TEMENOS

THE BANKING SOFTWARE COMPANY

Application Exit Point Hook

Customer How-to Guide 1.0 February 2020

Contents

1 About this how-to guide

The **Application Exit Point Hook How-to Guide** describes how to write hook routines in Java for an application exit point.

1.1 Prerequisites

This guide assumes that:

- You have already read the Java Extensibility Framework Overview.
- You have a basic understanding of Java.
- Have verified JD product is installed in SPF.

1.2 Legal

1.3 History

Version	Date	Change	Author
1.0	February 2020	Initial release	Lizen Bista

2 Introduction

The **Application Exit Point Hook How-to Guide** describes how to write hook routines in Java for an application exit point.

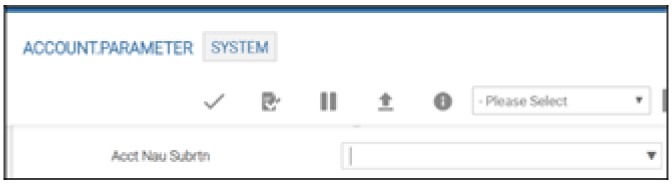
The use case in this document uses the application exit point field ACCOUNT.NAU.SUBRTN in the ACCOUNT.PARAMETER application. You can attach a local development Java routine to the field. This routine is invoked in the core accounting routine to raise overrides.

3 Determining the superclass and method

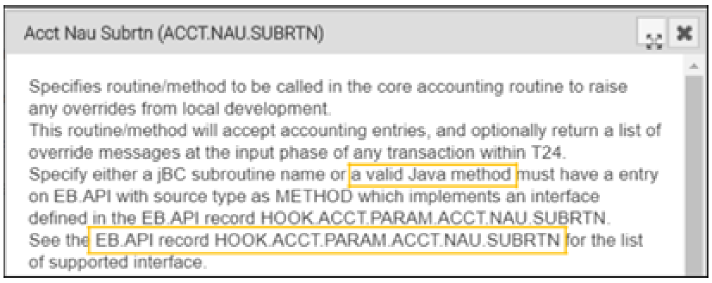
This section shows you how to determine the superclass and method you need to implement.

Procedure

1. Identify the application exit point, in this case, ACCOUNT.NAU.SUBRTN in ACCOUNT.PARAMETER.



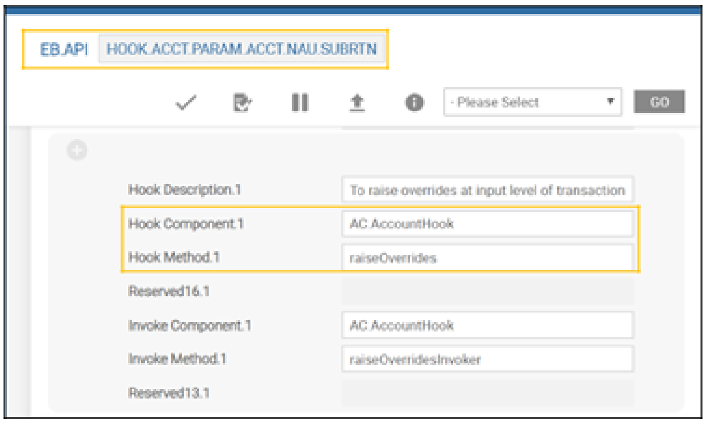
2. The help text for the exit point field tells you if a Java routine can be attached to the field.



Acct Nau Subrtn (ACCT.NAU.SUBRTN)

Specifies routine/method to be called in the core accounting routine to raise any overrides from local development.
This routine/method will accept accounting entries, and optionally return a list of override messages at the input phase of any transaction within T24.
Specify either a jBC subroutine name or a valid Java method must have a entry on EB.API with source type as METHOD which implements an interface defined in the EB.API record HOOK.ACCT.PARAM.ACCT.NAU.SUBRTN.
See the EB.API record HOOK.ACCT.PARAM.ACCT.NAU.SUBRTN for the list of supported interface.

3. View the EB.API record indicated in the help text. This record gives you the name of the hook method to override in the Java implementation of the exit point routine.



EB.API HOOK.ACCT.PARAM.ACCT.NAU.SUBRTN

Hook Description.1	To raise overrides at input level of transaction
Hook Component.1	AC.AccountHook
Hook Method.1	raiseOverrides
Reserved16.1	
Invoke Component.1	AC.AccountHook
Invoke Method.1	raiseOverridesInvoker
Reserved13.1	

4. The L3 API documentation provides a detailed description of the hook method and its parameters. You can download Javadocs for L3 API from either the Temenos customer support portal (TCSP) or the Temenos partner support portal (TPSP) as a zip file.

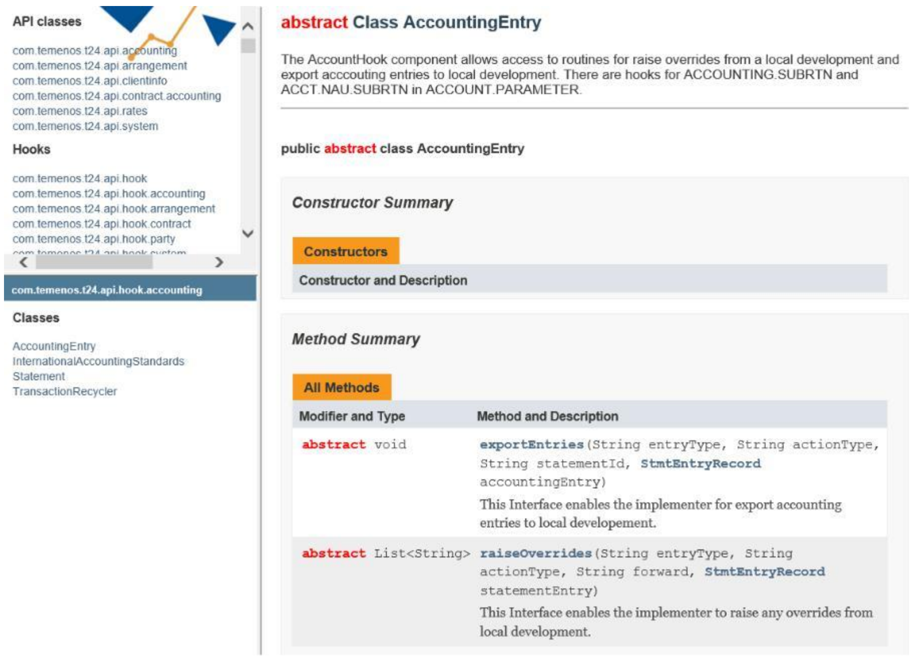
   After you have downloaded the zip file:

   a. Extract `T24.javadoc.jar` from the zip file.

   b. Double-click `T24.javadoc.jar`. This extracts the contents of the jar into a newly created subdirectory called `T24.javadoc`.

   c. Double-click `T24.javadoc/index.html` to view the complete L3 API documentation in the browser.

5. Select the package `com.temenos.t24.api.hook.accounting` and the class `AccountingEntry`.


API classes

com.temenos.t24.api.accounting
com.temenos.t24.api.arrangement
com.temenos.t24.api.clientinfo
com.temenos.t24.api.contract.accounting
com.temenos.t24.api.rates
com.temenos.t24.api.system

Hooks

com.temenos.t24.api.hook
com.temenos.t24.api.hook.accounting
com.temenos.t24.api.hook.arrangement
com.temenos.t24.api.hook.contract
com.temenos.t24.api.hook.party

com.temenos.t24.api.hook.accounting

Classes

AccountingEntry
InternationalAccountingStandards
Statement
TransactionRecycler

abstract Class AccountingEntry

The AccountHook component allows access to routines for raise overrides from a local development and export acccouting entries to local development. There are hooks for ACCOUNTING.SUBRTN and ACCT.NAU.SUBRTN in ACCOUNT.PARAMETER.

public abstract class AccountingEntry

Constructor Summary

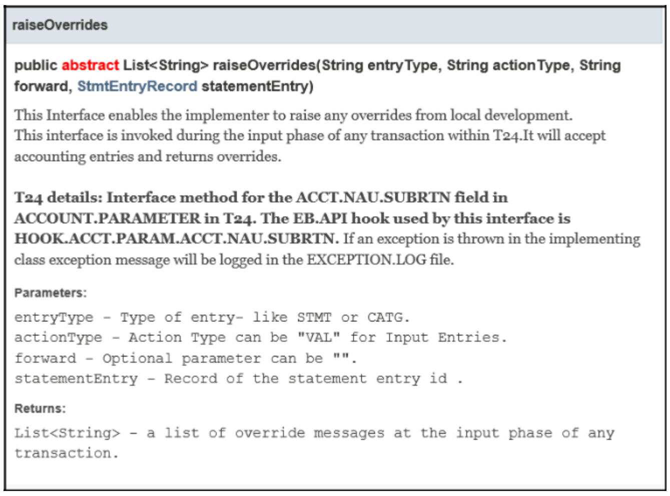
Constructors

Constructor and Description

Method Summary

All Methods

Modifier and Type	Method and Description
`abstract void`	`exportEntries(String entryType, String actionType, String statementId, StmtEntryRecord accountingEntry)` This Interface enables the implementer for export accounting entries to local developement.
`abstract List<String>`	`raiseOverrides(String entryType, String actionType, String forward, StmtEntryRecord statementEntry)` This Interface enables the implementer to raise any overrides from local development.


raiseOverrides

public abstract List<String> raiseOverrides(String entryType, String actionType, String forward, StmtEntryRecord statementEntry)

This Interface enables the implementer to raise any overrides from local development.
This interface is invoked during the input phase of any transaction within T24.It will accept accounting entries and returns overrides.

T24 details: Interface method for the ACCT.NAU.SUBRTN field in ACCOUNT.PARAMETER in T24. The EB.API hook used by this interface is HOOK.ACCT.PARAM.ACCT.NAU.SUBRTN. If an exception is thrown in the implementing class exception message will be logged in the EXCEPTION.LOG file.

Parameters:

```
entryType - Type of entry- like STMT or CATG.
actionType - Action Type can be "VAL" for Input Entries.
forward - Optional parameter can be "".
statementEntry - Record of the statement entry id .
```

Returns:

```
List<String> - a list of override messages at the input phase of any
transaction.
```

3.1.1 Javadoc updates

Javadoc updates are packaged with the T24 updates zip that you download from the Updates portal. The updates are component-wise Javadoc updates, for example, `AC_AccountHook.javadoc.jar` in the `Transact_L3_Javadoc` folder.

Procedure

1. Extract the jar.
2. Use the `{TAFJ_HOME}/bin/tJavadocMerge` tool to perform the merge.

```
tJavadocMerge -merge <List of component-wise.javadoc.jar>
<Base javadoc jar> -o <new_jar_name>
```

Example 1

```
tJavadocMerge -merge
C:\DocUpdates\AC_AccountHook.javadoc.jar
C:\JavaDoc\T24.javadoc.jar -o C:\JavaDoc\T24-1.javadoc.jar
```

Example 2

```
tJavadocMerge -merge C:\DocUpdates\*.javadoc.jar
C:\JavaDoc\T24.javadoc.jar -o C:\Javadoc\T24-2.javadoc.jar
```

In both examples, `T24.javadoc.jar` is the L3 API document base.

4 Writing the Java implementation

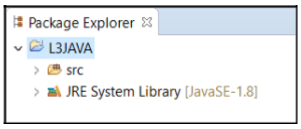
This section describes how to write the Java implementation for the application exit point routine.

To write the hook routine in Java, start Design Studio and switch to the Java perspective.

4.1 Create the new Java project

Procedure

1. In Design Studio, create the new Java project (**File > New > Java project**). In the wizard, supply a project name.

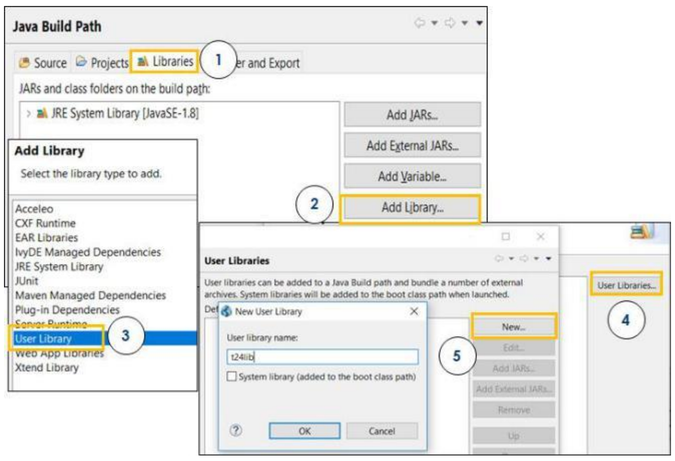


2. Configure the build path settings for the Java project to add dependent T24 and TAFJ libraries.

   a. Right click the **project > Build path > Configure build** path.

   b. Click the **Libraries tab > Add Library > User Library > User Libraries**.

   c. In the **User Libraries** window, click **New** and give the library a name, for example **t24lib**.



JARs and class folders on the build path:

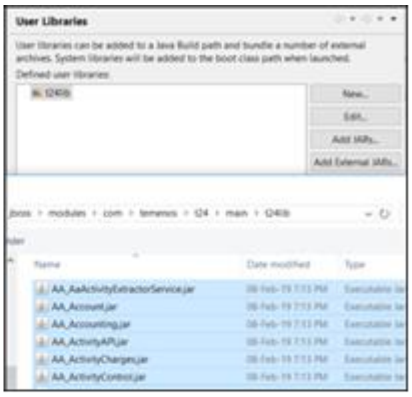
Select the library type to add.

User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

User library name: t24lib

d. Click **Add external jars**. Navigate to the T24 libraries folder under `%JBOSS_HOME%/modules`. Select all the jars and click **Open**. Click **OK** and finish.

> Alternatively, you can add the required T24 hook jars, such as EB_TemplateHook.jar and so on, using Add External JARS.

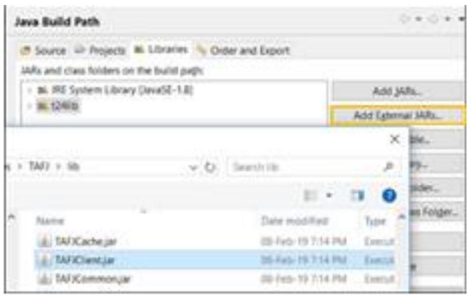


User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

Name	Date modified	Type
AA_AaActivityExtractorService.jar	08-Feb-19 7:13 PM	Executable Jar
AA_Account.jar	08-Feb-19 7:13 PM	Executable Jar
AA_Accounting.jar	08-Feb-19 7:13 PM	Executable Jar
AA_ActivityAPI.jar	08-Feb-19 7:13 PM	Executable Jar
AA_ActivityCharges.jar	08-Feb-19 7:13 PM	Executable Jar
AA_ActivityControl.jar	08-Feb-19 7:13 PM	Executable Jar

e. Click **Add External JARs** and add `TAFJClient.jar` from `%TAFJ_HOME%/lib`



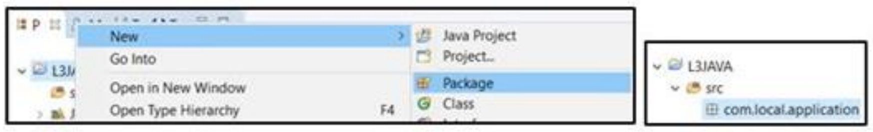
JARs and class folders on the build path:

Name	Date modified	Type
TAFJCache.jar	08-Feb-19 7:14 PM	Execut
TAFJClient.jar	08-Feb-19 7:14 PM	Execut
TAFJCommon.jar	08-Feb-19 7:14 PM	Execut

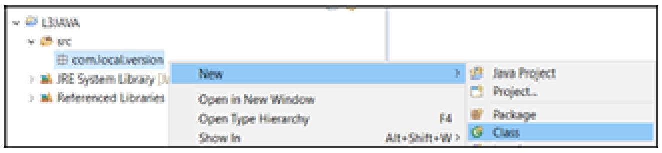
4.2 Extend the superclass identified in Javadoc

Procedure

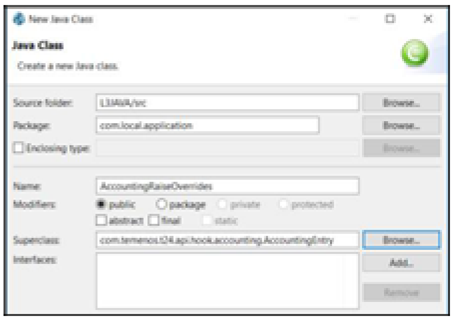
1. Create a new Java package. Right click **Project > New > Package** and give the package a name.



2. Right click the package and add a new Class to the package. Provide a class name, for example, `AccountingRaiseOverrides`.



3. Extend the superclass `AccountingEntry` for your class. Click **Finish**.

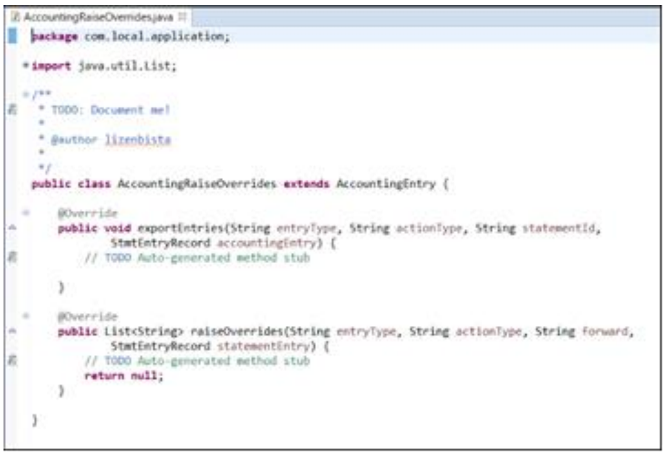

New Java Class

Java Class

Create a new Java class.

Source folder:	L3JAVA/src
Package:	com.local.application
☐ Enclosing type:	
Name:	AccountingRaiseOverrides
Modifiers:	● public ○ package ○ private ○ protected
	☐ abstract ☐ final ☐ static
Superclass:	com.temenos.t24.api.hook.accounting.AccountingEntry
Interfaces:	

4. The method outlines of the superclass `AccountingEntry` is generated. The class has two methods that can be overridden. Implement the hook method indicated in the EB.API record for the exit point.


AccountingRaiseOverrides.java

```
package com.local.application;

import java.util.List;

/**
 * TODO: Document me!
 *
 * @author lizenbista
 *
 */
public class AccountingRaiseOverrides extends AccountingEntry {

    @Override
    public void exportEntries(String entryType, String actionType, String statementId,
            StmtEntryRecord accountingEntry) {
        // TODO Auto-generated method stub

    }

    @Override
    public List<String> raiseOverrides(String entryType, String actionType, String forward,
            StmtEntryRecord statementEntry) {
        // TODO Auto-generated method stub
        return null;
    }

}
```

4.3 Write the Java Implementation

Procedure

1. Override the inherited method `raiseOverrides` to do the following:
    a. Raise override if the transaction is input beyond a specified time.
    b. Raise override if DEBIT.VALUE.DATE does not match the T24 TODAY date.
    c. Raise override if CREDIT.VALUE.DATE falls on a holiday.
2. Click CTRL + S to save the code. See code sample below.

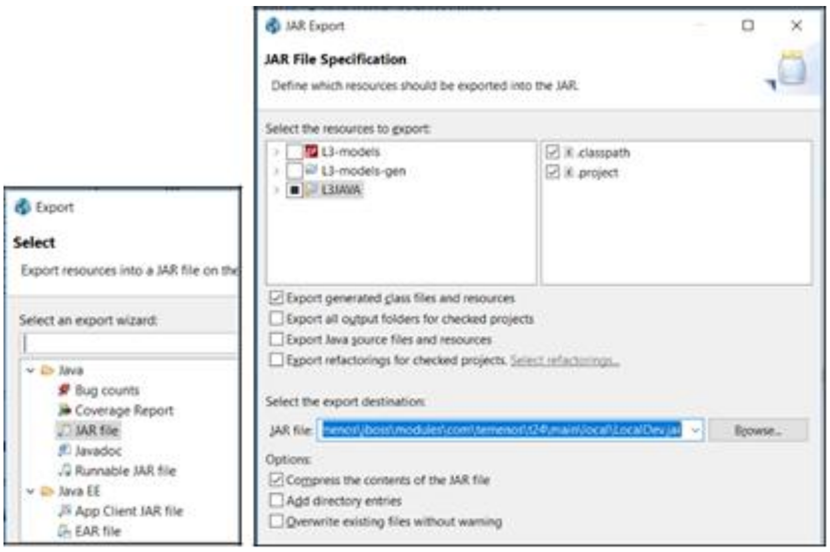
```
package com.local.application;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.temenos.api.TDate;
import
com.temenos.t24.api.hook.accounting.
AccountingEntry; import
com.temenos.t24.api.records.stmtentr
y.StmtEntryRecord; import
com.temenos.t24.api.system.Date;
```

```
    public class AccountingRaiseOverrides extends AccountingEntry {

    @Override
    public List<String> raiseOverrides(String entryType,
    String actionType, String forward,StmtEntryRecord
    statementEntry) {
        List<String> override = new
        ArrayList<String>(); Date
        date = new Date(this);

        String creditDate = statementEntry.getValueDate().toString();
        TDate creditValueDate = new TDate(creditDate);
        String type = date.getDayType(creditValueDate);

        if (isAfterCutoffTime("18:00:00")) {
            override.add("L3HOOK-CHECK.CUTOFF.TIME");
        }
        if (type.equals("HOLIDAY")) {
            override.add("L3HOOK-CREDIT.DATE.IS.HOLIDAY");
        }
        return override;
    }

    private boolean
        isAfterCutoffTime(String
        cutoffTime) { java.util.Date
        cutOff;
        java.util.Date systemTime;

        DateFormat df = new
        SimpleDateFormat("HH:mm:ss");
        Calendar calobj =
        Calendar.getInstance();
        System.out.println(df.format(
        calobj.getTime())); String
        formattedDate =
        df.format(calobj.getTime());
        try {
            systemTime = df.parse(formattedDate);
            cutOff = df.parse(cutoffTime);
        } catch (ParseException e) {
            throw new RuntimeException("Cannot parse time!");
        }
        return systemTime.after(cutOff);
    }

    @Override
    public void exportEntries(String entryType, String
            actionType, String statementId, StmtEntryRecord
            accountingEntry) {
        // TODO Auto-generated method stub
    }
}
```

4.4 Place the implementation in a library and load in JBoss Classpath

Procedure

1. After the Java code is written, right click the **project > Export > JAR** file. Select the export destination.



Select the resources to export: [✓ L3JAVA], [✓ .classpath], [✓ .project]

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

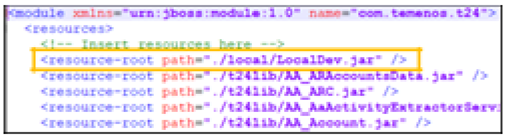
Select the export destination:

JAR file: menos\jboss\modules\com\temenos\t24\main\local\LocalDev.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

2. Update `module.xml` in `%JBOSS_HOME%/modules` with the new jar path and name. Restart JBoss.

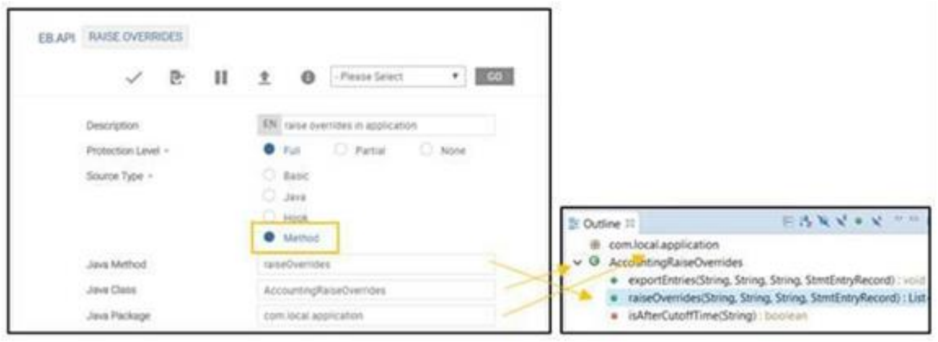


```
<module xmlns="urn:jboss:module:1.0" name="com.temenos.t24">
  <resources>
    <!-- Insert resources here -->
    <resource-root path="./local/LocalDev.jar" />
    <resource-root path="./t24lib/AA_ARAccountsData.jar" />
    <resource-root path="./t24lib/AA_ARC.jar" />
    <resource-root path="./t24lib/AA_AaActivityExtractorServi
    <resource-root path="./t24lib/AA_Account.jar" />
```

4.5 Link the Java method to the application template flow via exit point

Procedure

1. Create an EB.API entry for the hook routine.



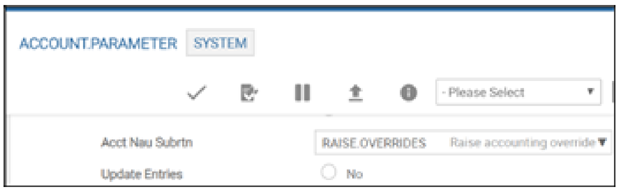
EB.API RAISE.OVERRIDES

- Please Select GO

Description	EN raise overrides in application
Protection Level *	[✓ Full] Partial None
Source Type *	Basic Java Hook [✓ Method]
Java Method	raiseOverrides
Java Class	AccountingRaiseOverrides
Java Package	com.local.application

```
Outline
  com.local.application
v AccountingRaiseOverrides
    exportEntries(String, String, String, StmtEntryRecord) : void
    raiseOverrides(String, String, String, StmtEntryRecord) : List
    isAfterCutoffTime(String) : boolean
```

2. Attach the hook routine to the application exit point by supplying the EB.API record ID created in the previous step.



ACCOUNT.PARAMETER SYSTEM

- Please Select

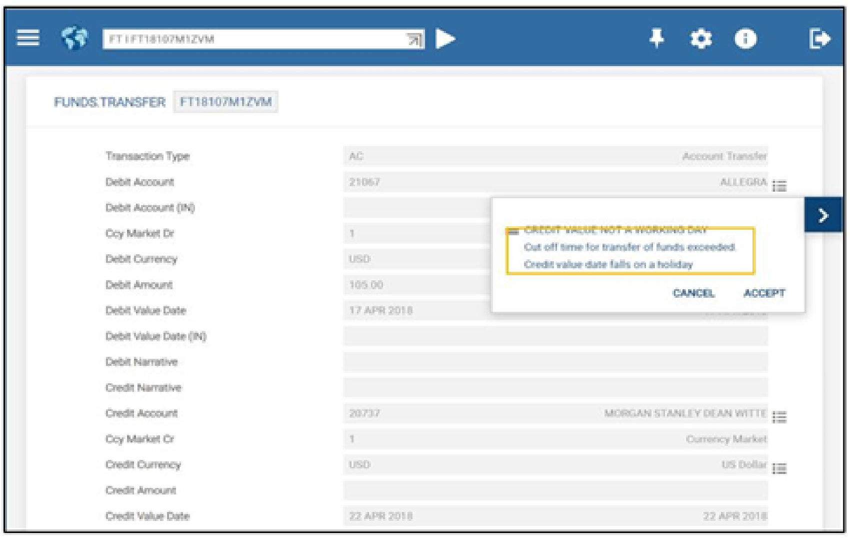
Acct Nau Subrtn	RAISE.OVERRIDES Raise accounting override
Update Entries	No

5 Test the application exit point

Procedure

1. Launch FUNDS.TRANSFER application.

   The java implementation of the Account unauthorized subroutine executes and raises overrides. The Override IDs in the code are replaced with message text from the OVERRIDE application.



FT I FT18107M1ZVM

FUNDS.TRANSFER FT18107M1ZVM

Transaction Type	AC	Account Transfer
Debit Account	21067	ALLEGRA
Debit Account (IN)		
Ccy Market Dr	1	
Debit Currency	USD	
Debit Amount	105.00	
Debit Value Date	17 APR 2018	
Debit Value Date (IN)		
Debit Narrative		
Credit Narrative		
Credit Account	20737	MORGAN STANLEY DEAN WITTE
Ccy Market Cr	1	Currency Market
Credit Currency	USD	US Dollar
Credit Amount		
Credit Value Date	22 APR 2018	22 APR 2018

~~CREDIT VALUE NOT A WORKING DAY~~
Cut off time for transfer of funds exceeded.
Credit value date falls on a holiday