

| Service Hooks

Transact Extensibility for Java (English)

2020 Q1



©2020 Temenos Headquarters SA - all rights reserved.

| Copyright Notice

©2020 Temenos Headquarters SA - all rights reserved.

Warning: This document is protected by copyright law and international treaties. Unauthorised reproduction of this document, or any portion of it, may result in severe and criminal penalties, and will be prosecuted to the maximum extent possible under law.

©2020 Temenos Headquarters SA - all rights reserved.



| Lesson Overview



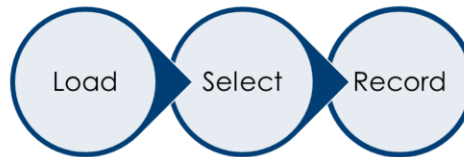
©2020 Temenos Headquarters SA - all rights reserved.



In this lesson, we will use the Java apis available to create a service in T24

| Services in T24

- Multithreaded processes
- Can be run independently or as part of COB



- Must implement `com.temenos.t24.api.hook.system.ServiceLifecycle`
- Equivalent of
 - LOAD – `initialize()`
 - SELECT – `getIds()` or `getTableName()`
 - RECORD – `inputRecord()` or `process()`

©2020 Temenos Headquarters SA - all rights reserved.



Services are multi threaded processes in core banking. They can be run independently or as part of COB. Multi threaded processes have three components – LOAD to initialize variables, SELECT – to prepare a list of IDs that Will be processed by the service, RECORD – contains the processing logic.

the **`com.temenos.t24.api.hook.system.ServiceLifecycle`** class has methods that are equivalent of LOAD, SELECT and RECORD.

getIds - This interface enables the implementer to return a list of Ids to the process, the RECORD routine (`process/inputRecord`) will then be invoked on all the ids in the list.

getTableName - This interface enables the implementer to return a table name, the RECORD routine (`process/inputRecord`) will then be invoked on all ids in the table.

Initialise - This interface enables the implementer to initialise the `ServiceLifecycle` instance e.g. set member variables.

inputRecord - Define a method to process a multiple records from the multiple tables specified in the 'versionName' parameter

postUpdateRequest - This interface enables the to post the request to update the records of any table in asynchronous mode.

Process - Define a method to process a single record from the table specified by `getTableName`

processSingleThreaded - This interface enables the implementer to define tasks to be executed during the running of services eg during end of day processing

| Initialise()

- This interface enables the implementer to initialise the ServiceLifecycle instance e.g. set member variables.
- Implementing this interface is optional and it will not be invoked if not specified.
- This interface is invoked from BATCH.JOB.CONTROL in place of the LOAD routine once before invoking the getTableName/getIds (SELECT) and process/updateRecord (RECORD) methods.
- The EB.API hook used by this interface is BATCH.JOB.NAME.LOAD.HOOK.
- This hook is specified by naming convention, the EB.API record for the initialise hook will append .LOAD to the job name

©2020 Temenos Headquarters SA - all rights reserved.



This interface enables the implementer to initialise the ServiceLifecycle instance e.g. set member variables.

Implementing this interface is optional and it will not be invoked if not specified.

This interface is invoked from BATCH.JOB.CONTROL in place of the LOAD routine once before invoking the getTableName/getIds (SELECT) and process/updateRecord (RECORD) methods.

The EB.API hook used by this interface is BATCH.JOB.NAME.LOAD.HOOK.

This hook is specified by naming convention, the EB.API record for the initialise hook will append .LOAD to the job name

| getIds()

- This interface enables the implementer to return a list of Ids to the process, the RECORD routine (process/updateRecord) will then be invoked on all the ids in the list
- If the control list is populated the getIds method will be invoked once for each item in the list and the process/updateRecord for every id in the list each time.
- The EB.API hook used by this interface is BATCH.JOB.NAME.SELECT.HOOK
- This hook is specified by naming convention, the EB.API record for the getIds hook will append .SELECT to the job name
- For example, where the job name is MY.JOB the hook will be MY.JOB.SELECT

public List<String> getIds([ServiceData](#) serviceData, List controlList)

- Parameters:
 - **serviceData** - It is all about the service like jobName, companyId, processId, sessionId and jobData
 - **controlList** - The control list to be set
- Returns:
 - **List<String>** - List of Ids to be processed

©2020 Temenos Headquarters SA - all rights reserved.



This interface enables the implementer to return a list of Ids to the process, the RECORD routine (process/updateRecord) will then be invoked on all the ids in the list. If the control list is populated the getIds method will be invoked once for each item in the list and the process/updateRecord for every id in the list each time.

The EB.API hook used by this interface is BATCH.JOB.NAME.SELECT.HOOK

This hook is specified by naming convention, the EB.API record for the getIds hook will append .SELECT to the job name

For example, where the job name is MY.JOB the hook will be MY.JOB.SELECT

Parameters:

serviceData - It is all about the service like jobName, companyId, processId, sessionId and jobData

controlList - The control list to be set

Returns:

List<String> - List of Ids to be processed

| getTableName()

- This interface enables the implementer to return a table name, the RECORD routine (process/updateRecord) will then be invoked on all ids in the table.
- If the control list is populated the getTableName method will be invoked once for each item
- This interface is invoked from BATCH.JOB.CONTROL in place of the SELECT routine BATCH.BUILD.LIST will be invoked for the table
- The EB.API hook used by this interface is BATCH.JOB.NAME.SELECT.HOOK
- This hook is specified by naming convention, the EB.API record for the getTableName hook will append .SELECT to the job name.
- For example, where the job name is MY.JOB the hook will be MY.JOB.SELECT

©2020 Temenos Headquarters SA - all rights reserved.



This interface enables the implementer to return a table name, the RECORD routine (process/updateRecord) will then be invoked on all ids in the table.

If the control list is populated the getTableName method will be invoked once for each item

This interface is invoked from BATCH.JOB.CONTROL in place of the SELECT routine BATCH.BUILD.LIST will be invoked for the table

The EB.API hook used by this interface is BATCH.JOB.NAME.SELECT.HOOK

This hook is specified by naming convention, the EB.API record for the getTableName hook will append .SELECT to the job name.

For example, where the job name is MY.JOB the hook will be MY.JOB.SELECT

| getTableName() - Parameters

public String getTableName([ServiceData](#) serviceData, List controlList)

- Parameters:
 - **serviceData** - It is all about the service like jobName, companyId, processId, sessionId and jobData.
 - **controlList** - The control list to be setReturns:List<String> - the list of ids are to be processed
- Returns:
 - **List<String>** - the list of ids are to be processed

©2020 Temenos Headquarters SA - all rights reserved.



Parameters:

serviceData - It is all about the service like jobName, companyId, processId, sessionId and jobData.

controlList - The control list to be setReturns:List<String> - the list of ids are to be processed

Returns:

List<String> - the list of ids are to be processed

| updateRecord()

- This interface enables the developer to update the records of any table.
- The update will be attempted before control is returned to the service dispatcher BATCH.JOB.CONTROL.
- This interface is invoked from BATCH.JOB.CONTROL in place of the RECORD routine.
- The EB.API hook used by this interface is BATCH.JOB.NAME.HOOK.
- The versionId should not be a comma version.
- The T24 field specifying the job name is the JOB.NAME field in BATCH.
- The OFS.SOURCE id should be specified in the first element of the transactionData parameter.
- The transaction will not be processed if the OFS.SOURCE is missing or invalid

©2020 Temenos Headquarters SA - all rights reserved.



This interface enables the developer to update the records of any table.

The update will be attempted before control is returned to the service dispatcher BATCH.JOB.CONTROL.

This interface is invoked from BATCH.JOB.CONTROL in place of the RECORD routine.

The EB.API hook used by this interface is BATCH.JOB.NAME.HOOK.

The versionId should not be a comma version.

The T24 field specifying the job name is the JOB.NAME field in BATCH.

The OFS.SOURCE id should be specified in the first element of the transactionData parameter.

The transaction will not be processed if the OFS.SOURCE is missing or invalid

| updateRecord () - Parameters

```
public void updateRecord(String id, ServiceData serviceData, String  
controllItem, TransactionControl transactionControl, List transactionData, List  
records)
```

Parameters:

- **id** - The id to process
- **serviceData** - Details about this service such as jobName, companyId, processId, sessionId, jobData and batchStartDate.
- **controllItem** - The item of the control list being processed for multi stage jobs.
- **transactionControl** - Items that control how the request is processed such as how to group transactions and handle errors.
- **transactionData** - List of transaction data each for a single update request including function, transactionId and versionId.
- **records** - List of records for the corresponding request index in transactionData. A record is only required for the INPUT function. This is an INOUT parameter
- Returns
 - **nothing**

©2020 Temenos Headquarters SA - all rights reserved.



```
public void updateRecord(String id, ServiceData serviceData, String controllItem,  
TransactionControl transactionControl, List transactionData, List records)
```

Parameters:

- id** - The id to process
- serviceData** - Details about this service such as jobName, companyId, processId, sessionId, jobData and batchStartDate.
- controllItem** - The item of the control list being processed for multi stage jobs.
- transactionControl** - Items that control how the request is processed such as how to group transactions and handle errors.
- transactionData** - List of transaction data each for a single update request including function, transactionId and versionId.
- records** - List of records for the corresponding request index in transactionData. A record is only required for the INPUT function

Returns

nothing

| postUpdateRequest()

- This interface enables the to post the request to update the records of any table in asynchronous mode.
- This interface writes the request into the file OFS.MESSAGE.QUEUE. To post this request into the T24, user has to run the OFS.MESSAGE.SERVICE service. This interface is invoked from BATCH.JOB.CONTROL in place of the RECORD routine.
- The EB.API hook used by this interface is BATCH.JOB.NAME.HOOK.
- The versionId should not be a comma version. The T24 field specifying the job name is the JOB.NAME field in BATCH. The OFS.SOURCE id should be specified in the DATA field (first multi value position) in BATCH.
- If no OFS.SOURCE mentioned or invalid OFS.SOURCE, then the process stops from being processed.

©2020 Temenos Headquarters SA - all rights reserved.



This interface enables the to post the request to update the records of any table in asynchronous mode.

This interface writes the request into the file OFS.MESSAGE.QUEUE. To post this request into the T24, user has to run the OFS.MESSAGE.SERVICE service. This interface is invoked from BATCH.JOB.CONTROL in place of the RECORD routine.

The EB.API hook used by this interface is BATCH.JOB.NAME.HOOK.

The versionId should not be a comma version. The T24 field specifying the job name is the JOB.NAME field in BATCH. The OFS.SOURCE id should be specified in the DATA field (first multi value position) in BATCH.

If no OFS.SOURCE mentioned or invalid OFS.SOURCE, then the process stops from being processed.

| postUpdateRequest() - Parameters

public void postUpdateRequest(String id, ServiceData serviceData, String controllItem, List transactionData, List records)

Parameters:

- **id** - The id to process
- **serviceData** - It is all about the service like jobName, companyId, processId, sessionId, jobData and batchStartDate.
- **controllItem** - The item of the control list being processed for multi stage jobs.
- **transactionData** - List of transaction data each for a single update request including function, transactionId and versionId.
- **records** - List of records for the corresponding request index in transactionData. A record is only required for the INPUT function. This is an INOUT parameter
- Returns:
 - **nothing.**

©2020 Temenos Headquarters SA - all rights reserved.



public void postUpdateRequest(String id, ServiceData serviceData, String controllItem, List transactionData, List records)

Parameters:

- id** - The id to process
- serviceData** - It is all about the service like jobName, companyId, processId, sessionId, jobData and batchStartDate.
- controllItem** - The item of the control list being processed for multi stage jobs.
- transactionData** - List of transaction data each for a single update request including function, transactionId and versionId.
- records** - List of records for the corresponding request index in transactionData. A record is only required for the INPUT function.

Returns:

nothing.

| Lesson Summary



| Practice 9.1

- Create a simple service that updates the ACCOUNT.TITLE of all accounts as MNEMONIC + TITLE of the CUSTOMERS with SECTOR 1001
- Steps
 - Select the Customers with SECTOR equals 1001
 - Read all the Accounts of the Customers one by one
 - Update all the Accounts ACCOUNT.TITLE in the below format
 - MNEMONIC + SHORT.TITLE

©2020 Temenos Headquarters SA - all rights reserved.



Create a simple service that updates the ACCOUNT.TITLE of all accounts as MNEMONIC + TITLE of the CUSTOMERS with SECTOR 1001

Steps

Select the Customers with SECTOR equals 1001

Read all the Accounts of the Customers one by one

Update all the Accounts ACCOUNT.TITLE in the below format

MNEMONIC + SHORT.TITLE

| Solution 9.1 - .SELECT

```
68=  @Override
69  public List<String> getIds(ServiceData serviceData, List<String> controllList) {
70
71      DataAccess dataAccess = new DataAccess(this);
72
73      // Select the Customers with Sector 1001
74      List<String> recIds = dataAccess.selectRecords("BNK", "CUSTOMER", "", "WITH SECTOR EQ 1001");
75
76      return recIds;
77  }
```

Solution 9.1 - .RECORD

```
29= @Override
30 public void updateRecord(String id, ServiceData serviceData, String controlItem,
31 TransactionControl transactionControl, List<SynchronousTransactionData> transactionData,
32 List<TStructure> records) {
33
34     DataAccess dataAccess = new DataAccess(this);
35
36     Customer customer = new Customer(this);
37
38     customer.setCustomerId(id); //Set the Customer Id
39
40     List<String> accountNumbers = customer.getAccountNumbers(); // Read all account numbers using Core API
41
42     for (String accountNo : accountNumbers) { // Loop through each account and update the title
43
44         AccountRecord accountRecOld = new AccountRecord(dataAccess.getRecord("ACCOUNT", accountNo)); // Read the Account
45
46         AccountRecord accountRecNew = new AccountRecord(); // Create an empty Acc record to set the modified field values
47
48         //Set the new Account Title in the Empty Account Record Object
49         accountRecNew.setAccountTitle1(accountRecOld.getMnemonic().toString() + accountRecOld.getShortTitle(0), 0);
50
51         SynchronousTransactionData sTxnData = new SynchronousTransactionData(); // Set the Transaction Data
52
53         sTxnData.setVersionId("ACCOUNT, JAVA"); //Version Name
54         sTxnData.setFunction("INPUT"); // Function
55         sTxnData.setNumberOfAuthoriser("0"); // No of Auth
56         sTxnData.setTransactionId(accountNo); // Record ID
57         sTxnData.setSourceId("BULK.OFS"); // OFS.SOURCE ID
58
59         transactionData.add(sTxnData);
60
61         records.add(accountRecNew.toStructure()); // Add the Records to the list
62     }
63 }
```

Solution 9.1 – EB.API Records

EB.API	SERVICE.DEMO.SELECT
Description	EN Service Demo Select
Protection Level	Full
Source Type	Method
Java Method	getIds
Java Class	UpdateShortNameService
Java Package	com.temenos.training
Curr No	2
Inputter.1	18_AUTHORISER_OFS_BROWSERTC
Date time.1	08 MAY 20 22:10
Authoriser	18_AUTHORISER_OFS_BROWSERTC
Company	GB0010001
Dept Code	1

EB.API	SERVICE.DEMO
Description	EN Service Demo
Protection Level	Full
Source Type	Method
Java Method	updateRecord
Java Class	UpdateShortNameService
Java Package	com.temenos.training
Curr No	2
Inputter.1	18_AUTHORISER_OFS_BROWSERTC
Date time.1	08 MAY 20 22:10
Authoriser	18_AUTHORISER_OFS_BROWSERTC
Company	GB0010001
Dept Code	1

©2020 Temenos Headquarters SA - all rights reserved.



We must create EB.API entry for the .SELECT as well as the record routine

| Solution 9.1 – PGM.FILE

PGM.FILE	SERVICE.DEMO
Type	B
Batch Job.1	@BATCH.JOB.CONTROL
Product	EB
Curr No	1
Inputter.1	7_AUTHORISER_OFS_BROWSERTC
Date time.1	09 APR 20 11:06
Authoriser	7_AUTHORISER_OFS_BROWSERTC
Company	GB0010001
Dept Code	1

©2020 Temenos Headquarters SA - all rights reserved.

 **TEMENOS**
Learning Community

Create a PGM.FILE entry with Type as Batch and BATCH.JOB as @BATCH.JOB.CONTROLS and Product as EB

Solution 9.1 – BATCH Record

BATCH BNK/SERVICE.DEMO	
Process Status	0
Batch Environment	F FOREGROUND
Job Name.1	SERVICE.DEMO
Frequency.1	D
Job Status.1	0
Last Payment.1	17 APR 2019
Curr No	1
Inputter.1	6_AUTHORISER_OFS_BROWSERTC
Date time.1	09 APR 20 11:07
Authoriser	6_AUTHORISER_OFS_BROWSERTC
Company	GB0010001
Dept Code	1

©2020 Temenos Headquarters SA - all rights reserved.

 **TEMENOS**
Learning Community

Create a BATCH Record with the ID as BNK/<NAME OF THE RECORD ROUTINE EB.API>

| Solution 9.1 – TSA.SERVICE

TSA.SERVICE	BNK/SERVICE.DEMO
Description.1	Service Demo
User	INPUTTER
Service Control	Stop
Date.1	17 APR 2019
Started.1	09/04/2020 11:25:35
Stopped.1	09/04/2020 11:28:25

©2020 Temenos Headquarters SA - all rights reserved.



Create TSA.SERVICE Record with the ID as BNK/<NAME OF THE RECORD ROUTINE
EB.API ID>

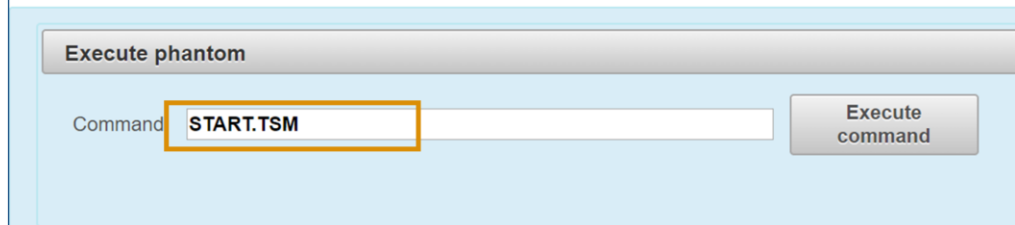
Start the service

| Solution 9.1

Execute servlet

You can post command to the ExecQueue by submitting your command in the field below.

e.g. to execute TSM service within the application server post command **START.TSM**



The screenshot shows a web interface titled "Execute phantom". It features a text input field labeled "Command" with the text "START.TSM" entered. The input field is highlighted with an orange border. To the right of the input field is a button labeled "Execute command".

Start the TSM from the TAFJEE console using the command START.TSM

Solution 9.1

ACCOUNT	88633	MAXCHUARDMax Chuard
Customer	190278	
Category	6001	
Name	EN	MAXCHUARDMax Chuard
Account Name	EN	Max Chuard
Mnemonic	MAXCHUARD	
Position Type	TR	
Ccy	USD	
Market	1	
Account Officer	1	
Group	3	
Cap Date Charge.1	30 APR 2019	

©2020 Temenos Headquarters SA - all rights reserved.

Account Name Updated



TEMENOS
Learning Community

thank.you

tlc.temenos.com

