TEMENOS

THE BANKING SOFTWARE COMPANY

Java Extensibility Framework

Customer Overview 1.0 February 2020

Contents

1 About this overview

The **Java Extensibility Framework Overview** describes how banks can write large amounts of custom Java code, called L3, to extend the capabilities of Transact.

1.1 Legal

The information in this guide relates to TEMENOS™ information, products and services. It also includes information, data and keys developed by other parties.

While all reasonable attempts have been made to ensure accuracy, currency and reliability of the content in this guide, all information is provided "as is".

There is no guarantee as to the completeness, accuracy, timeliness or the results obtained from the use of this information. No warranty of any kind is given, expressed or implied, including, but not limited to warranties of performance, merchantability and fitness for a particular purpose.

In no event will TEMENOS be liable to you or anyone else for any decision made or action taken in reliance on the information in this document or for any consequential, special or similar damages, even if advised of the possibility of such damages.

TEMENOS does not accept any responsibility for any errors or omissions, or for the results obtained from the use of this information. Information obtained from this guide should not be used as a substitute for consultation with TEMENOS.

References and links to external sites and documentation are provided as a service. TEMENOS is not endorsing any provider of products or services by facilitating access to these sites or documentation from this guide.

The content of this guide is protected by copyright and trademark law. Apart from fair dealing for the purposes of private study, research, criticism or review, as permitted under copyright law, no part may be reproduced or reused for any commercial purposes whatsoever without the prior written permission of the copyright owner. All trademarks, logos and other marks shown in this guide are the property of their respective owners.

1.2 History

Version	Date	Change	Author
1.0	February 2020	Initial release	Lizen Bista

2 Introduction

This **Java Extensibility Framework Customer Overview** provides you with an overview of the Java Extensibility Framework.

Temenos Transact is designed to allow the extensive addition of new capabilities by banks during implementation. The ability to extend Transact - formerly known as T24 - allows a bank to write large amounts of custom code, called L3. L3 Java enables customers and partners to write safe L3 customer implementation code in Java.

2.1 Benefits

The benefits of L3 Java for customisation for both customers and partners include:

- Extending Transact using Java.
- Simplifying Transact APIs. You only need a basic understanding of Transact to carry out customisation.
- Giving customers and partners the ability to use their own developers to carry out L3 customisation.

This is achieved by extending Transact customisation in Java and reducing the level of technical expertise in Transact that's needed, such as an understanding of the Transact context and its workflow.

2.2 Usage scenarios

Every Transact implementation requires some degree of customisation to satisfy customer requirements. L3 Java supports the following usage scenarios for a customer development:

- Updating User Defined Tables.
- Validating any transaction IDs entered.
- Auto populating fields.
- Cross-validating records and fields.
- Altering and/or defaulting other field values in the record based on a field value.
- Updating local reference fields.
- Raising errors / overrides.

- Defining services.
- Defining COB jobs.
- Combining data from different applications.

.

3 Transact extensibility workflow

The following sections describe the high-level workflow for writing a hook routine in Java and attaching it to an application exit point. For detailed steps and prerequisites, see Jumpstart's Java Extensibility Framework customer how-to guides.

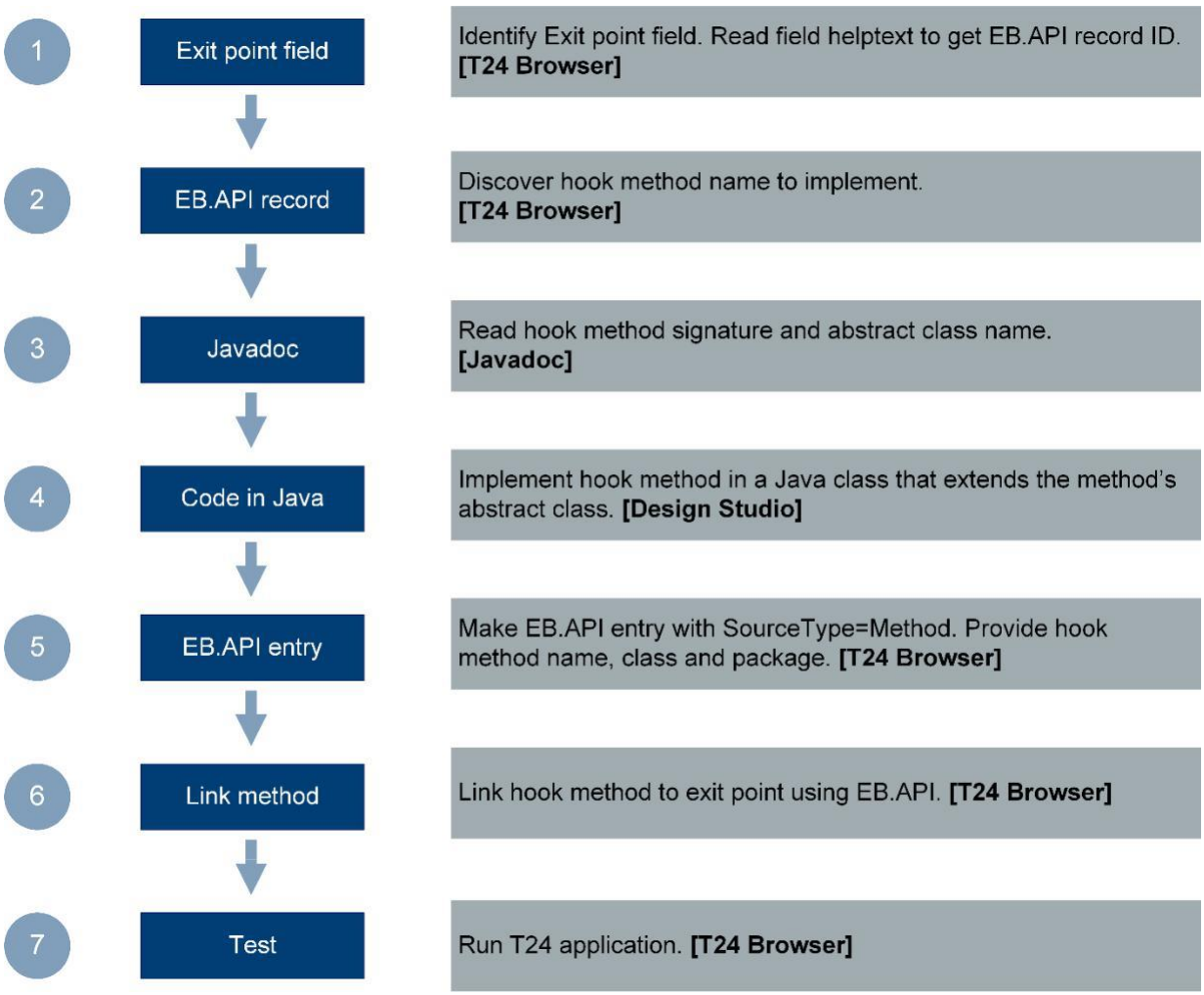
The programmatic extension of Transact is carried out using hooks. A hook is an exit point (typically a field) in a Transact application where L3 developers can attach their own logic to modify or extend the existing behaviour of the application.

The attached code is called from core at specific points in the application flow during execution. You can find examples of these hooks in VERSION, VERSION.CONTROL, ENQUIRY, EB.TABLE.PROCEDURES, SERVICES etc.

Temenos has enabled many of these hooks for java. The L3 developer can write extensions to Transact in Java using readily available, user-friendly, intuitive APIs and attach them to Java enabled hooks or exit points.

> You can find a list of all APIs, their signature and use, in the Javadoc for L3 API.

3.1 Workflow diagram



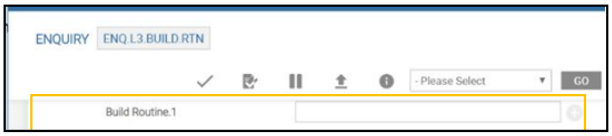
3.2 Example procedure

The following example procedure describes the workflow for attaching an enquiry build routine written in Java that modifies the selection criteria at runtime.

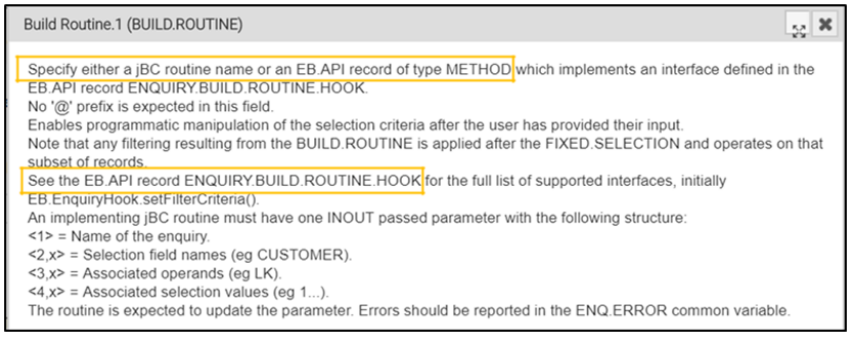
Procedure

1. Identify the exit point field in the ENQUIRY application for the Java hook routine.

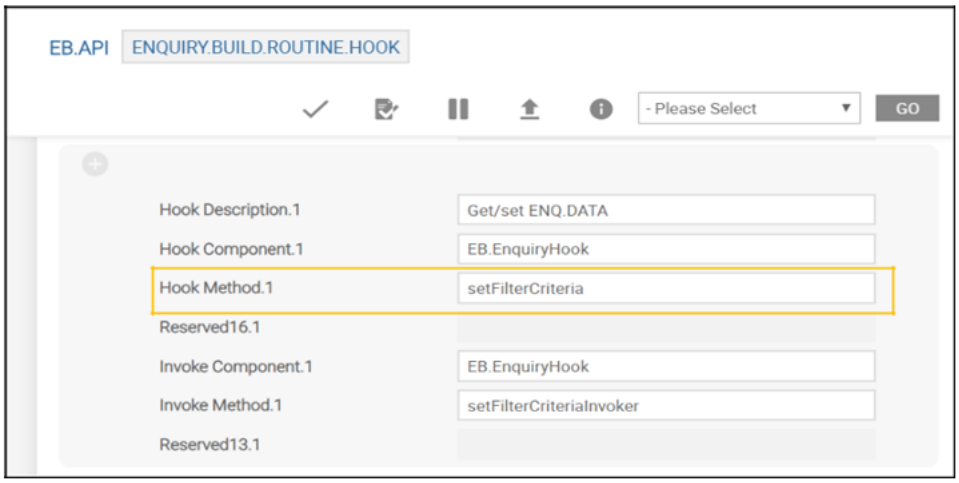
Requirement	Attach a selection criteria 'SECTOR.CODE LK 1…' at runtime for an enquiry. Raise an error if SHORT.NAME is used as selection criteria.
Exit point field	BUILD.ROUTINE in ENQUIRY



2. View the help text of the exit point field. The help text tells you whether the field is a Java enabled exit point. It also gives you the name of an EB.API record.



3. View the EB.API record indicated in the help text to discover the name of the hook method to implement in Java.

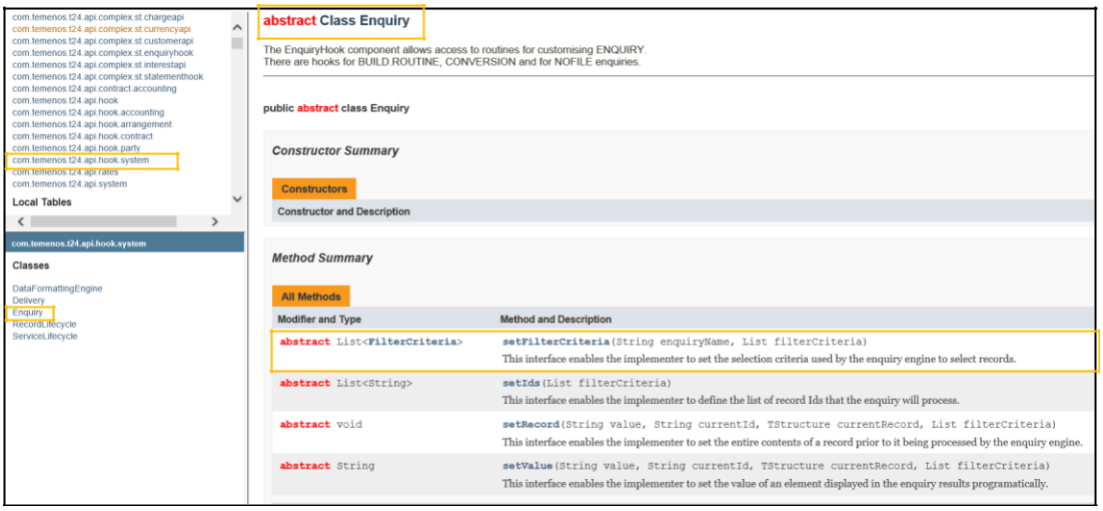


4. View Javadoc to understand the method signature and the abstract class to which it belongs. You can download Javadoc from the customer support (TCSP) and partner support (TPSP) portals.

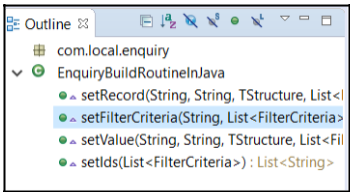
    a. Extract `T24.javadoc.jar` from the zip file.

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

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



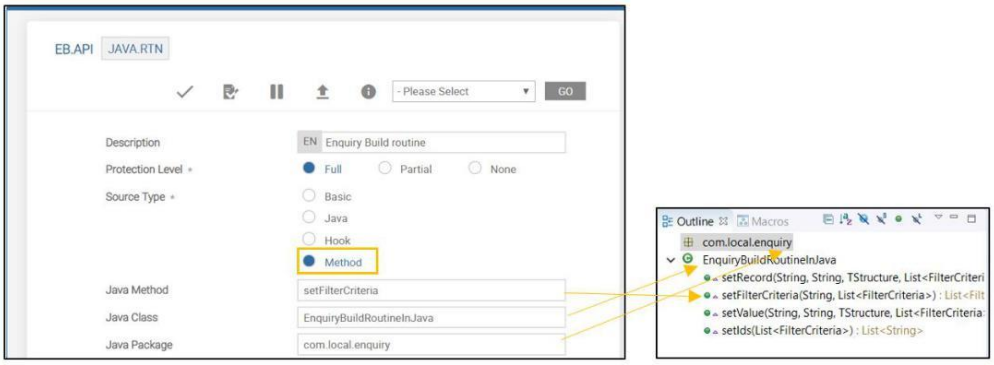
5. Create a Java project in Design Studio to generate the method outlines of the abstract class. Implement the method `setFilterCriteria()` in the Java class that extends Enquiry abstract class.



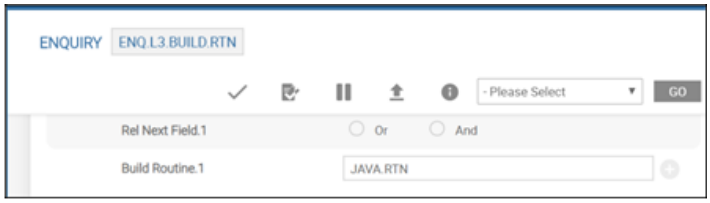

```java
package com.local.enquiry;

import java.util.List;

import com.temenos.api.TStructure;
import com.temenos.api.exceptions.T24CoreException;
import com.temenos.t24.api.complex.eb.enquiryhook.FilterCriteria;
import com.temenos.t24.api.hook.system.Enquiry;

public class EnquiryBuildRoutineInJava extends Enquiry {

    public void setRecord(String value, String currentId, TStructure currentRecord,

    @Override
    public List<FilterCriteria> setFilterCriteria(String enquiryName, List<FilterCriteria> filterCriteria) {

        if (filterCriteria.get(0).getFieldname().equals("SHORT.NAME")) {
            throw new T24CoreException("SHORT.NAME. Field not allowed for selection", "EB-ERROR.SELECTION");
        } else {
            FilterCriteria criteria = new FilterCriteria();
            criteria.setFieldname("SECTOR.CODE");
            criteria.setOperand("LK");
            criteria.setValue("1...");
            filterCriteria.add(criteria);
        }
        return filterCriteria;
    }
```

6. Place the Java implementation in a library and make it available in the JBoss classpath.

7. Create an EB.API record and enter details of the java implementation. Set **Source Type=Method**, and provide the name of the overridden method, its class and package name.

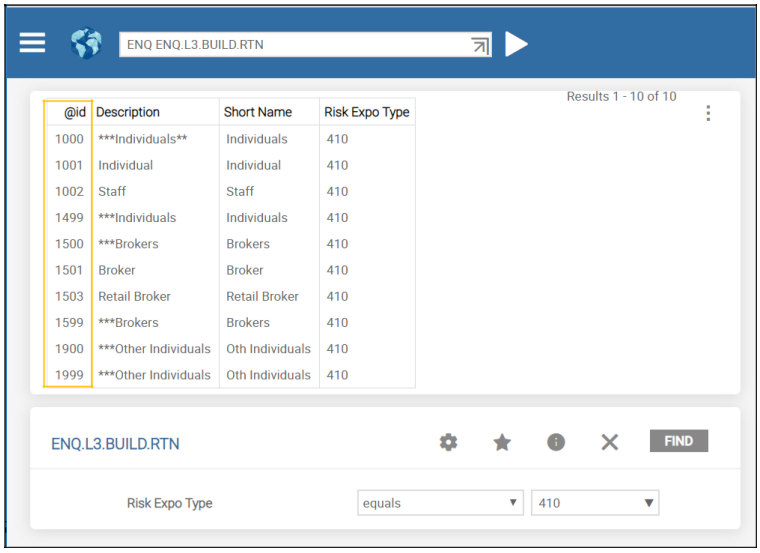


8. Attach the EB.API record to exit point field. For example, Build routine in ENQUIRY.



9. Launch the enquiry. You will see an error when the SHORT.NAME field is used as the selection criteria.

10. Launch the enquiry with selection criteria. The additional selection criteria of “SECTOR.CODE LK 1…” in the Java build routine is applied to the result set.



You can write a conversion routine, similar to the build routine, to modify field values.

4 Packages in Javadoc for L3 API

The APIs for L3 developers are grouped as packages. There are four main levels.

Level	Description
`com.temenos.t24.api.`**`hook.`**`*`	Contains hook related API information like the abstract classes to extend for the exit point hook routines and details of all the class methods to override. **Example** `com.temenos.t24.api.hook.system` package is a group of system hook APIs for enquiry, version, service etc. hook routines.
`com.temenos.t24.api.`**`complex.`**`*`	Contains complex class types to represent objects which are not necessarily records in a table. A complex class type groups different types of related information together. **Example** `com.temenos.t24.api.complex.eb.enquiryhook` package contains a class by name `FilterCriteria`. The class holds the enquiry selection criteria field information and has methods `getFieldName()`, `getOperand()` and `getValue()` to access the same.
`com.temenos.t24.api.`**`records.`**`*`	Contains record class types to represent T24 Transact table records **Example** `com.temenos.t24.api.records.account` package contains `AccountRecord` class with getters, setters and other methods to access and modify the single-value and multi-value fields in ACCOUNT application. Associated multi-value set become individual classes in the package. For example, `EventClass`.

	 
`com.temenos.t24.api.*`	Excluding the above three packages, contains Java wrapper APIs for T24 Transact standard APIs. **Example** `com.temenos.t24.api.rates` package contains a `Charge` class with methods to calculate charge amount, commission amount, tax amount etc., in both local and deal currency for many types of transactions like loans and deposits.

5 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, `EB_TemplateHook.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\EB_TemplateHook.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.