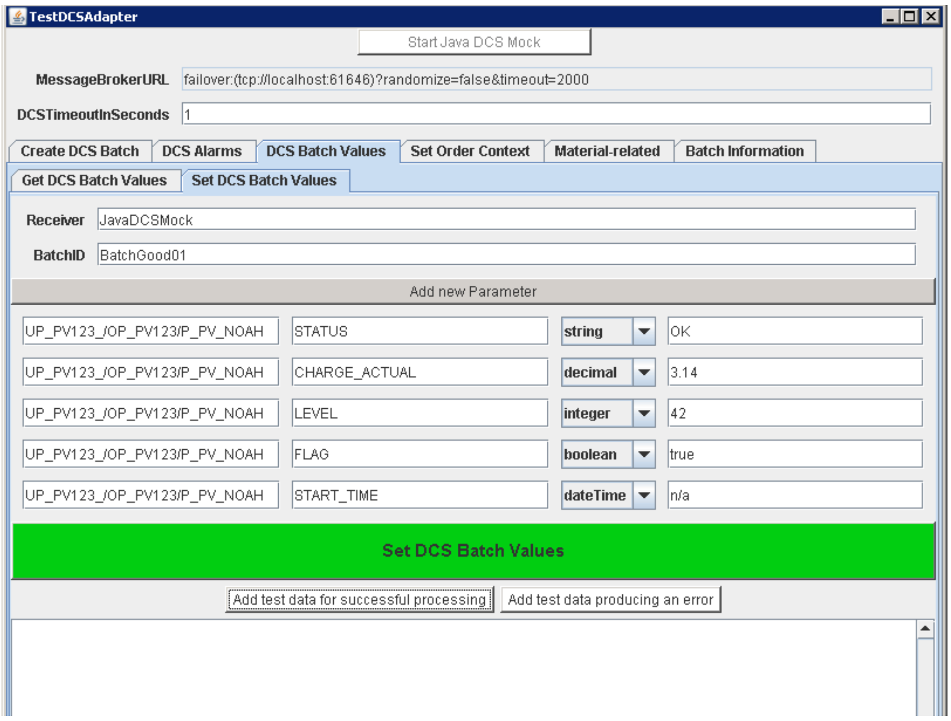


MessageBrokerURL: failover:(tcp://localhost:61646)?randomize=false&timeout=2000
DCSTimeoutInSeconds: 1

BatchID	BatchGood01
EquipmentID	Unit01
ControlModule	controlModule01
Event	An event has occured.
Comment	This is a comment.

Figure 3: Adding alarm events to the mock

- In the *Set DCS Batch Values* tab, you can dynamically add batch values which can be retrieved later by the *Get DCS Batch Values* tab or any phase building block of PharmaSuite that reads values from a DCS.
  The **dateTime** type is not supported. If you request a **dateTime** value in the *Get DCS Batch Values* tab, the current time is reported.



MessageBrokerURL failover:(tcp://localhost:61646)?randomize=false&timeout=2000

DCSTimeoutInSeconds 1

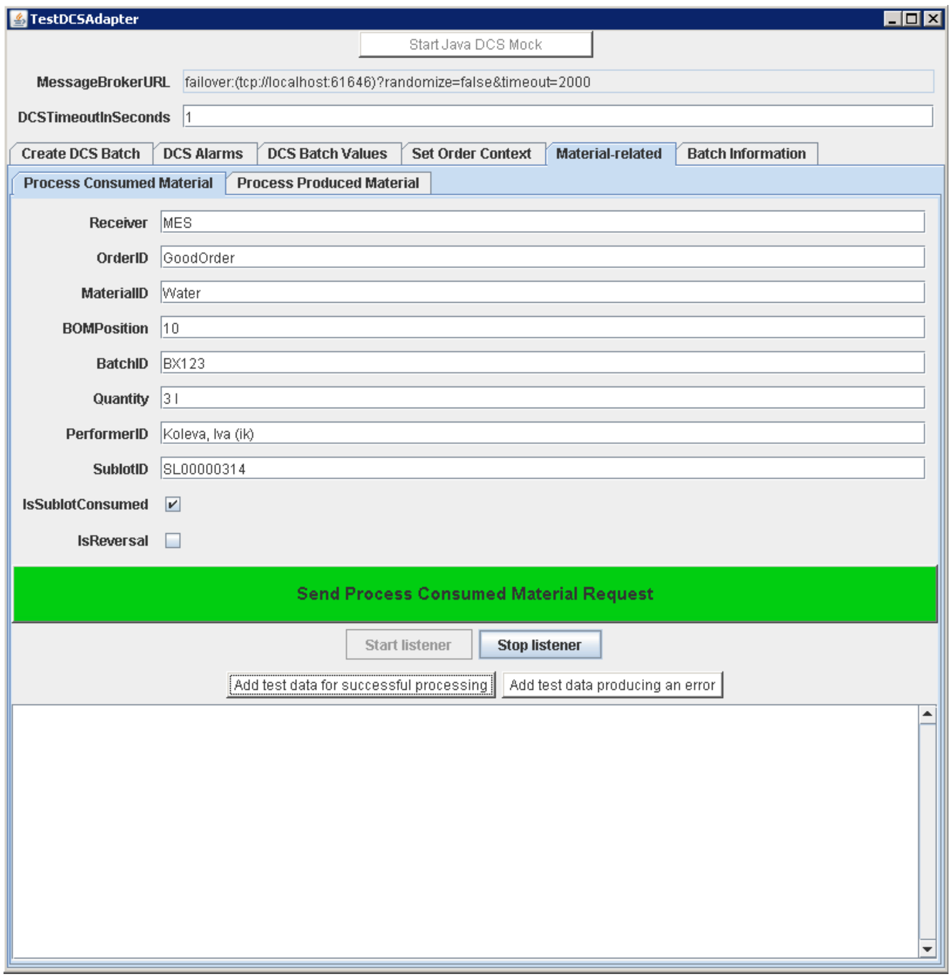
Receiver JavaDCSMock

BatchID BatchGood01

UP_PV123_/OP_PV123/P_PV_NOAH	STATUS	string	OK
UP_PV123_/OP_PV123/P_PV_NOAH	CHARGE_ACTUAL	decimal	3.14
UP_PV123_/OP_PV123/P_PV_NOAH	LEVEL	integer	42
UP_PV123_/OP_PV123/P_PV_NOAH	FLAG	boolean	true
UP_PV123_/OP_PV123/P_PV_NOAH	START_TIME	dateTime	n/a

Figure 4: Adding batch values to the mock

5. You can use the *Batch Information* tab and the two sub-tabs of the *Material-related* tab to test the DCS integration (see "Testing the DCS Integration without an MES Client" (page 38)) or to test the high-level listeners implemented by an MES. By default, a listener mock is started for each of the tabs. This allows you to test the DCS integration. However, if you wish to test a high-level listener, you can stop the mock listener with the **Stop listener** button in the corresponding tab. Then all messages are processed by the running MES listener.
   In both tabs, requests are sent that would normally be sent by a DCS. This is different from the other tabs where requests are sent that are normally sent by an MES.


TestDCSAdapter
Start Java DCS Mock
Create DCS Batch
DCS Alarms
DCS Batch Values
Set Order Context
Material-related
Batch Information
Process Consumed Material
Process Produced Material
Send Process Consumed Material Request
Start listener
Stop listener
Add test data for successful processing
Add test data producing an error

MessageBrokerURL	failover:(tcp://localhost:61646)?randomize=false&timeout=2000
DCSTimeoutInSeconds	1
Receiver	MES
OrderID	GoodOrder
MaterialID	Water
BOMPosition	10
BatchID	BX123
Quantity	3 l
PerformerID	Koleva, Iva (ik)
SublotID	SL00000314
IsSublotConsumed	✓
IsReversal	

Figure 5: Stopping and starting listener mocks

6. The other tabs of the form are mainly intended for testing the DCS integration and to simulate an MES (see section "Testing the DCS Integration without an MES Client" (page 38)).

Chapter 4

Integration of a DCS

To integrate a DCS, a component must be created that listens to the DCS requests of the MES for the given DCS and translates it into the language of the specific DCS. For integrating several DCSs, it makes sense to create an MSB that communicates with multiple DCSs according to their technology.
For simple integration scenarios, you can also implement a specific JMS message handler to integrate a specific DCS.

For details of the tasks of an MSB, see section "Responsibility of the Components" (page 6).

The integration of a DCS makes use of:

- Channels for integration touchpoints (page 20)
- XML schema of the interface (page 20)
- Touchpoint-specific requests:
  - ProcessCreateDCSBatch (page 21)
  - GetDCSAlarmEvents (page 24)
  - GetDCSBatchValues (page 26)
  - ProcessOrderContext (page 28)
  - ProcessConsumedMaterial (page 30)
  - ProcessProducedMaterial (page 32)
  - GetBatchInformation (page 33)
- Touchpoint-specific replies
  - AcknowledgeCreateDCSBatch (page 23)
  - ShowDCSAlarmEvents (page 24)
  - ShowDCSBatchValues (page 27)
  - AcknowledgeOrderContext (page 29)
  - AcknowledgeConsumedMaterial (page 31)
  - AcknowledgeProducedMaterial (page 32)
  - ShowBatchInformation (page 34)

The example of an MSB implementation based on EIHub (page 36) describes how to structure such an integration flow.

We recommend to perform the following task before using the DCS Adapter in a productive environment:

- Test the DCS integration without an MES client (page 38).

Channels for Integration Touchpoints

For each DCS, a set of four channels is used, i.e. one channel per integration touch point:

Integration touchpoint	Channel
Create DCS Batch	DCSRequest_<DCSTargetSystem>_CreateDCSBatch
Get DCS Alarms	DCSRequest_<DCSTargetSystem>_GetDCSAlarmEvents
Get DCS Batch Values	DCSRequest_<DCSTargetSystem>_GetDCSBatchValues
Set Order Context	DCSRequest_<DCSTargetSystem>_ProcessOrderContext

Additionally, three channels are used for the messages received by an MES:

Integration touchpoint	Channel
Process Consumed Material	DCSRequest_<MESSystemName>_ProcessConsumedMaterial
Process Produced Material	DCSRequest_< MESSystemName>_ProcessProducedMaterial
Get Batch Information	DCSRequest_<DCSTargetSystem>_GetBatchInformation

The common pattern of a channel is: DCSRequest_<LogicalDCSName>_<RequestType>

XML Schema of the Interface

The interface between the DCS Adapter and the DCS or MSB is an XML document sent via JMS.

The corresponding XML schema is defined in *mes-dcs-interface.xsd* and provided with the *mes-dcs-interface-xml.jar*. The schema is compliant to the S88/S95 standards and based on the B2MML standard schema.

For each integration touchpoint, the schema definition contains two top-level XML elements, one for the request and one for the reply:

Top-level XML element for requests	Top-level XML element for replies
ProcessCreateDCSBatch (page 21)	AcknowledgeCreateDCSBatch (page 23)
GetDCSAlarmEvents (page 24)	ShowDCSAlarmEvents (page 24)
GetDCSBatchValues (page 26)	ShowDCSBatchValues (page 27)
ProcessOrderContext (page 28)	AcknowledgeOrderContext (page 29)
ProcessConsumedMaterial (page 30)	AcknowledgeConsumedMaterial (page 31)
ProcessProducedMaterial (page 32)	AcknowledgeProducedMaterial (page 32)
GetBatchInformation (page 33)	ShowBatchInformation (page 34)

The DCS Adapter sends an XML request of a specific document type (e.g. **ProcessCreateDCSBatch**) and expects a reply of the corresponding document type (e.g. **AcknowledgeCreateDCSBatch**).

The messages in both directions are text messages containing XML-formatted strings.

TIP

To validate the sample XML documents listed for the requests and replies, move the files to the ../../../*main/xsd/mes-dcs-interface.xsd* subdirectory.

General Structure of the XML Requests and Replies

In general each XML request and reply consists of three containers:

- **ApplicationArea** contains the sender and the receiver of a message.
- **DataArea**
  - For requests, it contains the parameters of the request.
  - For replies, it contains the information whether the request has been accepted or rejected and an error message in the latter case.
- **OtherInformations** is a container that can be used to transfer additional data that is needed and not contained in the standard. Optionally, it contains a list of additional key/value pairs (see section "Extending the Standard" (page 39)).

Request: ProcessCreateDCSBatch

The request for creation of a new DCS batch is an XML document of the **ProcessCreateDCSBatch** type. The parameters are located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **ControlRecipe/Parameters** contains the list of any additional parameters.

- **ControlRecipeEquipmentRequirements** contains the list of the required units.

Example of a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessCreateDCSBatch xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>Building_API_220</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BS101338_27Oct15_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
    <ControlRecipe>
      <Parameters>
        <Parameter>
          <ID>R_BUF_VOL</ID>
          <ValueNumeric>765</ValueNumeric>
        </Parameter>
        <Parameter>
          <ID>ENCODER</ID>
          <ValueString>04PV51400ENC1</ValueString>
        </Parameter>
        <Parameter>
          <ID>R_INT_WIFI_RINSE</ID>
          <ValueBoolean>true</ValueBoolean>
        </Parameter>
      </Parameters>
      <EquipmentRequirements>
        <EquipmentRequirement>
          <Constraint>UnitProcA</Constraint>
          <ID>PV3553</ID>
        </EquipmentRequirement>
        <EquipmentRequirement>
          <Constraint>EqmClassX</Constraint>
          <ID>PV4253</ID>
        </EquipmentRequirement>
      </EquipmentRequirements>
    </ControlRecipe>

    <!-- all optional fields filled -->
    <CampaignID>DeltaV_V11_CAMP</CampaignID>
    <ScaledSize>100.00</ScaledSize>
    <FormulaID>SFPHC_PV515400_MAKEUP</FormulaID>
    <Description>This is a long description</Description>
  </DataArea>
</ProcessCreateDCSBatch>
```

Reply: AcknowledgeCreateDCSBatch

The reply to the **ProcessCreateDCSBatch** request is an XML document of the **AcknowledgeCreateDCSBatch** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/InternalBatchID** contains the unique internal ID of the newly created batch on the DCS.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch xsi:schemaLocation="
       http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
    <InternalBatchID>batchID_20160819_104003</InternalBatchID>
  </DataArea>

</AcknowledgeCreateDCSBatch>
```

Example of a Rejected reply to a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected">
      Error during batch creation. Batch ID already exists.
    </ResponseCriteria>
    <InternalBatchID></InternalBatchID>
  </DataArea>

</AcknowledgeCreateDCSBatch>
```

Request: GetDCSAlarmEvents

The request for getting alarm events from a DCS is an XML document of the **GetDCSAlarmEvents** type. The parameters are located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/EquipmentID** contains the unit.
- **DataArea/RecipeElement** contains the list of control modules.

Example of a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetDCSAlarmEvents xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <!-- Maximal filter criteria: use all fields -->
      <BatchID>BX01815-20160114_142303_unitA</BatchID>
    <EquipmentID>VesselA</EquipmentID>
      <RecipeElement>
        <ActualEquipmentID>controlModuleA</ActualEquipmentID>
        <ActualEquipmentID>controlModuleB</ActualEquipmentID>
        <ActualEquipmentID>controlModuleC</ActualEquipmentID>
      </RecipeElement>
    <StartTime>2016-01-14T13:45:00</StartTime>
    <EndTime>2016-01-14T17:45:00</EndTime>
  </DataArea>

</GetDCSAlarmEvents>
```

Reply: ShowDCSAlarmEvents

The reply to the **GetDCSAlarmsEvents** request is an XML document of the **ShowDCSAlarmEvents** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/Events** contains the returned list of alarm events if **actionCode** contains **Accepted**. The list may be empty.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSAlarmEvents xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
    <Events>
      <AlarmEvent>
        <TimeStamp>2016-01-14T13:45:34</TimeStamp>
        <Value>Errorcode:3;Critical;Upper limit violation</Value>
        <EquipmentID>VesselA/TempSensor</EquipmentID>
        <MessageText>An arbitrary comment.</MessageText>
      </AlarmEvent>
      <AlarmEvent>
        <TimeStamp>2016-01-14T14:44:14</TimeStamp>
        <Value>Errorcode:4;Critical;Lower limit violation</Value>
        <EquipmentID>VesselA/TempSensor</EquipmentID>
      </AlarmEvent>
    </Events>
  </DataArea>

</ShowDCSAlarmEvents>
```

Example of a Rejected reply to a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSAlarmEvents xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
    <Events>       </Events>
  </DataArea>

</ShowDCSAlarmEvents>
```

Request: GetDCSBatchValues

The request for getting batch values from a DCS is an XML document of the **GetDCSABatchValues** type. The parameters are located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/ControlRecipe** contains the list of requested batch values:
  - **RecipeElement/ID** contains the path of the batch value.
  - **RecipeElement/ParameterID** contains the name of the batch value.
  - **RecipeElement/DataType** contains the requested data type of the batch value.

Example of a 'get DCS batch values' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetDCSBatchValues xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BX01815-20160114_142303_unitA</BatchID>

    <ControlRecipe>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>STATUS</ParameterID>
        <DataType>string</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>FLAG</ParameterID>
        <DataType>boolean</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>CHARGE_ACTUAL</ParameterID>
        <DataType>decimal</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>START_TIME</ParameterID>
        <DataType>dateTime</DataType>
      </RecipeElement>
    </ControlRecipe>
  </DataArea>

</GetDCSBatchValues>
```

Reply: ShowDCSBatchValues

The reply to the **GetDCSBatchValues** request is an XML document of the **ShowDCSBatchValues** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ControlRecipe** contains the list of returned batch values:
  - **RecipeElement/ID** contains the path of the batch value.
  - **RecipeElement/ParameterID** contains the name of the batch value.
  - **RecipeElement/Value<data type>** contains the returned batch value of the requested data type.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'get DCS batch values' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSBatchValues xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>

    <ControlRecipe>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>STATUS</ParameterID>
        <ValueString>started</ValueString>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>START_TIME</ParameterID>
        <ValueDatetime>2016-01-14T14:54:43</ValueDatetime>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>FLAG</ParameterID>
        <ValueBoolean>true</ValueBoolean>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>LEVEL</ParameterID>
        <ValueInteger>12</ValueInteger>
      </RecipeElement>

    </ControlRecipe>
  </DataArea>

</ShowDCSBatchValues>
```

Request: ProcessOrderContext

The request for setting an order context in a DCS is an XML document of the **ProcessOrderContext** type. The order information is located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/MaterialParameters** contains the list of material parameters for this order.

Example of a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessOrderContext xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>
  <DataArea>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <BatchID>BS101338_10Jan16_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
    <MasterRecipeVersion>1</MasterRecipeVersion>
    <PhaseID>Set Order Context (RS) [1.0]</PhaseID>
    <MaterialParameters>
      <Parameter>
        <MaterialDefinitionID>Water</MaterialDefinitionID>
        <MaterialPositionID>10</MaterialPositionID>
        <ParameterType>ProcessInput</ParameterType>
        <Value>
          <Quantity>42</Quantity>
          <UnitOfMeasure>l</UnitOfMeasure>
        </Value>
        <AllocatedBatchID>BX01</AllocatedBatchID>
        <AllocatedBatchID>BX185</AllocatedBatchID>
        <OrderStepID>Dispense</OrderStepID>
      </Parameter>
      <Parameter>
```

```
        <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
        <MaterialPositionID>20</MaterialPositionID>
        <ParameterType>ProcessOutput</ParameterType>
        <Value>
          <Quantity>10</Quantity>
          <UnitOfMeasure>kg</UnitOfMeasure>
        </Value>
        <OrderStepID>Packaging</OrderStepID>
      </Parameter>
    </MaterialParameters>
  </DataArea>
</ProcessOrderContext>
```

Reply: AcknowledgeOrderContext

The reply to the **ProcessOrderContext** request is an XML document of the **AcknowledgeOrderContext** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeOrderContext xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeOrderContext>
```

Example of a Rejected reply to a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeOrderContext xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeOrderContext >
```

Request: ProcessConsumedMaterial

The request for processing consumed material in an MES is an XML document of the **ProcessConsumedMaterial** type. The information of the consumed material is located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/doFixMistake** indicates whether the message is related to a previously sent request that was sent by mistake. This field is intended for usage from MES (as part of an error handling mechanism). Any request coming from the automation system that has this field set to **true** will be rejected by MES. If the field is empty, **false** is used by default (i.e. straightforward consumption).
- **DataArea/ConsumeMaterialEvent** contains additional information of the consumed material.

Example of a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessConsumedMaterial xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <doFixMistake>false</doFixMistake>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Water</MaterialDefinitionID>
    <MaterialPositionID>10</MaterialPositionID>
    <BatchID>BX123</BatchID>
    <ConsumeMaterialEvent>
      <TimeStamp>2017-08-15T13:37:41.157+02:00</TimeStamp>
      <Value>
        <Quantity>3</Quantity>
        <UnitOfMeasure>l</UnitOfMeasure>
      </Value>
      <SublotID>SL00000314</SublotID>
      <SublotIsEmpty>true</SublotIsEmpty>
      <PersonID>Gerr, Detamana (dg)</PersonID>
    </ConsumeMaterialEvent>
  </DataArea>
</ProcessConsumedMaterial>
```

Reply: AcknowledgeConsumedMaterial

The reply to the **ProcessConsumedMaterial** request is an XML document of the **AcknowledgeConsumedMaterial** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**. The error messages supported by MES are located in the *DCSAdapterMsgPack.properties* file.

Example of an Accepted reply to a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeConsumedMaterial xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeConsumedMaterial>
```

Example of a Rejected reply to a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeConsumedMaterial xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      Could not find a matching batch.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeConsumedMaterial>
```

Request: ProcessProducedMaterial

The request for processing produced material in an MES is an XML document of the **ProcessProducedMaterial** type. The information of the produced material is located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/ProduceMaterialEvent** contains additional information of the produced material.

Example of a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessProducedMaterial xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <MaterialPositionID>110</MaterialPositionID>
    <BatchID>BX123</BatchID>
    <ProduceMaterialEvent>
      <TimeStamp>2017-08-15T13:48:18.807+02:00</TimeStamp>
      <Value>
        <Quantity>10</Quantity>
        <UnitOfMeasure>kg</UnitOfMeasure>
      </Value>
      <SublotID>SL00000315</SublotID>
      <PersonID>Gerr, Detamana (dg)</PersonID>
    </ProduceMaterialEvent>
  </DataArea>
</ProcessProducedMaterial>
```

Reply: AcknowledgeProducedMaterial

The reply to the **ProcessProducedMaterial** request is an XML document of the **AcknowledgeProducedMaterial** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.

- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**. The error messages supported by MES are located in the *DCSAdapterMsgPack.properties* file.

Example of an Accepted reply to a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeProducedMaterial xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeProducedMaterial>
```

Example of a Rejected reply to a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeProducedMaterial xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      The corresponding order is not running.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeProducedMaterial>
```

Request: GetBatchInformation

The request for retrieving batch information from an MES is an XML document of the **GetBatchInformation** type. The batch information is located in the **DataArea** container (page 21):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.

Example of a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetBatchInformation xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <BatchID>ReleasedBatch</BatchID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
 </DataArea>
</GetBatchInformation>
```

Reply: ShowBatchInformation

The reply to the **GetBatchInformation** request is an XML document of the **ShowBatchInformation** type. The results are located in the **DataArea** container (page 21):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.
- **DataArea/BatchStatus** contains an enumeration value representing the batch status.

Example of an Accepted reply to a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowBatchInformation xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"/>
    <BatchID>ReleasedBatch</BatchID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <ifc:BatchStatus>Released</ifc:BatchStatus>
    <BatchQuantity>
      <Quantity>1000.0</Quantity>
      <UnitOfMeasure>kg</UnitOfMeasure>
    </BatchQuantity>
    <Potency>
      <ifc:Quantity>87.00</ifc:Quantity>
      <ifc:UnitOfMeasure>%</ifc:UnitOfMeasure>
    </Potency>
    <ProductionDate>2017-08-14T14:10:12.666+02:00</ProductionDate>
    <ExpiryDate>2018-08-15T14:10:12.666+02:00</ExpiryDate>
    <RetestDate>2017-09-15T14:10:12.666+02:00</RetestDate>
  </DataArea>

</ShowBatchInformation>
```

Example of a Rejected reply to a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowBatchInformation xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
  </DataArea>

</ShowBatchInformation>
```

Example of an MSB Implementation Based on EIHub

This example describes how to structure an integration flow in the Manufacturing Service Bus. The example is based on Enterprise Integration Hub (EIHub). However, the overall approach is similar for other MSBs/ESBs.

The figure below illustrates the flow for integrating the PlantPAx® DCS, which provides a RESTful web service interface for batch creation. The flow only handles the **Create DCS Batch** integration touchpoint. The other touchpoints are not included. For including them, a similar structure may be needed.

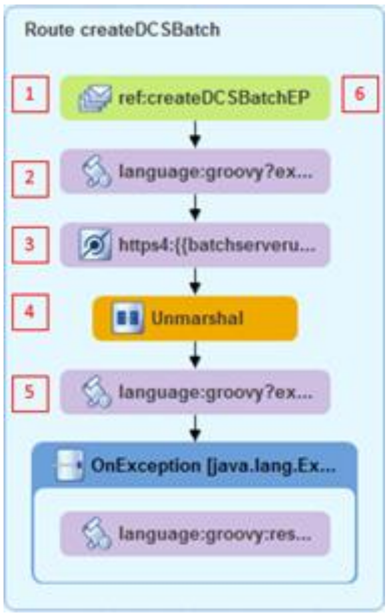


Figure 6: Example of an MSB flow

The flow consists of the following steps:

1. The JMS endpoint accepts the request message from the DCS Adapter.
   The message contains the XML structure as described in "General Structure of the XML Requests and Replies" (page 21).
2. The data transformation script converts this XML document into a JSON document that complies with the JSON data structure for the PlantPAx RESTful service.
3. The RESTful web service offered by the PlantPAx DCS is called.
4. The RESTful web service returns the result of the action in the DCS.
5. The vendor-specific result structure is converted back into the XML schema expected by the DCS Adapter for the reply, by means of an appropriate script.
6. The JMS reply is returned to the DCS Adapter

The figure below illustrates an alternative approach with a mock flow. For simulation purposes, you can also set up a mock flow in the MSB without a real connection to a DCS. In this case, you only log the message received from the DCS Adapter and perform a simple conversion into the reply XML structure.

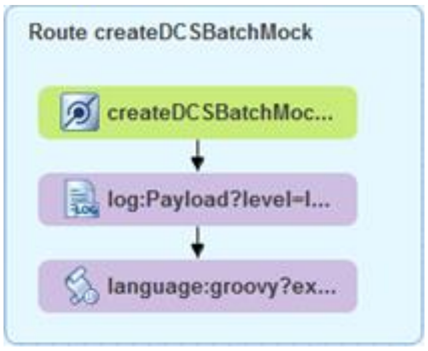
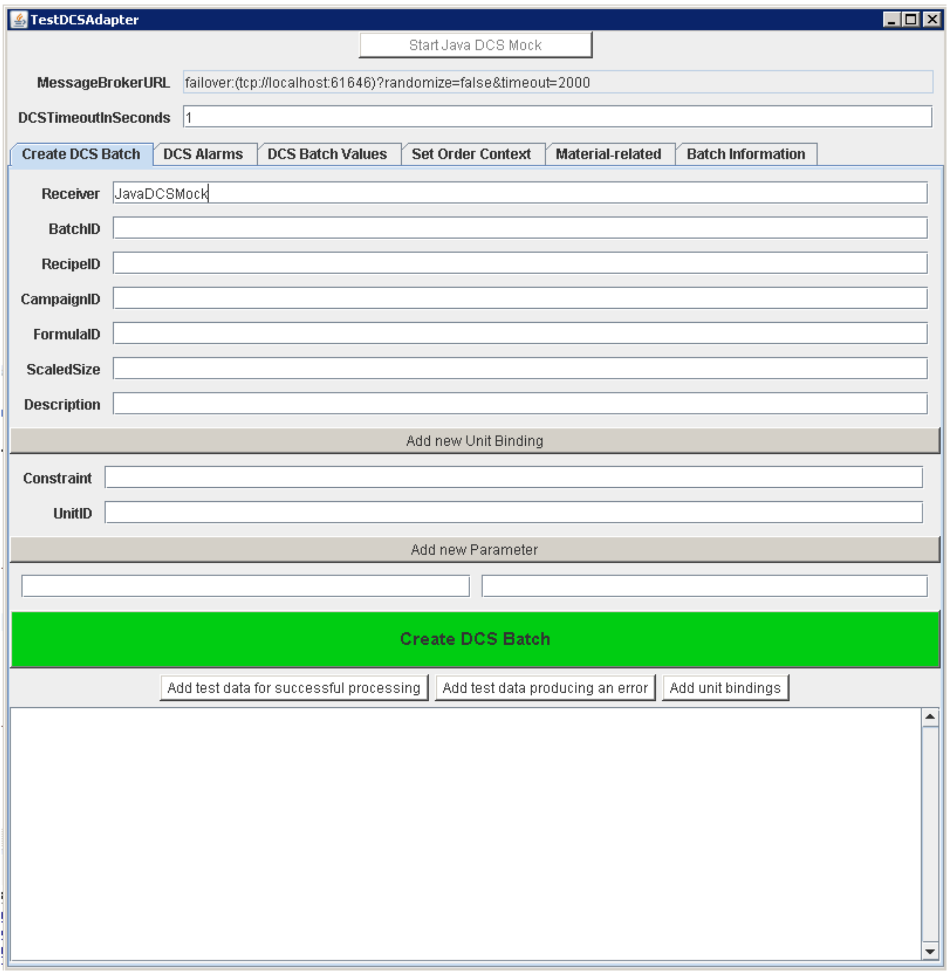


Figure 7: Example of an MSB mock flow

Testing the DCS Integration without an MES Client

To test the DCS integration without an MES client, you can use the **TestDCSAdapter** tool provided for this purpose (see section "Testing the MES Client without a DCS" (page 15)).

The **TestDCSAdapter** tool contains tabs for each integration touchpoint. The *Process Consumed Material* and *Process Produced Material* tabs are combined in the *Material-related* tab. In each tab, you can provide the data needed to build a request. The touchpoint-specific **Process** button will trigger the DCS Adapter to send the corresponding request to the given receiver. Thus you trigger a request message to be sent to your DCS.



MessageBrokerURL	failover:(tcp://localhost:61646)?randomize=false&timeout=2000
DCSTimeoutInSeconds	1
Receiver	JavaDCSMock
BatchID	
RecipeID	
CampaignID	
FormulaID	
ScaledSize	
Description	
Constraint	
UnitID	

Figure 8: Create DCS batch tab of the TestDCSAdapter tool

The upper part of the form contains the configuration of the DCS Adapter. Here you can configure the message broker URL.
In the **Receiver** box, type the name of your DCS. In the additional boxes, you can enter further test data.
When you have completed the data for your request, click the touchpoint-specific **Process** button to trigger the DCS Adapter to send the corresponding request.

Extending the Standard

If there is a need to transfer further data that is not contained in the standard, it is possible to add a list of key/value pairs to the XML document for both directions. The *IExtendedDCSService* interface provides methods for each integration touchpoint. Each method has an additional parameter for a map of keys and values. Furthermore, the return value of these methods returns a map that contains the key/value pairs included in the reply message.

In the XML document, the additional data is located in the **OtherInformations** container (page 21).

Example of a 'create process XML document with OtherInformations'

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessCreateDCSBatch xsi:schemaLocation=
        "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>Building_API_220</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BS101338_27Oct15_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
      <ControlRecipe>
        <Parameters> </Parameters>
        <EquipmentRequirements> </EquipmentRequirements>
      </ControlRecipe>
      <CampaignID>DeltaV_V11_CAMP</CampaignID>
      <ScaledSize>100.00</ScaledSize>
      <FormulaID>SFPHC_PV515400_MAKEUP</FormulaID>
    <Description>This is a long description</Description>
  </DataArea>

  <OtherInformations>
    <OtherInformation>
      <ID>AdditionalData01</ID>
      <ValueString>AValue</ValueString>
    </OtherInformation>
      <OtherInformation>
      <ID>AdditionalData02</ID>
      <ValueDatetime>2016-01-14T14:54:43</ValueDatetime>
    </OtherInformation>

  </OtherInformations>
</ProcessCreateDCSBatch>
```

Example of a reply message with 'OtherInformations'

```xml
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch xsi:schemaLocation=
       "http://www.rockwell.com/mes/dcs/ifc ../../../main/xsd/mes-dcs-interface.xsd"
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="1.0">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected">
      Error during batch creation. Batch ID already exists.
    </ResponseCriteria>
  </DataArea>

  <OtherInformations>
    <OtherInformation>
      <ID>Custom field: DeltaV Internal Id</ID>
      <ValueString>DV_323433232</ValueString>
    </OtherInformation>
  </OtherInformations>

</AcknowledgeCreateDCSBatch>
```

Extending Standard Implementation of Listeners in an MES Client

To answer and handle requests sent by a DCS to an MES Client, an MES must implement the correspondent listeners. For details, see "Implementing Listeners" (page 11).

The DCS Adapter provides the possibility to transfer further data in a request. To use the additional data in the implementation of the MES, the MES must be customized. For details, please refer to the documentation of the respective MES client. For PharmaSuite, see "Adapting Material Handling for DCS" in Volume 3 of the "Technical Manual Configuration and Extension" [A3] (page 41).

Reference Documents

The following documents are available from the Rockwell Automation Download Site.

No.	Document Title	Part Number
A1	PharmaSuite Technical Manual Installation - Enterprise Edition	PSEN-IN008E-EN-E
A2	FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced	PCJBAD IN104A EN E
A3	PharmaSuite Technical Manual Configuration & Extension - Volume 3	PSCEV3-GR008E-EN-E

TIP

To access the Rockwell Automation Download Site, you need to acquire a user account from Rockwell Automation Sales or Support.

Appendix A

Revision History

The following table describes the history of this document.

Changes related to the document:

Object	Description	Document
---	---	---

Changes related to "Introduction" (page 1):

Object	Description	Document
Introduction (page 1)	Requests can be sent between the MES and any DCS.	1.0
Integration Touchpoints (page 3)	Additional touchpoints are supported: Set Order Context, Process Consumed Material, Process Produced Material, Get Batch Information.	1.0

Changes related to "Architecture of an MES Integrated with the DCS Adapter" (page 5):

Object	Description	Document
Architecture of an MES Integrated with the DCS Adapter (page 5)	The communication is bidirectional, i.e. a DCS can also send requests to an MES and receive replies.	1.0
Responsibility of the Components (page 6)	MES client: Defines listeners that process requests coming from a DCS on a high level. DCS Adapter: Defines listeners that process requests coming from a DCS on a low level. Manufacturing Service Bus: Listens to the requests of all systems connected to it. The sender of a request could be the DCS Adapter or a DCS.	1.0
Error Handling (page 6)	Types of errors updated.	1.0

Changes related to "Integration into an MES" (page 9):

Object	Description	Document
Prerequisites Checklist (page 9)	ActiveMQ version updated to 5.15.0.	1.0
Installing the DCS Adapter (page 9)	Starting with PharmaSuite 8.4, the DCS Adapter is included in PharmaSuite and will be installed along with PharmaSuite. However, a DCS Adapter-specific license is required.	1.0
Setting up the DCS Adapter Configuration (page 10)	New section.	1.0
Making a Message Broker Available (page 10)	With PharmaSuite 8.4, the DCS Adapter is installed along with PharmaSuite. Thus, the ActiveMQ broker of PharmaSuite or FactoryTalk ProductionCentre are available anyhow.	1.0
Implementing Listeners (page 11)	New section.	1.0
Configuring the DCS Adapter (page 13)	Description of *ReplyTimeoutInSeconds* parameter updated.	1.0
Simulating a DCS with a Java Mock (page 14)	Additional requests are supported: Set Order Context, Process Consumed Material, Process Produced Material, Get Batch Information.	1.0
Testing the MES Client without a DCS (page 15)	Step 5, *Batch Information* tab and *Material-related* tab added.	1.0

Changes related to "Integration of a DCS" (page 19):

Object	Description	Document
Integration of a DCS (page 19)	Additional touchpoint-specific requests: ProcessOrderContext, ProcessConsumedMaterial, ProcessProducedMaterial, GetBatchInformation. Additional touchpoint-specific replies: AcknowledgeOrderContext, AcknowledgeConsumedMaterial, AcknowledgeProducedMaterial, ShowBatchInformation.	1.0
Channels for Integration Touchpoints (page 20)	Additional integration touchpoint: Set Order Context, Process Consumed Material, Process Produced Material, Get Batch Information.	1.0
XML Schema of the Interface (page 20)	Additional top-level XML elements: ProcessOrderContext, AcknowledgeProcessOrderContext, ProcessConsumedMaterial, AcknowledgeProcessConsumedMaterial, ProcessProducedMaterial, AcknowledgeProcessProducedMaterial, GetBatchInformation, ShowBatchInformation.	1.0

Object	Description	Document
Request: ProcessOrderContext (page 28)	New request.	1.0
Reply: AcknowledgeOrderContext (page 29)	New reply.	1.0
Request: ProcessConsumedMaterial (page 30)	New request.	1.0
Reply: AcknowledgeConsumedMaterial (page 31)	New reply.	1.0
Request: ProcessProducedMaterial (page 32)	New request.	1.0
Reply: AcknowledgeProducedMaterial (page 32)	New reply.	1.0
Request: GetBatchInformation (page 33)	New request.	1.0
Reply: ShowBatchInformation (page 34)	New reply.	1.0
Example of an MSB Implementation Based on EIHub (page 36)	The example of an MSB implementation is based on Enterprise Integration Hub (EIHub). The flow only handles the **Create DCS Batch** integration touchpoint. The other touchpoints are included.	1.0
Testing the DCS Integration without an MES Client (page 38)	The **TestDCSAdapter** tool contains tabs for each integration touchpoint. The *Process Consumed Material* and *Process Produced Material* tabs are combined in the *Material-related* tab.	1.0

Changes related to "Extending the Standard" (page 39):

Object	Description	Document
Extending Standard Implementation of Listeners in an MES Client (page 40)	New section.	1.0

Index

A

Abbreviation • 2
AcknowledgeConsumedMaterial reply • 31
AcknowledgeCreateDCSBatch reply • 23
AcknowledgeOrderContext reply • 29
AcknowledgeProducedMaterial reply • 32
Architecture • 5
Audience • 1

C

Channels for integration touchpoints • 20
Configuration of DCS Adapter • 13
Connecting an MES client to a DCS • 11
Create a DCS batch • 3

D

DCS Adapter (configuration setup) • 10
DCS Adapter (configuration) • 13
DCS Adapter (installation) • 9
DCS Adapter (responsibility) • 6
Download Site • 41

E

Error handling • 6
Extending the standard • 39

G

Get DCS alarms • 3
Get DCS batch values • 3
GetBatchInformation request • 33
GetDCSAlarmEvents request • 24
GetDCSBatchValues request • 26

I

Installation of DCS Adapter • 9
Integration into an MES • 9
Integration of a DCS • 19
Integration touchpoint (channel) • 20
Intended Audience • 1

L

Listeners • 11

M

MES client (responsibility) • 6
Message broker • 10
MSB (responsibility) • 6
MSB implementation • 36

P

Prerequisites • 9
ProcessConsumedMaterial request • 30
ProcessCreateDCSBatch request • 21
ProcessOrderContext request • 28
ProcessProducedMaterial request • 32

R

Reply
    AcknowledgeConsumedMaterial • 31
    AcknowledgeCreateDCSBatch • 23
    AcknowledgeOrderContext • 29
    AcknowledgeProducedMaterial • 32
    ShowBatchInformation • 34
    ShowDCSAlarmEvents • 24
    ShowDCSBatchValues • 27
Request
    GetBatchInformation • 33
    GetDCSAlarmEvents • 24
    GetDCSBatchValues • 26
    ProcessConsumedMaterial • 30
    ProcessCreateDCSBatch • 21
    ProcessOrderContext • 28
    ProcessProducedMaterial • 32
Rockwell Automation Download Site • 41

S

Setup the DCS Adapter configuration • 10

ShowBatchInformation reply • 34

ShowDCSAlarmEvents reply • 24

ShowDCSBatchValues reply • 27

Simulation of a DCS • 14

T

Test of the MES client without a DCS • 15

Testing the DCS integration without an MES client • 38

X

XML schema (interface) • 20

XML structure • 21