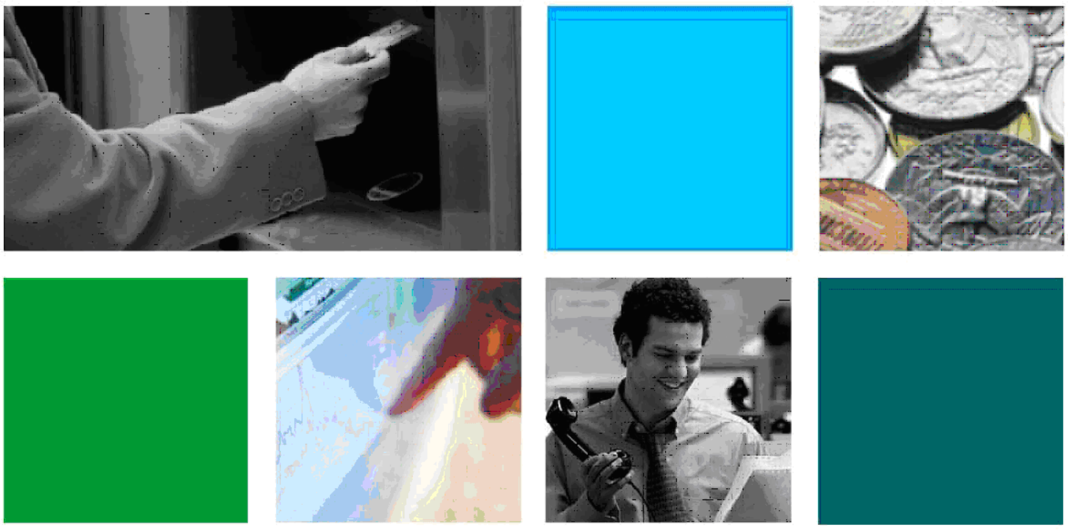
experience
solutions
results



TI Plus

Customisation Guide

2.3

Copyrights and Trade Marks

Contents

Preface - Customisation

This Guide describes how to add extra data fields to standard TI by using Customisation facilities   This is not an end-user task and so this document is targeted at technical consultants and bank staff It assumes that you are familiar with the basic principles of trade finance and with TI Plus itself. Experience of programming is a requirement for understanding the more technical parts of the document.

The Guide contains the following chapters and appendices:

Chapter 1	Provides an overview of customisation functionality and the architecture and technology involved. It also introduces some terminology used throughout this Guide.
Chapter 2	Identifies the prerequisites for using TI Plus's customisation facilities, and explains how to install the customisation software and create the customisation definition tables, which are needed before you can start a customisation project.
Chapter 3	Explains how to start the Customisation Editor and use it to set up new static data tables and add extra fields to tables.
Chapter 4	Explains how to extend the TI Plus interface to allow the extra fields to be incorporated into the TI Plus work flow.
Chapter 5	Explains how to build a project, once you have completed defining the tables, extra field and associated user interfaces it will use.
Chapter 6	Covers the call back and exposed methods that support customisation.
Chapter 7	Provides some general advice on writing code to customise TI Plus.
Chapter 8	Explains how to move your customisation into a live environment.
Chapter 9	Discusses key points relating to extending postings generated by the system.
Chapter 10	Explains how to switch the customisation functionality off, and how to remove customisation from a TI Plus installation.
Appendix A	Provides reference information.
Appendix B	Explains how to update a customised TI Plus database.
Appendix C	Contains information on changing your own extra fields in a subsequent project.
Appendix D	Provides an invoking chart - TI Plus to the Java code.
Appendix E	Provides an example of how to import customer data from back office into TI Plus.

Some Document Conventions

This style is used to indicate text that contains information of more than usual significance, such as warnings or hints on using the software.

Italics are used to distinguish terms which have a specific meaning in the context of this document when they are introduced.

The programming language used by TI Plus is Procedure Definition Language, which is abbreviated to PDL in this Guide.

Chapter 1 Introduction

This chapter provides an overview of customisation functionality and the architecture and technology involved. It also introduces some terminology used throughout this Guide.

TI Plus's Customisation Functionality

TI Plus's customisation facilities allow you to define additional data fields, referred to in this Guide as *extra fields*, as extensions to the following tables in the TI Plus database:

- Master
- Event
- Customer
- TICustomerDetails
- Postings

This is done using the central component of TI Plus's customisation facilities, the Customisation Editor.

You can also add new static data tables to the TI Plus database. The extra fields and tables can be viewed using the TI Plus data model viewer.

You cannot use customisation functionality to extend part-payment, foreign exchange or departmental limit details.

So that users will be able to access the extra fields that you have defined for the Master, Event, Customer and TICustomerDetails tables, you can extend the TI Plus user interface to include additional html panes with customised layouts. However, for extra fields held in new tables that you have defined, a standard user interface is provided by the TI deployment; you cannot alter this user interface.

The data held in extra fields (as well as that held in existing data fields) can be included in TI Plus documents, and used in postings sent to back office systems integrated with TI Plus. It can also be used in TI Plus rules (see the *TI Plus System Tailoring User Guide*), thus allowing the standard transaction processing functionality delivered with TI Plus to be modified through customisation. The basic code produced by the Customisation Editor is sufficient to enable simple TI Plus customisations (for the input and maintenance of extra fields) without any additional development being required. More complex customisations, involving either the extension of postings or the retrieval of TI Plus's own transaction data to generate or validate extra fields, can be implemented by extending the generated code.

The Architecture of a Customised System

A customised system has the following components in addition to those delivered with TI Plus:

- AN XML schema to hold customisation definitions needed by the TI Plus runtime software
- Database tables to hold any extra tables introduced by your customisation
- Extra fields added to a specific set of existing tables
- HTML, XML and deployed Java libraries containing artefacts, frameworks and classes to implement the user-defined user interfaces that form part of the customisation

- Other functions, provided as part of the TI Plus deployment, to support both the customised user interface and a static data editor which can be used to maintain the contents of new tables introduced by customisation

At Run-time

Once the customisation XML schema has been generated and deployed to the TI Plus application server, the Customisation objects will be called as required as the TI Plus application is run.

On initial start up, the TI run-time will check for the presence of the Customisation XML schema. If found then the unit is treated as customised.

During transaction processing, calls into the Customisation objects are automatically made in the following circumstances:

- When accessing Customer and/or TI Customer details
- When accessing Master or Event data
- If you press the Refresh button on the html page
- If you press the OK or Pend buttons on the html page for Extra Data, thus saving data that may include data in extra fields
- If a posting is generated and extra fields have been defined for the Posting table
- If you press the Extra Data html button when the system option CustomisedDataInSeparatePane is used in masters and events

More Sophisticated Customisations

TI Plus's customisation facilities also support more sophisticated customisation requirements, for example to calculate extra fields from standard TI Plus fields or to allow postings to be extended. This is done by enhancing the code generated by the Customisation system.

Master Records and Events

Each business transaction that TI Plus processes, such as a letter of credit, is based around a *master record*. The master record is used to store the current status of a trade finance transaction and includes, for example:

- The current amount of the transaction
- Current liability balance
- Date of expiry
- Whether the record is active or inactive - for example, whether it has expired
- Other data of a transaction-specific nature

All trade finance transactions move through a defined life-cycle. A typical life-cycle would involve, for example:

- Creation or issuance
- Amendments
- Payment
- Expiry
- Book off

Each stage in the life-cycle is handled as an event within the life of the transaction. The first event creates the master record for the transaction. Each modification to the record, such as an amendment or a payment, is achieved by adding an additional event, and expiry and book-off are typically the last events that affect the content of the master record.

In a customised system, in order for the copying of data between event and master tables to be performed, you must specify the relationship between any extra fields you add at event level, and the equivalent extra fields you add at master level. You can do this using the Customisation Editor, which will then generate the code necessary to perform the required copying.

Linked Events

All the data fields in an event can be categorised as being event-level or master-level.

- Event-level data consists of data fields which are used within the particular event only
- Master-level data consists of data fields of a transactional nature, which are copied back to the master record at appropriate points during the data processing

Where events are linked, it is important that the customised data is defined against both events to ensure it is available to be carried forward. Examples of linked events include:

- A Claim Received event, followed by Outstanding Claim event
- An Amendment event, followed by Beneficiary Response to Amendment event

Chapter 2 Installing the Customisation Facilities

This chapter identifies the prerequisites for using TI Plus's customisation facilities, and explains how to install the customisation software and create the customisation definition tables, which are needed before you can start a customisation project.

Prerequisites

In order to customise TI Plus you will need the following:

- The Customisation Editor
- Java run-time version 2.0_05 or above
- The client software for the database to run scripts
- The standard data file staticdefs.txt. During installation, a registry entry is set up with its location

In order to use the event field mechanism to access TI Plus data from the code, you must also have access to TI Plus.

The Customisation Editor will run on a PC that fully meets the TI Plus client PC specification. However, Misys recommend that the PC be equipped with a processor running at 400 MHz or faster.

Installing the Customisation Facilities

The Customisation Editor has a separate and freestanding installation. You must install the version that exactly matches the run-time TI you will be deploying on. This installation package includes all required program and help files for you to use the Customisation editor.

The staticdefs.txt file (which defines the products and tables that can be customised) is also installed by this package. When the Customisation Editor is run, the program searches for this file using a registry key shown below. The named text file should not be moved unless this registry key is also changed. The file itself should never be edited directly as it has a syntax and content that is specific to a release of TI Plus and to the Customisation Editor.

To find the registry entry, look in:

HKEY_LOCAL_MACHINE\SOFTWARE\MKI\CustomiseTI\Settings

for an entry:

```
StartFile d:\enigma\bin\staticdefs.txt
```

Chapter 3 Creating New Tables and Extra Fields

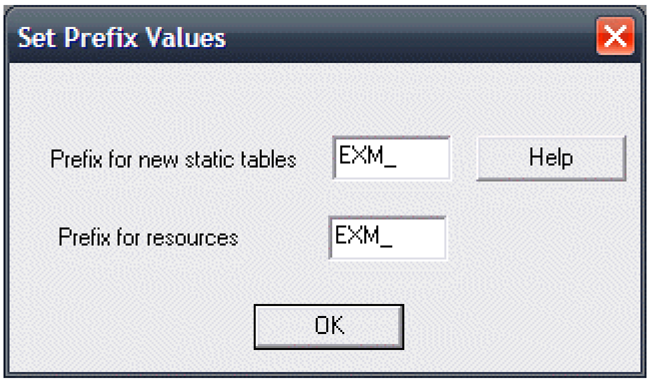
This chapter explains how to start the Customisation Editor and use it to:

- Open and name your project
- Set up new static data tables
- Add fields to these tables

Once you have done this, you will need to build your customisation.

Starting the Customisation Editor and Creating a Project

The Customisation Editor is opened from the TI Plus group on the Start menu. In the window that appears select File|New. The Set Prefix Values window is displayed.



Prefix for new static tables: EXM_

Prefix for resources: EXM_

Use the two fields to enter prefixes to be used when creating the names for the new tables and resources your customisation will use. These prefixes are used to avoid name clashes with existing TI Plus file and resource names.

You can leave these fields blank, but Misys do not recommend this. If you do leave these fields blank, take care to ensure that any new tables you create do not have the same names as existing standard TI ones.

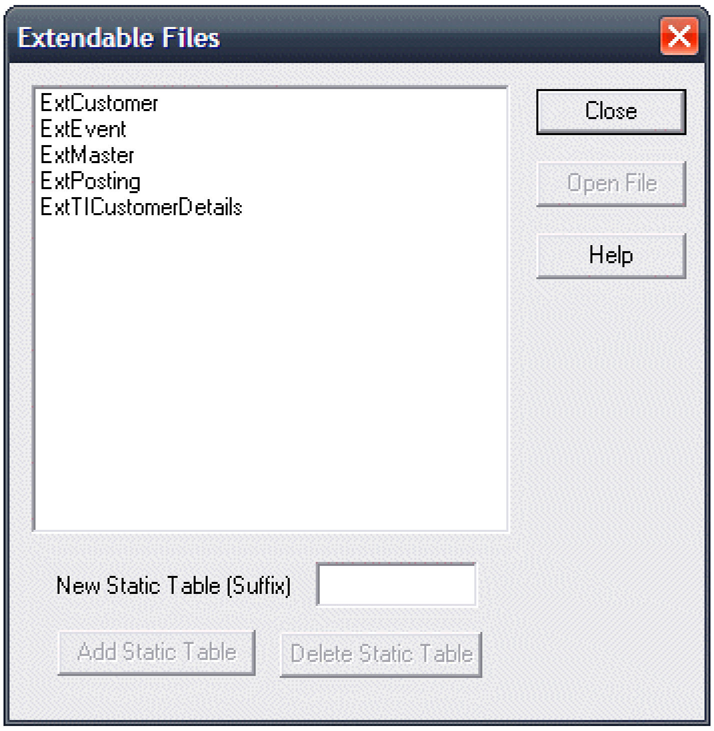
Once you press OK in this window a new, empty project is created. Save the project as a .txt file using the File|Save option.

The resulting .txt file should never be edited manually using a text editor, since it has a syntax, layout and content controlled and owned by the Customisation Editor.

Misys recommend that you save the project in a new and separate sub-directory set up for the purpose, as this will help you to keep all parts of the customisation together in one place. This will be useful if you intend to develop several separate and different customisations over time.

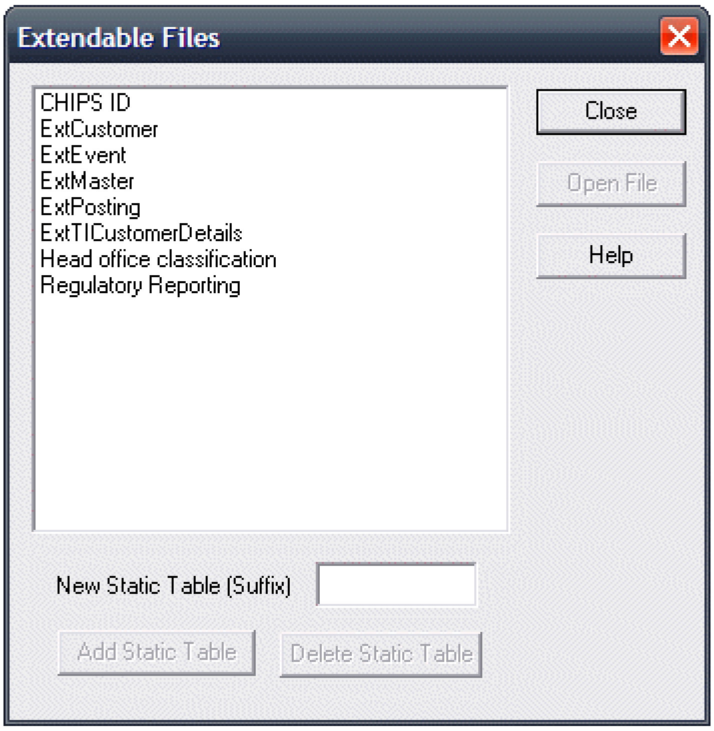
Setting Up New Tables

Once you have created a project, additional menu options are enabled. One of these - the Edit|Tables menu option - allows you to create new tables. When you select this menu option the Extendable Files window is displayed.



This window lists the five tables used to extend the existing TI Plus tables Customer, Event, Master, Posting and TICustomerDetails.

To create a new table, enter an upper case, alphanumeric value to identify the table into the New Static Table (Suffix) field, then press the Add Static Table button. The Customisation Editor creates a unique ID for the new table by appending the value you enter here to the prefix for tables you entered when you first created the project. The resulting table ID cannot be more than ten characters long. The new table is added to the list displayed.



CHIPS ID
ExtCustomer
ExtEvent
ExtMaster
ExtPosting
ExtTICustomerDetails
Head office classification
Regulatory Reporting

New Static Table (Suffix)

If you entered an empty string as the table prefix, the value you enter in the New Static Table (Suffix) field must begin with a letter.

You can remove a table from this list by selecting it and then pressing the Delete Static Table button. The table is removed immediately. You cannot delete any of the pre-existing customisation tables delivered with TI Plus (such as ExtCustomer) in this way.

Adding Extra Fields to a Table

To add extra fields to a table, select the table in the Extendable Files window, then press the Open File button. The Columns window is displayed.

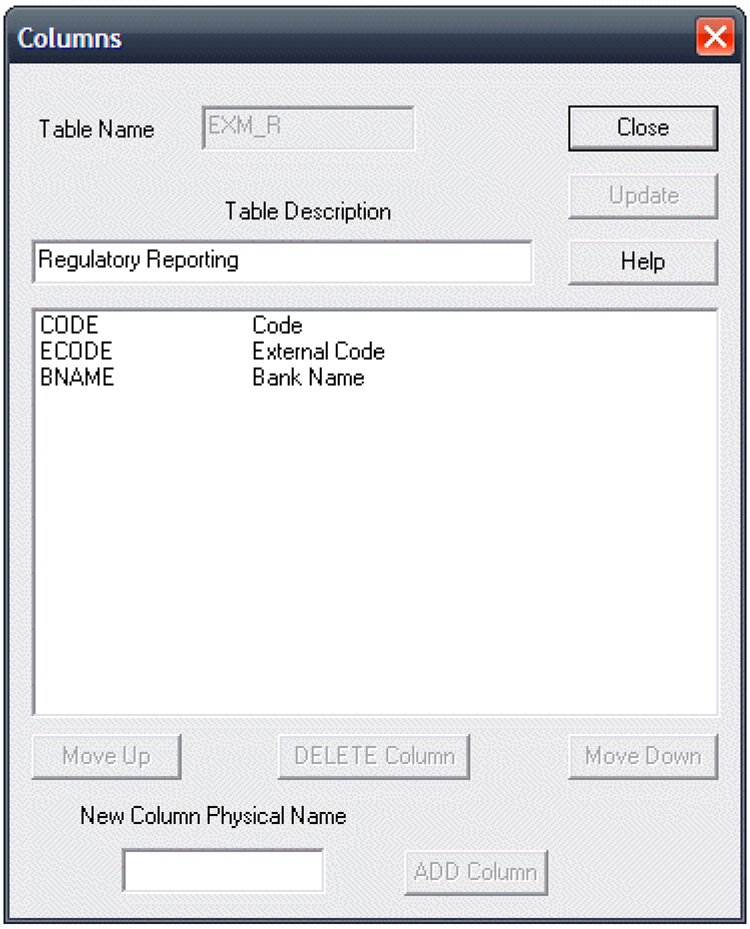


Table Name: EXM_R

Table Description: Regulatory Reporting

CODE	Code
ECODE	External Code
BNAME	Bank Name

New Column Physical Name

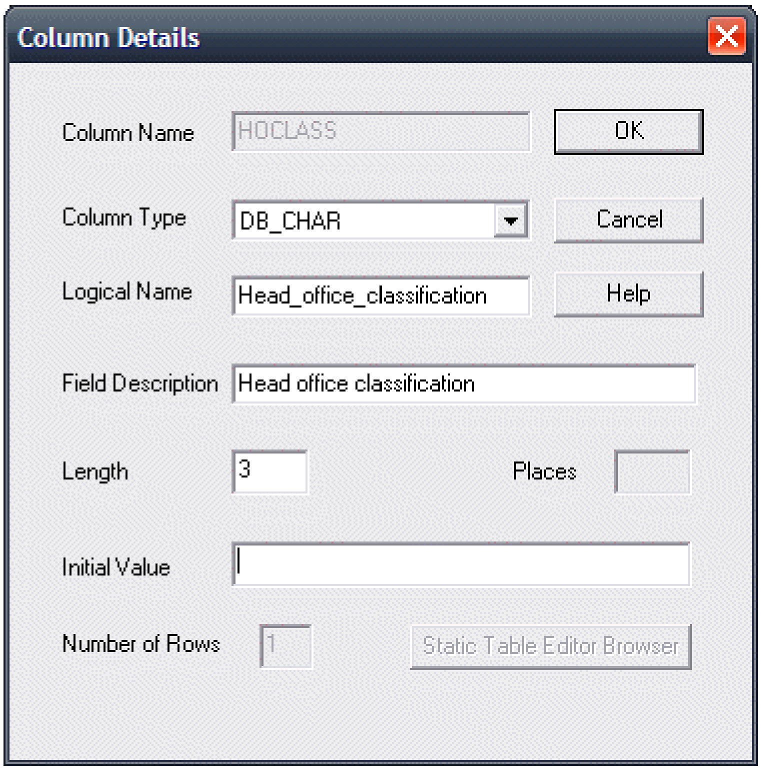
The Table Description field shows the name constructed for the table by the Customisation Editor. For files you have created yourself, overtype this value to give it a meaningful name.

To add an extra field to the table, enter a name for the field (alphanumeric, up to eight characters long) into the New Column Physical Name field, then press the Add Column button. The Column Details window is displayed next, but the number of fields on it will differ according to the table you are working with, as follows:

- For any new tables you have defined, or for the tables ExtCustomer, ExtTICustomerDetails or ExtPosting, the Column Details screen allows you to enter the basic details about the field (see page 19)
- For any new tables you have defined, their expected purpose is to provide a list of values in drop lists or browsers. For this purpose such tables must be given a first column that is a unique business value. If any extra static tables are not used as browser or combo box sources then a non-business value that is made unique is still required. Other columns in such tables can be any required business values
- For ExtEvent and ExtMaster tables, the Column details allows you to enter the same basic details, but it also allows you to define the specific products and events where the extra field is required (see page 20)

Entering Basic Details for an Extra Field

For any new tables you have defined, or for the ExtCustomer, ExtTICustomerDetails or ExtPosting tables, the window illustrated below is displayed.



Column Name	HOCLASS
Column Type	DB_CHAR
Logical Name	Head_office_classification
Field Description	Head office classification
Length	3
Places	
Initial Value	
Number of Rows	1

This window allows you to define the basic attributes of an extra field that you are adding to a table. If the table is an extra static table the Static Data Editor Browser button (see page 23) is enabled and allows you to set the link between the current field and a field of the same type on another extra static table.

The following table describes what to enter into each of the fields in the Column Details window for any new tables you have defined, or for the ExtCustomer, ExtTICustomerDetails or ExtPosting tables:

Field	What you should enter
Column Name	Displays the name given to the extra field in the previous window.
Column Type	Select the type of field. The fields available for input vary, depending on what you enter here.
Logical Name	Used to provide a longer meaningful name for the extra field.
Field Description	Enter a text string to be used as the label for the extra field in the user interface used to enter data into it.
Length	Where appropriate, use this field to define the length of the extra field.
Places	Where appropriate, use this field to define the number of decimal places the extra field will have.
Initial Value	You can specify the value to which the column will be set on initialisation. This value must be compatible with the data type of the column.
Number of Rows	Used to specify the number of text lines required for a field. The entered value will alter the displayed size of the edit box within the generalised static data editor. For other customised data you may draw the user for the field to

Field	What you should enter
	the required shape or size.

Editing an Extra Field

To edit an extra field, first select the Edit|Tables menu option. A dialog appears that lists the names of all the tables that can contain additional fields. Double click on the name of the table you wish to edit.

This displays another dialog box with all the currently defined extra fields for the selected table. You can change the order in which the fields appear by using the Move Up and Move Down buttons.

To edit the field itself, double click on the field name you require to open the Column details window.

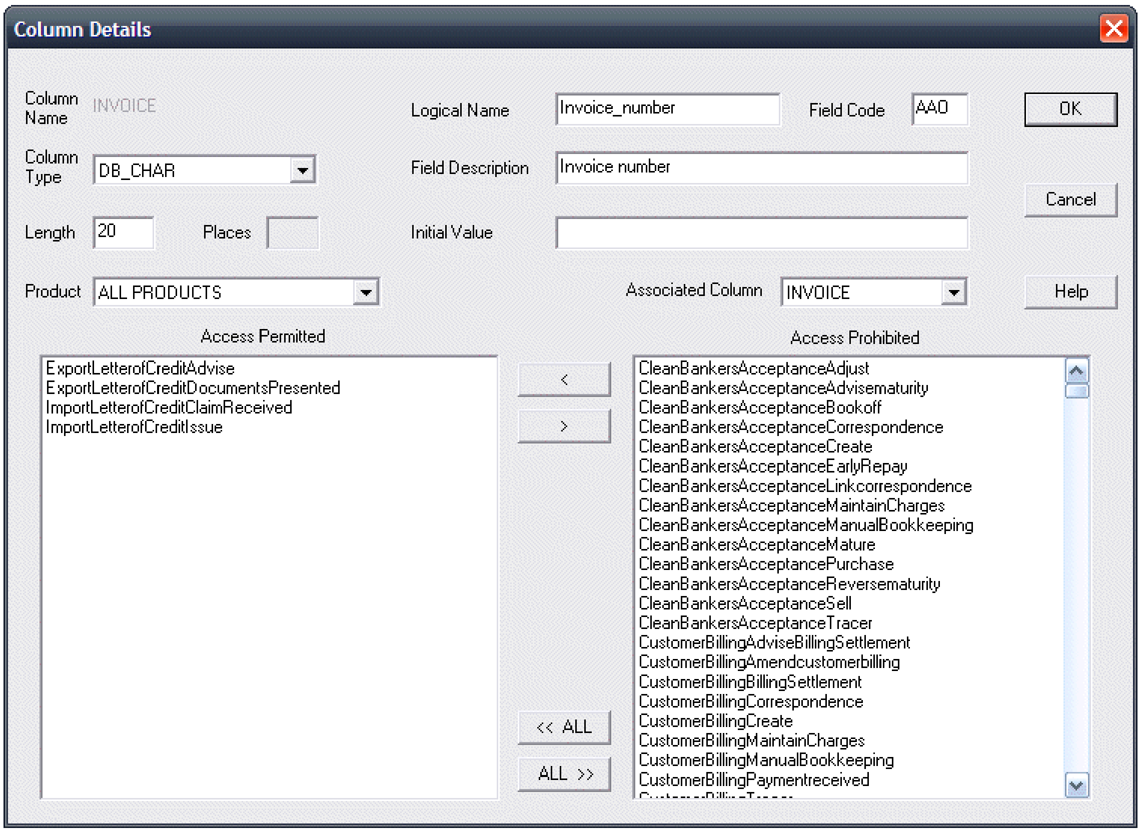
You can now edit the information that has been defined for that field. What you enter in each of the fields on the Column details window for new tables Customer, ExtTICustomerDetails or ExtPosting differs from what you enter for ExtEvent or ExtMaster tables.

Press OK when you have completed your changes and they will be saved.

Entering Details for the Extra Field (in a ExtEvent or ExtMaster Table)

For the ExtEvent and ExtMaster tables, the Columns Details window allows you to enter the basic details of the extra field as well as additional information about the specific products and events where the extra field is required.

Use the fields in the top half of the screen to enter the basic details of the extra field - the table later in this section describes what to enter in all the fields on this window.


Column Details

Column Name: INVOICE
Logical Name: Invoice_number
Field Code: AA0
Column Type: DB_CHAR
Field Description: Invoice number
Length: 20
Places:
Initial Value:
Product: ALL PRODUCTS
Associated Column: INVOICE

Access Permitted:
ExportLetterofCreditAdvise
ExportLetterofCreditDocumentsPresented
ImportLetterofCreditClaimReceived
ImportLetterofCreditIssue

Access Prohibited:
CleanBankersAcceptanceAdjust
CleanBankersAcceptanceAdvisematurity
CleanBankersAcceptanceBookoff
CleanBankersAcceptanceCorrespondence
CleanBankersAcceptanceCreate
CleanBankersAcceptanceEarlyRepay
CleanBankersAcceptanceLinkcorrespondence
CleanBankersAcceptanceMaintainCharges
CleanBankersAcceptanceManualBookkeeping
CleanBankersAcceptanceMature
CleanBankersAcceptancePurchase
CleanBankersAcceptanceReversematurity
CleanBankersAcceptanceSell
CleanBankersAcceptanceTracer
CustomerBillingAdviseBillingSettlement
CustomerBillingAmendcustomerbilling
CustomerBillingBillingSettlement
CustomerBillingCorrespondence
CustomerBillingCreate
CustomerBillingMaintainCharges
CustomerBillingManualBookkeeping
CustomerBillingPaymentreceived

The Product field, the Access Permitted and Access Prohibited fields allow you to set the products and events for which the extra field will be made available.

If you fail to map the products and events that a field are used in, the customisation project will default to the field being mapped to ALL products and events

Having selected the product(s), you set the access for the different events by moving them from the Access Permitted list to the Access Prohibited list (or vice versa): either double-click on the event or select it event and use the > and < buttons, as appropriate. To move all the events from one column to the other, click on the << ALL or ALL >> button.

It is important to restrict the accessibility of fields as much as possible, for the following reasons:

- Interface code is generated for all customisation points where data can be accessed
- Specifying that an extra field is accessible from an interface will cause additional code to be generated, even if it is not accessed
- At runtime the data fields are made available in the interface between the TI Plus application and the customisation code

Field	What you should enter
Column Name	Displays the name given to the extra field in the previous window.
Column Type	Select the type of field. The fields available for input vary, depending on what you enter here.
Length	Where appropriate, use this field to define the length of the extra field.
Places	Where appropriate, use this field to define the number of decimal places the extra field will have.

Field	What you should enter
Product	Select a product for which the extra field will be available, or select ALL PRODUCTS from the drop-down list to make the field available to all products. Depending on what you have selected, the appropriate events are displayed in the Access Prohibited and Access Permitted columns.
Logical Name	Used to provide a longer meaningful name for the extra field.
Field Description	Enter a text string to be used in displays of this field in the View Event Fields user interface.
Field Code	Enter a unique three-character alphabetic code for the extra field. The value you enter here, prefixed by a 'c', is used to access the field in customisation code as well as internally by TI Plus when listing fields available for inclusion in, for example, documents, clauses and rules. If you leave this field blank, the system will generate a field code value here automatically. You can check the field code values set up for your customisation and amend them later (see page 25).
Initial Value	You can specify the value to which the column will be set on initialisation. This value must be compatible with the data type of the column.
Associated Column	Used to specify that pairs of columns (one in the ExtMaster table, the other in the ExtEvent extension table) are to be copied, from one to the other, when an event is created or released. The drop-down list contains the names of master (or event) columns of the same type that are not currently associated. Selecting a member of the drop down list will create the association (this can be done either from the event column or the member column; you do not need to specify the connection for both columns individually). If you change the data type of an extra field, this will reset the association (as the two types must match). Every field defined on ExtMaster must be associated with a field on ExtEvent; otherwise the generated project will be incomplete and the project will not build successfully.
Access Permitted	Depending on what is selected in the Product field, this panel lists all the events for either a specific product or for all products. Use this panel to specify the events for which the extra field may be used. Move any events for which the field may not be used to the Access Prohibited list.
Access Prohibited	Depending on what is selected in the Product field, this panel lists all the events for either a specific product or for all products. Use this panel to specify the events for which the extra field may NOT be used. Move any events for which the field may be used to the Access Permitted list. If you do not specify which products and events can use the extra field, it will initially be accessible everywhere (as a defined field must be accessed somewhere in the customised application). This is not recommended.

When you have completed these fields and clicked OK, the new extra field is listed in the Columns window.

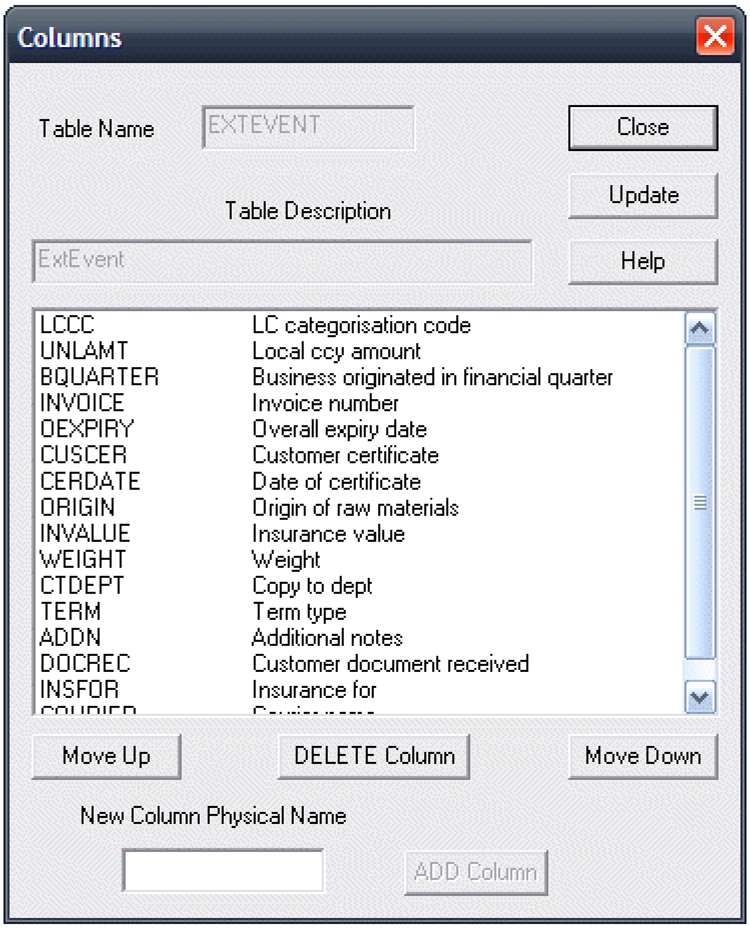


Table Name: EXTEVENT

Table Description: ExtEvent

LCCC	LC categorisation code
UNLAMT	Local ccy amount
BQUARTER	Business originated in financial quarter
INVOICE	Invoice number
OEXPIRY	Overall expiry date
CUSCER	Customer certificate
CERDATE	Date of certificate
ORIGIN	Origin of raw materials
INVALUE	Insurance value
WEIGHT	Weight
CTDEPT	Copy to dept
TERM	Term type
ADDN	Additional notes
DOCREC	Customer document received
INSFOR	Insurance for

New Column Physical Name

As you build up the list, you can use the Move Up and Move Down buttons to move a selected extra field one place up or down in the list. The Delete Column button permits you to delete a selected extra field (provided that it has not been attached to a user interface control. In this case, you must remove it from the user before deleting it here).

Deleting an Extra Field

You can delete an extra field as follows.

First, select the Tables option from the edit menu.

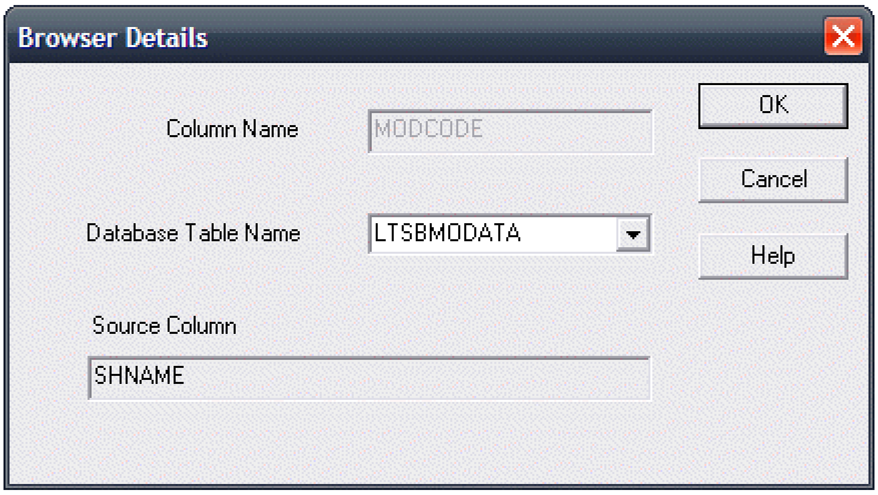
From the dialog box that lists the names of all the tables that contain additional fields, double click on the name of the table from which you wish to delete a field.

This brings up another dialog box with all the currently defined extra fields for the selected table. Select the field name you wish to delete, and click on the DELETE Column button.

Linking an Extra Field to a Field in an Extra Static Table

If the extra field you are specifying in a new table is going to contain data taken from another extra static table, press the Static Table Editor Browser button to set the link between the current field and a field of the same type in the other table. (This button is disabled when setting up TICustomerDetails and Customer columns, because in these cases the browsers are defined during the user interface construction instead of in the table definitions.)

The Browser Details window appears.



Column Name: MODCODE

Database Table Name: LTSBMODATA

Source Column: SHNAME

Use the Database Table Name field to identify the other extra static table, and the Source Column field is set to the first column of that table to identify the extra field within that table to which the one you are currently setting up will be linked.

Extra Fields on the Customer and TICustomer Entities

This section identifies some constraints on creating extra fields for customer data.

The Customer entity contains details normally provided by your back office system. You can create extra fields on the Customer table only if you are running TI Plus standalone, or if a bespoke interface has been created to TI Plus from a back office system other than Equation or Midas.

- The Customer customisation provides the following functionality:
- A jsp pane for entering and viewing the customer data in the GUI
- The customisation methods required for the above pane
- Event field code processing

The TICustomer entity is TI Plus's own customer entity and contains data not provided from the back office. If you are running TI Plus integrated with Equation or Midas, or with another back office system in a way that prevents customisation of the Customer entity, you can use the TICustomer entity instead to create extra fields for customers.

The TICustomer customisation provides the following functionality:

- A jsp pane for entering and viewing the TI Plus customer data in the GUI
- The customisation methods required for the above pane
- Event field code processing

Static Data Maintenance

The maintenance of data in extra static tables is handled within standard TI activated by the deployment of the customisation. Each extra static table is given a hotlink at the bottom of the other standard static functions of TI.

Checking and Amending the Generated Field Codes

An event field code is a four-character (maximum) identifier used as a means of labelling data within the TI Plus application. By accessing data using a code rather than the actual variable name, the underlying data structures can then be changed while leaving external data access untouched. Event field codes provide a layer of abstraction to the data. Event field codes are used by TI Plus's document and posting definitions.

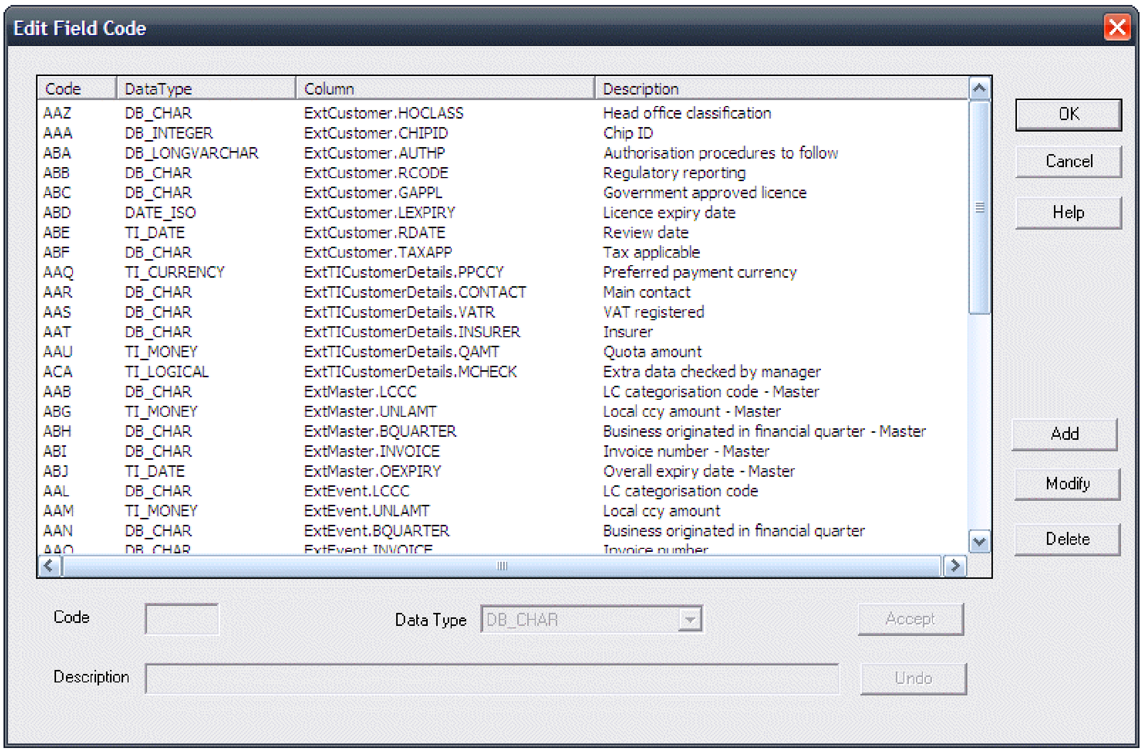
Most standard TI Plus master, event and customer fields (and many derived fields) are available by field code, and the customisation field codes are added to the list of those available.

As fields are added to the customisation using the Edit|Tables menu option, each needs to be allocated a unique three-character field code. You can enter these manually; otherwise they will be allocated automatically by the Customisation Editor.

You can review and amend these field codes, if required, using the Customisation Editor's Edit|Field Codes menu option, regardless of how they were derived.

The field codes are made available to the TI Plus application by being automatically defined inside the customisation objects. When this happens, the field codes are prefixed by the letter 'c' – for example, field code AAA becomes cAAA, and it is this four character field code that is used within TI Plus.

The Edit Event Field window can be accessed from the Edit|Field Codes menu option and allows you to view or amend field codes.



Code	DataType	Column	Description
AAZ	DB_CHAR	ExtCustomer.HOCLASS	Head office classification
AAA	DB_INTEGER	ExtCustomer.CHIPID	Chip ID
ABA	DB_LONGVARCHAR	ExtCustomer.AUTHP	Authorisation procedures to follow
ABB	DB_CHAR	ExtCustomer.RCODE	Regulatory reporting
ABC	DB_CHAR	ExtCustomer.GAPPL	Government approved licence
ABD	DATE_ISO	ExtCustomer.LEXPIRY	Licence expiry date
ABE	TI_DATE	ExtCustomer.RDATE	Review date
ABF	DB_CHAR	ExtCustomer.TAXAPP	Tax applicable
AAQ	TI_CURRENCY	ExtTICustomerDetails.PPCCY	Preferred payment currency
AAR	DB_CHAR	ExtTICustomerDetails.CONTACT	Main contact
AAS	DB_CHAR	ExtTICustomerDetails.VATR	VAT registered
AAT	DB_CHAR	ExtTICustomerDetails.INSURER	Insurer
AAU	TI_MONEY	ExtTICustomerDetails.QAMT	Quota amount
ACA	TI_LOGICAL	ExtTICustomerDetails.MCHECK	Extra data checked by manager
AAB	DB_CHAR	ExtMaster.LCCC	LC categorisation code - Master
ABG	TI_MONEY	ExtMaster.UNLAMT	Local ccy amount - Master
ABH	DB_CHAR	ExtMaster.BQUARTER	Business originated in financial quarter - Master
ABI	DB_CHAR	ExtMaster.INVOICE	Invoice number - Master
ABJ	TI_DATE	ExtMaster.OEXPIRY	Overall expiry date - Master
AAL	DB_CHAR	ExtEvent.LCCC	LC categorisation code
AAM	TI_MONEY	ExtEvent.UNLAMT	Local ccy amount
AAN	DB_CHAR	ExtEvent.BQUARTER	Business originated in financial quarter
AAO	DB_CHAR	ExtEvent.INVOICE	Invoice number

When you select an entry in this list, the associated code, description and data type are displayed and enabled in the fields at the bottom of the window. You can overtype the code and description. If you do so, press the Accept button to save your changes, or the Undo button to abandon them.

If you press the Add button, these three fields are enabled for you to enter information to define a new extra field. If you do so, press the Accept button to save the information and enter the new extra field to the list of those displayed.

You can also delete an item from the list by selecting it and pressing the Delete button.

Field codes are available with the application for inclusion in rules, clauses and documents. They must be given a full description using this screen to ensure that a user will pick the correct field. This is especially important where a field is held both on events and on the master table. In this case it is recommended they are given different descriptions (perhaps with "(Master)" appended.

Every field defined in the customisation MUST be given an event field code, or run-time errors will be caused

Chapter 4 Customising the TI Plus User Interface

The TI Plus application allows you to build your own user interfaces to provide access to extra fields. The customisation objects are used to implement the customisation.

There are two sets of functions:

- User exit functions, which send out requests from TI Plus to exposed methods in the customisation object classes
- Exported methods in TI Plus, which service call back requests the customisation object classes
- Each business transaction that TI Plus processes, such as a letter of credit, is based around a master record that contains information about the transaction. This includes the current status of a trade finance transaction, as well as the current amount of the transaction, party names, expiry and book-off dates.

All trade finance transactions move through a pre-defined life-cycle - typically from creation or issuance, through amendments and payments to expiry and book-off. Each stage in the life-cycle is handled as an event within the life of the transaction.

When you customise your installation of TI Plus by defining extra fields all the functionality relating to a particular area is contained in a set of customisation java library classes. For example, all the processing and dialog for a particular event will be found in a class file for that specific event. Thus, for every event (or master) that can access the extra fields that have been defined, there is a class in the Java source code.

These class files are used to implement the customisation and are deployed on the TI server in the 'WIS' section. Each class calls specific methods according to where it is in the application, and are used to control the transaction processing within your customisation.

Customisation Areas

There are a number of areas that may be customised. These include:

- Event processing and dialogs. The Event extra fields (held on the ExtEvent table) are accessible. You are able to initialise Event extra fields from Master extra fields and update Master extra fields from Event extra fields at appropriate places in the TI Plus work-flow
- Master processing and dialogs. The Master extra fields (held on the ExtMaster table) may be displayed, read-only, in the dialogs
- Customer processing and dialogs. The Customer extra fields (held on the ExtCustomer table) are accessible
- TICustomer processing and dialogs. The TICustomer extra fields (held on the ExtTICustomer table) are accessible
- User-defined event fields
- Documents
- Postings processing. The Posting extra fields (held on the ExtPosting table) are accessible

Customisation Entities

TI Plus's data model is designed using an object-oriented approach. Each object is represented by a specific entity (that is equivalent to a class in C++ or Java). The entity is mapped to a table in a relational database. Each instance of an entity corresponds to a specific record in the table for that entity.

Each of the five main customisation entities has a foreign key that points to the entity instance for which extended data is being held. The following table lists the main customisation entities:

ExtEvent	extends the Event entity
ExtMaster	extends the Master entity
ExtCustomer	extends the Customer entity
ExtTICustomer	extends the TICustomer entity
ExtPosting	extends the Posting, PeriodicChargeAccrualPosting, and FinancingAccrualPosting entities

There is also another customisation entity called ExtAPI which extends the EQ3Posting and EQ3Movement entities. Unlike the main customisation entities, however, the underlying table is an API table that is used to pass postings from TI Plus to the back office. You do not have to define this table, as the customisation software automatically adds each field you add to ExtPosting to the ExtAPI table.

Entity instances also have an integer member each, called 'Initialised', that identifies whether the instance of the extended data has been initialised yet, or not. This enables the object class to determine whether to initialise the instance. When a new entity is created, 'Initialised' is automatically set to zero. If the object class finds that the value of 'Initialised' is 0, then it will initialise the instance and set the value of 'Initialised' to 1.

Customisation Object Classes

The names of these files have prefixes consisting of a product code (usually three characters) followed by an underscore and an event code (or M if for master) and another underscore. For example:

ILC_ISI_ImportLetterofCredit	would be the name for the file for the import letter of credit Issue event
ILC_M_ImportLetterofCredit	would be the name for the file for the import letter of credit master

Exceptions to this naming convention are as follows:

C_C_Customer	the Customer file
T_T_TICustomer	the TICustomer file
P_P_Posting	the Posting file

Defining the User Interface Extensions

A user interface point is a point in the TI application which exits from the TI Plus application, and go to a required object class generated by a customisation project. These allow your users to enter data in the customised fields that have been defined in the tables. To set these up, you need to add user interface controls to the user interface and then associate each control with an extra field.

The following customised data can be accessed using user interface controls:

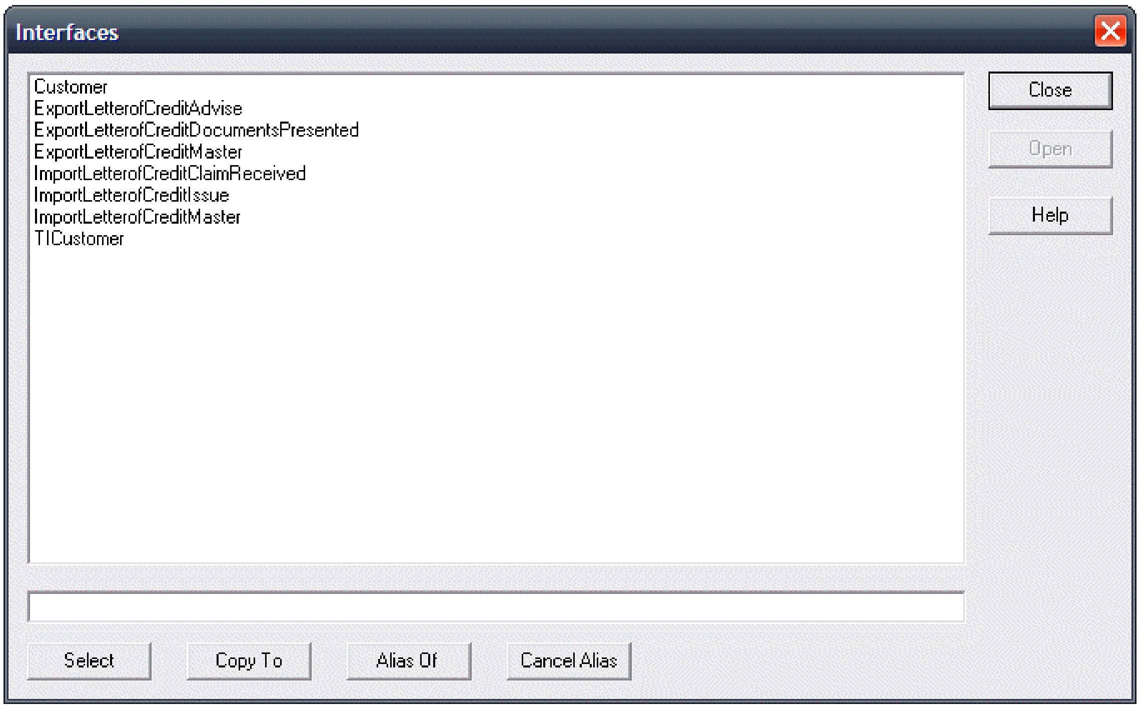
- Extra Customer data
- Extra TICustomerDetails data
- Extra data for an event
- Extra data for a master (view only)

The user interfaces for new tables are all handled by a separate program called static data editor which produces a standard user interface screen.

Note for programmers: the Customisation Editor will create a complete functional customisation. If you wish to hand code the user interface for a customisation point (in other words, to build your own instead of using the generated one), you still need to specify the data access for it.

Locating the User Interface Point

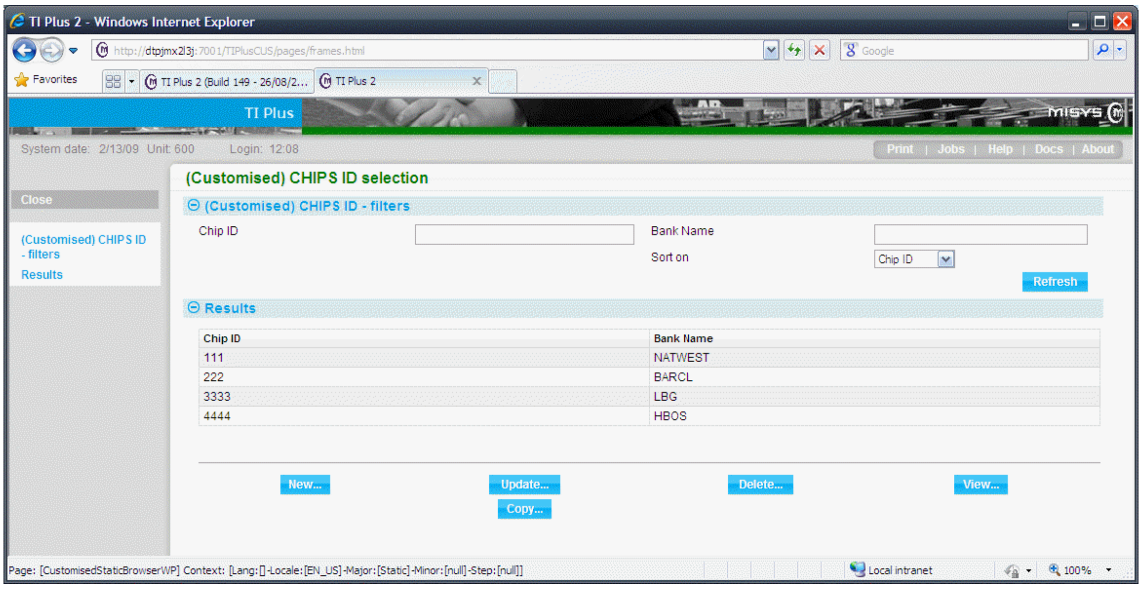
Select the Edit|Interfaces menu option. The Interfaces window appears.



Customer
ExportLetterofCreditAdvise
ExportLetterofCreditDocumentsPresented
ExportLetterofCreditMaster
ImportLetterofCreditClaimReceived
ImportLetterofCreditIssue
ImportLetterofCreditMaster
TICustomer

This lists the user interface points that support the addition of customisation code (because you have a least one field mapped to the interface point). The window allows you to add controls to each user interface.

The interfaces for any extra static tables are not listed here, as they are handled automatically by standard TI. Here is a screenshot of this in TI Plus2 showing the list of records in one extra static table called CHIPSID. The standard function allows the viewing, addition, modification and deletion of data in each extra static table.


TI Plus 2 - Windows Internet Explorer
http://dtpjmx2l3j:7001/TIPlusCUS/pages/frames.html
Google
Favorites
TI Plus 2 (Build 149 - 26/08/2...
TI Plus 2
TI Plus
MISYS
Print | Jobs | Help | Docs | About
Close
(Customised) CHIPS ID - filters
Results
Refresh
New...
Update...
Delete...
View...
Copy...
Local intranet
100%

System date: 2/13/09 Unit: 600 Login: 12:08

(Customised) CHIPS ID selection

(Customised) CHIPS ID - filters

Chip ID		Bank Name	
		Sort on	Chip ID

Results

Chip ID	Bank Name
111	NATWEST
222	BARCL
3333	LBG
4444	HBOS

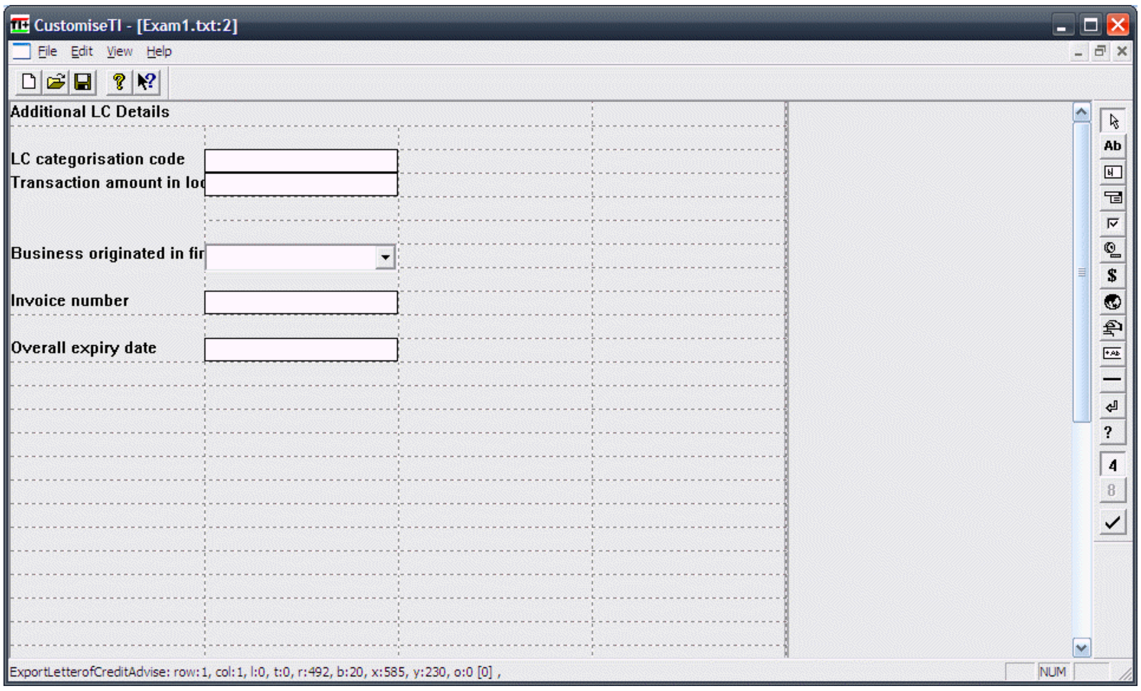
Page: [CustomisedStaticBrowserWP] Context: [Lang:[]-Locale:[EN_US]-Major:[Static]-Minor:[null]-Step:[null]]

For customisation points that relate to events within a particular product, the name consists of the product name followed by the name of the event. To select a value, you can scroll down the list of customisation points, or to speed up the process you can type the first letter of the name in the empty field below the list of names.

Adding a User Interface Control to the Interface

When you select an entry in the list of customisation points (either by double-clicking on it or by positioning the cursor on it and pressing the Select button), the user interface for that interface point is displayed. At first the interface will appear blank, with a dark line around it that shows the limits within which the user interface must be contained.

The following example shows a typical user interface design.



Additional LC Details

LC categorisation code

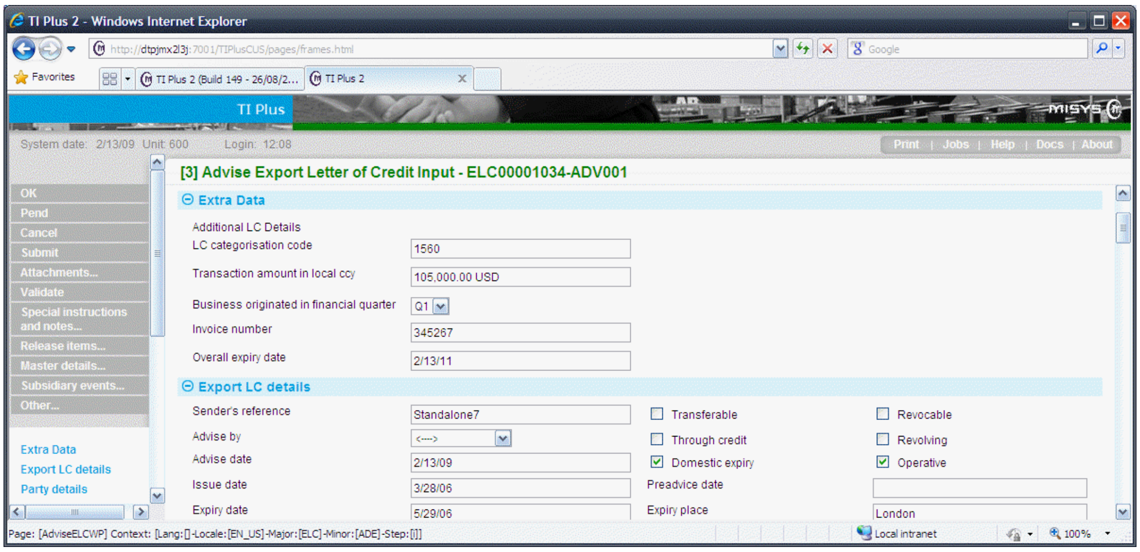
Transaction amount in loc

Business originated in fir

Invoice number

Overall expiry date

ExportLetterofCreditAdvise: row:1, col:1, l:0, t:0, r:492, b:20, x:585, y:230, o:0 [0] , This is NOT how it will look at run-time to the user. This is just a four column representation of the screen layout and will look much better at runtime! Here is what the above field layout looks like at run-time.



System date: 2/13/09 Unit: 600 Login: 12:08

[3] Advise Export Letter of Credit Input - ELC00001034-ADV001

⊖ Extra Data

Additional LC Details

LC categorisation code	1560
Transaction amount in local ccy	105,000.00 USD
Business originated in financial quarter	Q1
Invoice number	345267
Overall expiry date	2/13/11

⊖ Export LC details

Sender's reference	Standalone7	☐ Transferable	☐ Revocable
Advise by	<--->	☐ Through credit	☐ Revolving
Advise date	2/13/09	☑ Domestic expiry	☑ Operative
Issue date	3/28/06	Preadvice date	
Expiry date	5/29/06	Expiry place	London

Page: [AdviseELCWP] Context: [Lang:[]-Locale:[EN_US]-Major:[ELC]-Minor:[ADE]-Step:[I]]

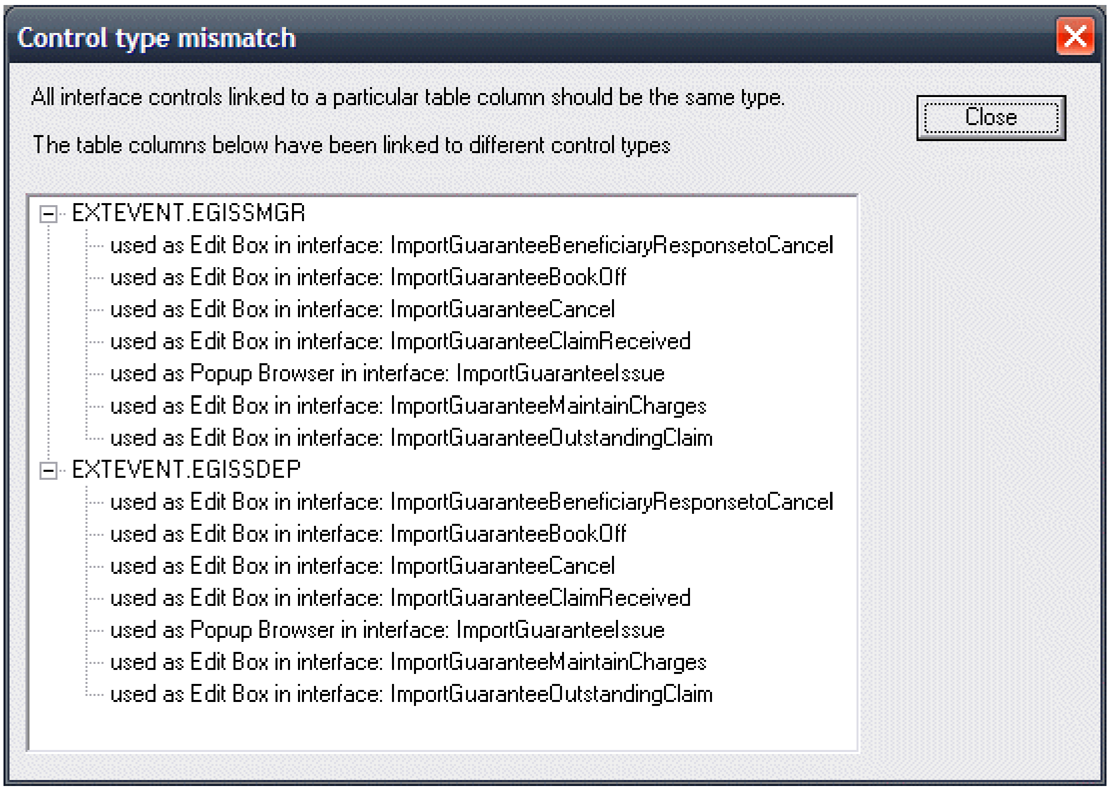
The TI style is to put labels in columns one and three and to put data fields in columns two and four. It is recommended to NOT leave blank lines down the screen between each field. Fields will look best at run-time placed one beside the next down the screen.

Down the right-hand side of the window is a toolbar from which the supported user interface controls may be selected. To find out the name of a user interface item in this toolbar, point and hover on its icon and an information label will be displayed.

Initially, the user interface will be blank. To add an item, first click on the icon in the toolbar to select it, then click the left mouse button at the position on the interface where you wish the item to be shown. (Once you have placed the item, you can re-position it at any time by dragging it, so there is no need to be too exact at this stage.)

You do not have to set tab order as this will be managed by the TI Plus run-time software.

If you have mapped a particular field to more than one event or product, it is important that you consistently use the same control on each screen you paint. A new menu feature has been added to assist this, on the file menu is a function "Check for Control type mismatch" which will display to you any problems. Here is an example:

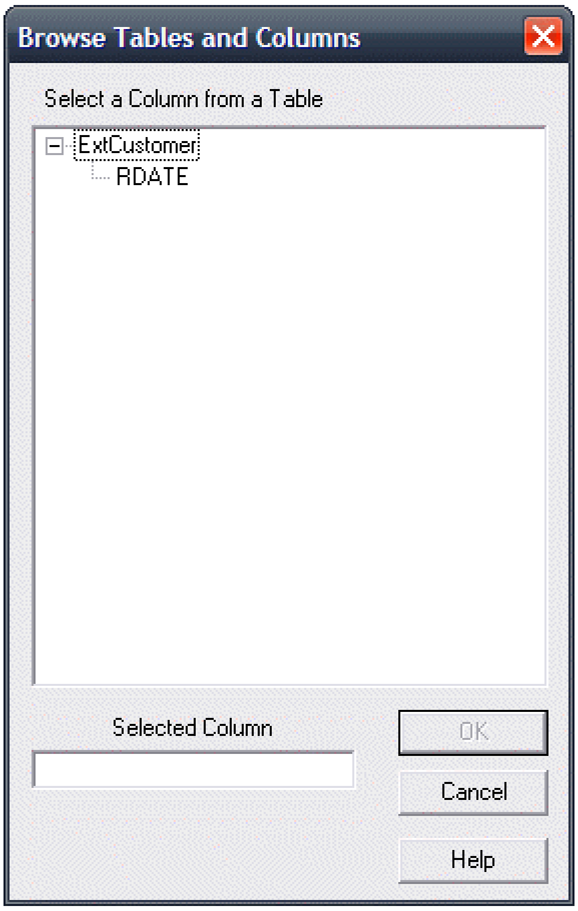


All interface controls linked to a particular table column should be the same type.

The table columns below have been linked to different control types

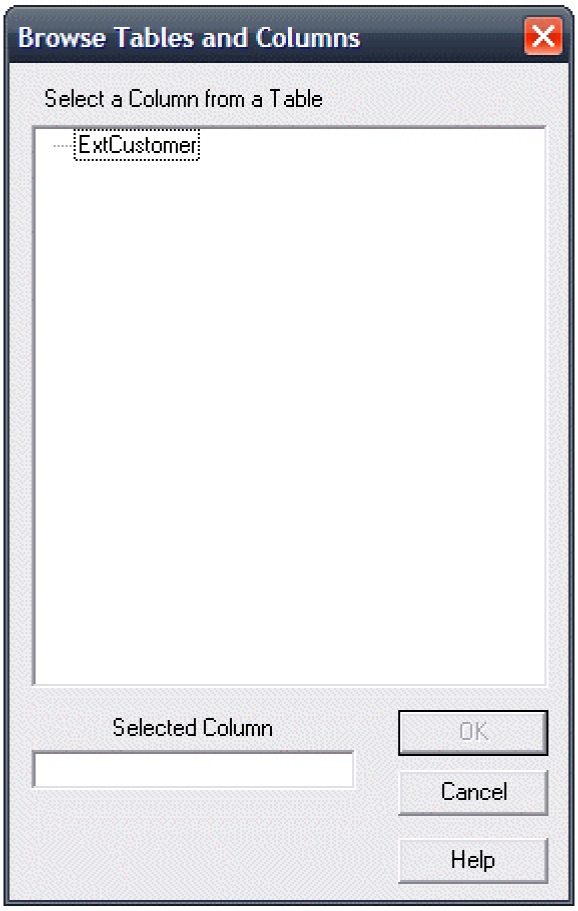
```
EXTEVENT.EGISSMGR
    used as Edit Box in interface: ImportGuaranteeBeneficiaryResponsetoCancel
    used as Edit Box in interface: ImportGuaranteeBookOff
    used as Edit Box in interface: ImportGuaranteeCancel
    used as Edit Box in interface: ImportGuaranteeClaimReceived
    used as Popup Browser in interface: ImportGuaranteeIssue
    used as Edit Box in interface: ImportGuaranteeMaintainCharges
    used as Edit Box in interface: ImportGuaranteeOutstandingClaim
EXTEVENT.EGISSDEP
    used as Edit Box in interface: ImportGuaranteeBeneficiaryResponsetoCancel
    used as Edit Box in interface: ImportGuaranteeBookOff
    used as Edit Box in interface: ImportGuaranteeCancel
    used as Edit Box in interface: ImportGuaranteeClaimReceived
    used as Popup Browser in interface: ImportGuaranteeIssue
    used as Edit Box in interface: ImportGuaranteeMaintainCharges
    used as Edit Box in interface: ImportGuaranteeOutstandingClaim
```

As you click on the interface to position the control, the Browser Table and Columns window appears (except if the control is a label or a group box, as these are not associated with an extra field).

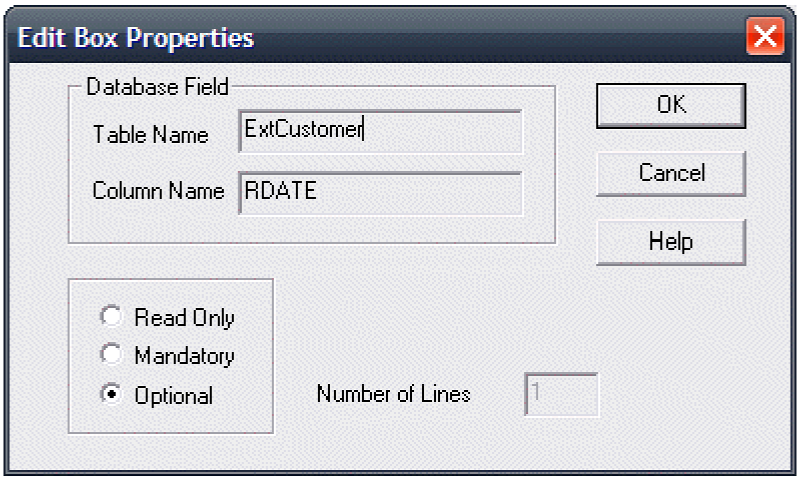


This shows the name of the table with which you are working. If there are fields not yet placed onto the screen a + sign will be shown to the left of the Table. When you click on the + icon the system displays a list of the extra fields defined for the table that have not already been associated with a control and that have a type compatible with that of the extra field. Select the one to which the control is to be attached.

When the table is shown preceded by …. (as shown below), there are no further fields suitable for this control left to place onto the GUI.



Select the appropriate extra field and click OK. A properties box will be displayed so that you can specify the properties that you require.



Database Field

Table Name: ExtCustomer

Column Name: RDATE

( ) Read Only
( ) Mandatory
[✓ Optional]

Number of Lines: 1

You may specify if this field is a read-only field, mandatory or optional.

For large text fields, you can set the number of visible lines allowed as input to this field.

Click OK and the user interface item will now be displayed in the position that you have specified. You can move it and re-size it using the usual Windows drag and drop methods.

At any time you can display a list of all the extra fields from the table with which you are working which have not yet been incorporated into the user interface using the View|Unmapped menu option.

The following table lists the user interface controls you can associate with extra fields.

User interface control		Description
Edit box		Typically used for free-format entries where no predefined entries exist. No user assistance is provided on entries. An edit box can be used for text, date or simple numeric fields. A date field allows the entry of absolute dates. If the associated column is specifically defined as a TI-DATE then the field will also allow the entry of relative dates such as 1M, 3D validated against the local run-date calendar.
Drop Combo box		An edit box with a drop down list to assist choosing a value or, if required, the entry of another value not in the drop-down list. Typically used where there is a small range of values from which the user may select, or may instead enter their own choice of new value.
Check box		A Boolean (logical field) that can indicate a Yes/No value for a database member. Typically used to indicate a true/false value, for example 'Transferable?'.
Money		Used for entering amount and currency. A database member of data type money which means an amount and a currency kept logically together and working exactly the way that amounts are entered in the normal TI Plus user interface.
Currency	$	A drop-down list showing the currencies set up in TI Plus.
Country		A drop-down list showing the countries set up in TI Plus.
Account officer		A drop-down list showing the account officers set up in TI Plus.
Browser	**?**	A browser control with filters from which the end user can choose a single value. The first column of which is the return value

The following icons are used to label fields and improve the user interface, and cannot be associated with extra fields:

Icon		Description
Label	Ab	Used to label fields. Can be linked to TI controls for example Money and Currency.
Horizontal banner	—	Used to improve the user interface by creating horizontal lines for section breaks.
Toggle banner		Used to improve the user interface by creating expandable and collapsible sub sections.
Blank line	↵	Used to insert a blank line to separate the user interface controls

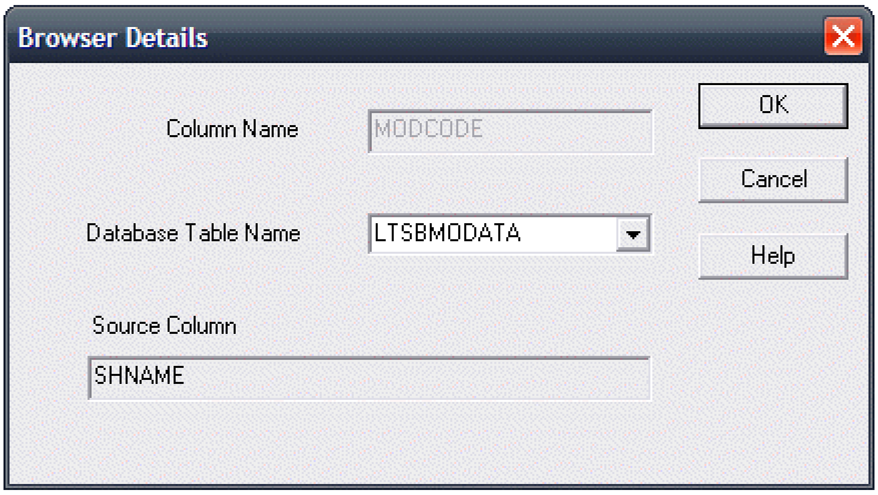
The following icons are used to control the grid properties, and to check for overlapping controls when a customisation has been upgraded

User interface control		Description
Number of columns	4 8	Customisations in TI Plus can have only four columns. However, if you are upgrading from a Trade Innovation customisation to TI Plus, then eight columns are shown. Once you have reorganised the controls to fit into a four-column

		layout you end of work by pressing the button.
Check for overlapping controls	✓	This button allows you to identify overlapping controls. When you press this button, TI Plus finds the first overlapping control and produces a pop-up window identifying it and its location. You can then fix the layout and use the button again to check for the next overlapping control.

Using drop-lists and browsers

These are called high quality controls, because the user will choose from a list of valid values, rather than keying in a value that may be wrong.


Browser Details

Column Name: MODCODE

Database Table Name: LTSBMODATA

Source Column: SHNAME

The browser control just needs you to specify which extra static table is to provide the list of valid values.

The drop-list (also known as Combo-box) will allow you to do the same and choose a database table to provide values. It also allows you to enter a fixed list of values, for use where it is known that the list will never change. It is recommended that if there is any chance of the list needing a new value in the future, that it is made an extra static table. This allows such change to be made simply by users, rather than needing a development project to be carried out on the customisation.

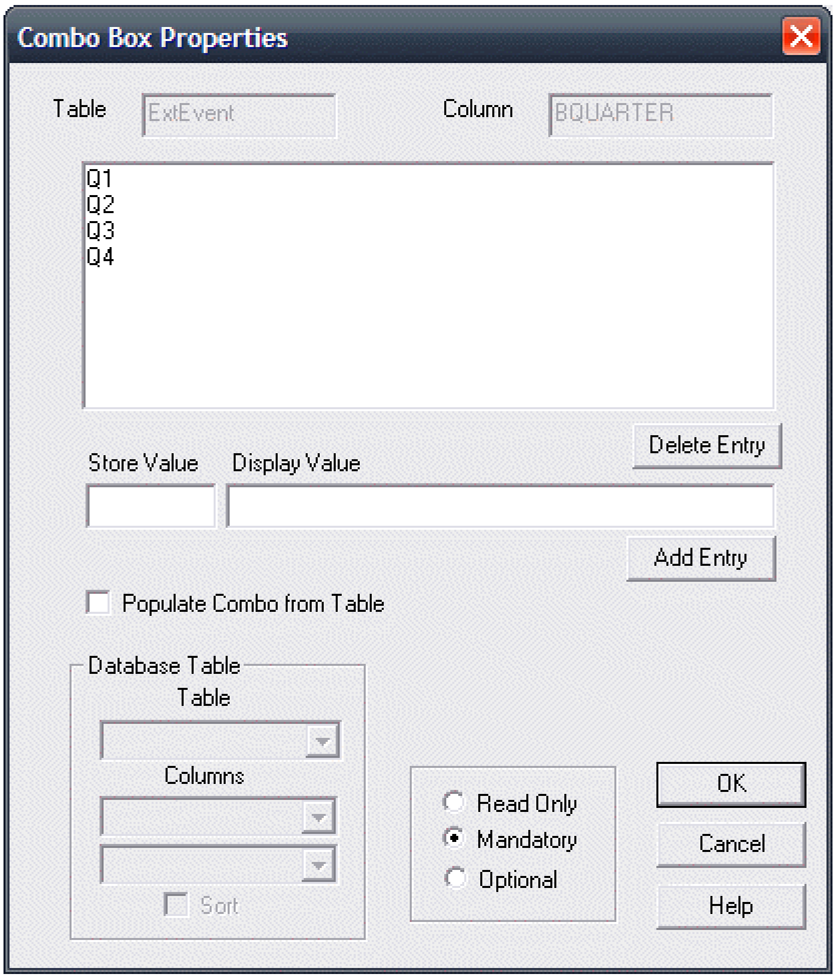


Table: ExtEvent
Column: BQUARTER

Q1
Q2
Q3
Q4

[ ] Populate Combo from Table

Read Only [✓ Mandatory] Optional

Re-using User Interface Designs

Once you have created one user interface for each different event or master, these can then be re-used for other (compatible) events and masters. This functionality is provided so that you can reduce the time required to develop the project by re-using user interface components.

User interface components can be re-used in the following ways:

- Using the Copy To button. This creates a straight replication of the work that you have done on one pane, creating a copy of it for another. Any subsequent changes must be applied to every pane that is affected. (For more detailed instructions on how to copy the user interface component, refer to the online help file.)
- Using the Alias Of button. This re-uses the definition of a user interface pane within the customisation to create controls for other masters/events that have an identical layout and content. The subsequent Customisation Editor build function will create a separate Java class for each event master as if they had each been built separately. The classes created would then all have to be added and take part in the customisation manager project to make a working customisation Use this feature if you know that you want to have separate Java code for each control (for example if you know that you want to override the Java separately and differently for each control)

Editing and Deleting User Interface Controls

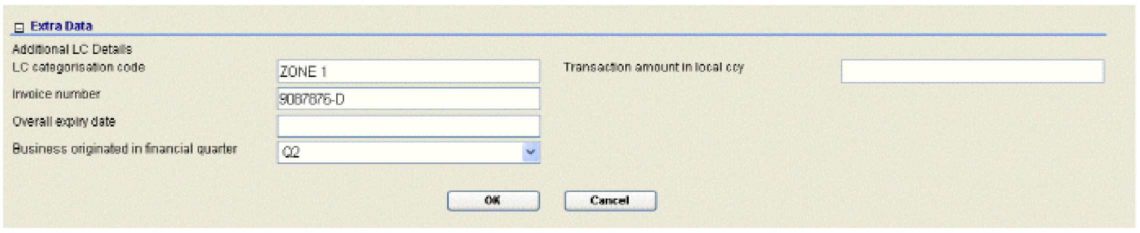
Once you have created the basic user interface, you can:

- Drag and drop user interface controls to re-position them, any associated labels must be moved manually to match
- Edit the properties of the extra fields by selecting it on the Browse Tables And Columns window, and clicking OK. This opens the Properties box where you can edit the position of the user interface item
- Delete a user interface control by selecting it and pressing the Delete key.

Examples of Extra Data at Run-time

The illustrations in this section show examples of extra data displayed at run-time.

The following illustration shows the extra data input fields during the Advise event.

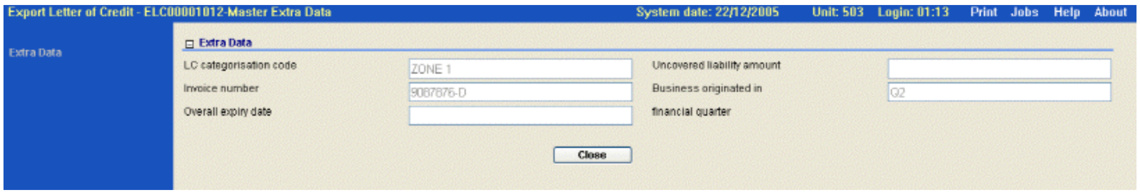

Extra Data

Additional LC Details

LC categorisation code	ZONE 1	Transaction amount in local ccy	
Invoice number	9087876-D		
Overall expiry date			
Business originated in financial quarter	Q2		

Depending on the setting of the system option CustomisedDataInSeparatePane, these fields may be displayed in the main window, along with the standard panes; or they may be displayed in a separate window, when the user selects the Extra Data link from within the event.

The following illustration shows the window used to display any extra data fields and their content when you select the Extra Data link from the Master Summary window


Export Letter of Credit - ELC00001012-Master Extra Data System date: 22/12/2005 Unit: 503 Login: 01:13

Extra Data

LC categorisation code	ZONE 1	Uncovered liability amount	
Invoice number	9087876-D	Business originated in financial quarter	Q2
Overall expiry date			

Chapter 5 Building Your Customisation

This chapter explains how to build a project, once you have completed defining the tables, extra field and associated user interfaces it will use.

Building Your Customisation

When you have completed defining the tables and extra fields your customisation will use and the user interface extensions to support them, you can use the Save button.

Save button in Customisation Editor creates the outputs required for the customisation builder (described in this chapter)

It saves your project text file as <project>.txt

It outputs the project definitions as an XML file which is the basis of a customisation being applied to a TI unit. This is named <project>.txt.xml

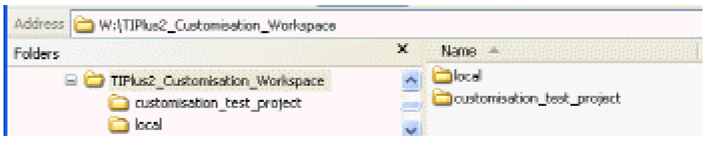
The save button also writes out a text file called nnnnnnnn.cnv (where nnnnnnnn is the name you gave to the project). This file is an input to the Data Model Viewer and allows the customised data definitions to be viewed with the other standard TI Plus table definitions.

Deploying the Customisation

The Java framework and classes need to be built into a set of libraries and deployed on the TI server. An Ant script has been provided which performs this task for you automatically once you have defined the relevant server/file location information.

To customise your test unit, first ensure that you have Java 1.5 or higher installed on your development environment. Next, install the j2ee edition of eclipse onto the machine. Once you have set up your development environment, follow the steps outlined below to customise your test unit:

Create a workspace directory, with a sub directory to contain your customisation java project. You can name these directories whatever you wish. Now create a new directory under the workspace directory which must be named "local". Your directory structure should be similar to the following:

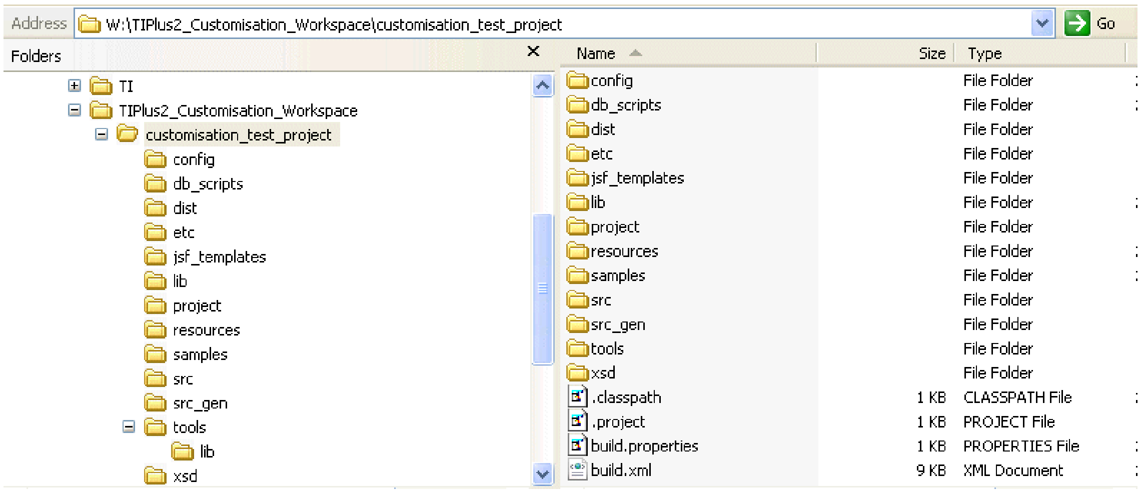

Address
W:\TIPlus2_Customisation_Workspace
Folders
TIPlus2_Customisation_Workspace
customisation_test_project
local
Name
local
customisation_test_project

Inside the local directory, create a new file named "customisation.properties".

Unzip the contents of tiplus_empty_customisation_project.zip into the project directory, which in this example:

```
W:\TIPlus2_Customisation_Workspace\customisation_test_project
```

Your project folder should now have the following structure:



Address W:\TIPlus2_Customisation_Workspace\customisation_test_project Go

Folders

```
⊞ TI
⊟ TIPlus2_Customisation_Workspace
  ⊟ customisation_test_project
      config
      db_scripts
      dist
      etc
      jsf_templates
      lib
      project
      resources
      samples
      src
      src_gen
    ⊟ tools
        lib
      xsd
```

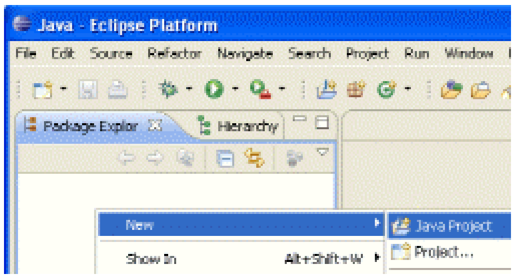
Name	Size	Type
config		File Folder
db_scripts		File Folder
dist		File Folder
etc		File Folder
jsf_templates		File Folder
lib		File Folder
project		File Folder
resources		File Folder
samples		File Folder
src		File Folder
src_gen		File Folder
tools		File Folder
xsd		File Folder
.classpath	1 KB	CLASSPATH File
.project	1 KB	PROJECT File
build.properties	1 KB	PROPERTIES File
build.xml	9 KB	XML Document

Rename your <project>.txt.xml file, created when you saved your customisation project in the Customisation Editor tool, to customisation.txt.xml. Then copy it into the "project" sub directory of your java project directory.

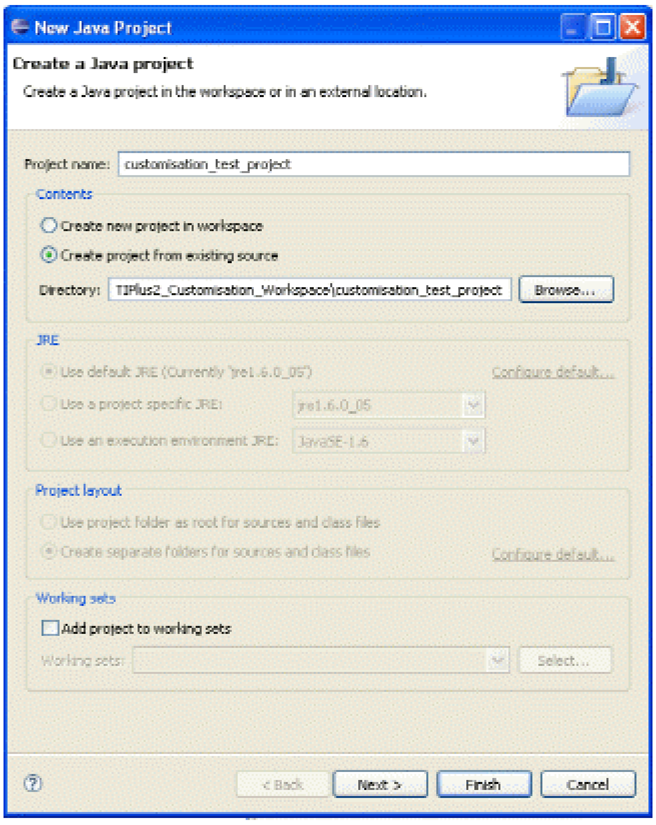
Start Eclipse, specifying the relevant workspace location, for example:

```
W:\TIPlus2_Customisation_Workspace
```

Once you have started eclipse, open the Java perspective and inside the Package Explorer window right click and select new Java Project:



Create the project from the existing source in your project directory, as below:


New Java Project
Create a Java project
Create a Java project in the workspace or in an external location.
Contents
Create new project in workspace
JRE
Use default JRE (Currently 'jre1.6.0_05')
Configure default...
Use a project specific JRE:
jre1.6.0_05
Use an execution environment JRE:
JavaSE-1.6
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

Project name: customisation_test_project

(●) Create project from existing source

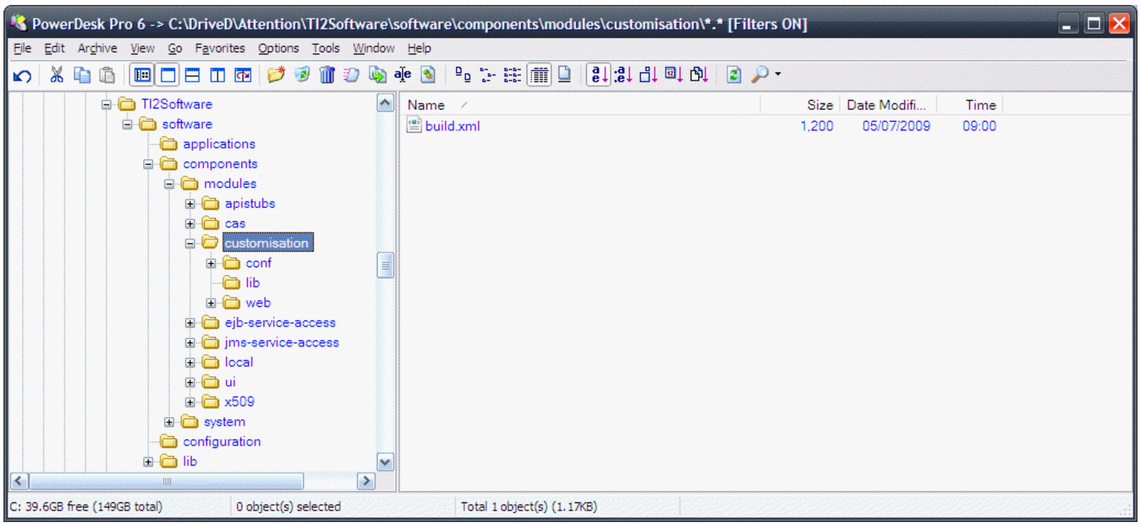
Directory: T:Plus2_Customisation_Workspace\customisation_test_project Browse...

Add a new file in the local directory you created earlier and name it local.properties. This file will be used to define your target tiplus2 software folder structure.

Add the following content to this file, replacing the dollar marked sections appropriately:

```
customisation.base.target.folder=$your test tiplus2 folder
name$/software/components/modules/customisation      (all in one
line!)
config.folder= conf/path/conf/customisation
pages_gen.folder=web/pages_gen_customisation
lib.folder=lib
package.path=$your package name$      (optional, but recommended)
```

For the example shown in the picture



PowerDesk Pro 6 -> C:\DriveD\Attention\TI2Software\software\components\modules\customisation*.* [Filters ON]

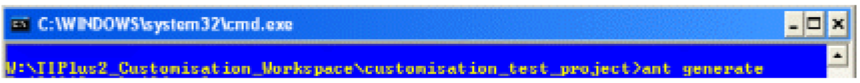
Name	Size	Date Modifi...	Time
build.xml	1,200	05/07/2009	09:00

C: 39.6GB free (149GB total) 0 object(s) selected Total 1 object(s) (1.17KB)

```
customisation.base.target.folder=C:/DriveD/Attention/TI2Software/softw
are/components/modules/customisation
config.folder= conf/path/conf/customisation
pages_gen.folder=web/pages_gen_customisation
lib.folder=lib
package.path=com.misys.tiplus2.customisation
```

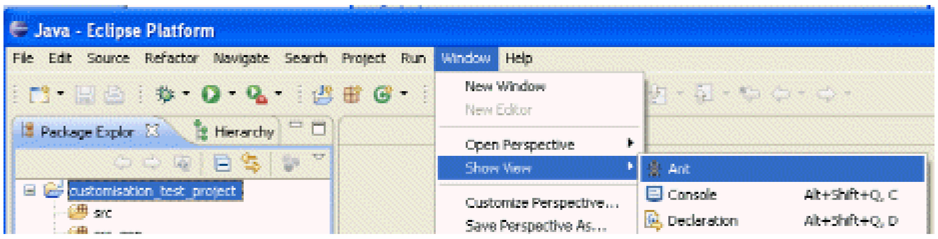
Now you are ready to run the customisation builder for your project. The builder is executed using the Ant script "build.xml" in the project root directory. You can run this from within eclipse or from the command line. To run from the command line you must first have installed Ant version 1.7 or higher. Simply navigate to the project root directory from within a command prompt and type:

```
ant generate
```

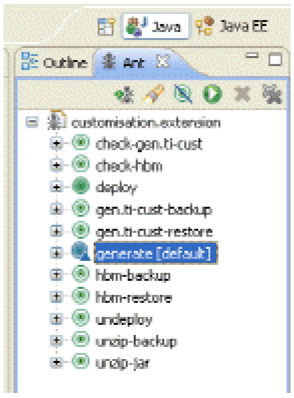


```
W:\TIPlus2_Customisation_Workspace\customisation_test_project>ant generate
```

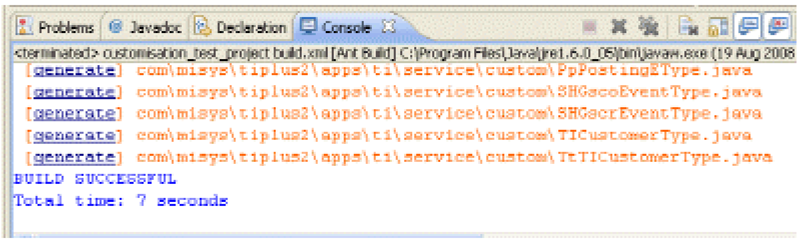
Alternatively, to run the script from within eclipse, first open the Ant view:



Next drag the build.xml file into the Ant window and drop down the displayed tasks. Generate is the default task. Run this task by double clicking on the Generate task within the Ant window:



You should see some output in the console window to show the progress of the generator. The generator may take a few seconds to complete. If the generator completes successfully, the following will be displayed in the console window:



```
<terminated> customisation_test_project build.xml [Ant Build] C:\Program Files\Java\jre1.6.0_05\bin\javaw.exe (19 Aug 2008
 [generate] com\misys\tiplus2\apps\ti\service\custom\PpPostingEType.java
 [generate] com\misys\tiplus2\apps\ti\service\custom\SHGscoEventType.java
 [generate] com\misys\tiplus2\apps\ti\service\custom\SHGscrEventType.java
 [generate] com\misys\tiplus2\apps\ti\service\custom\TICustomerType.java
 [generate] com\misys\tiplus2\apps\ti\service\custom\TtTICustomerType.java
BUILD SUCCESSFUL
Total time: 7 seconds
```

If the project does not generate successfully it is possible that there is an incompatibility in your current project and you should contact Misys.

Select the project directory and right click and refresh the contents. You should now see the generated Java source code in the "src_gen" folder. You are now ready to begin extending any of the Customisation classes/pane classes required to add specific functionality, such as business validation or adding posting field values.

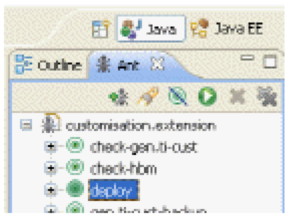
In this example, we will look at extending PostingCustomisation in order to add some values to the extra posting fields. When creating your extension classes you may find some of the sample code under the "samples" directory to be useful as a guide. It is very important that you adhere to strict naming conventions for your extension classes to ensure that the customisation builder can recognise the classes you have extended and where they should be invoked from within TI Plus. When creating an extension, the naming convention to follow is as follows:

```
$Major Code$_$Minor Code$_Customisation.java (for customisation class
extensions)

$Major Code$_$Minor Code$_CustomisedPane.java (for pane class
extensions)
```

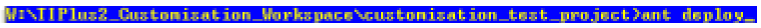
For example, to extend PostingCustomisation you should create a class called P_P_Customisation.java. To extend EventCustomisation for Import Letter of Credit Issue events you should create a class called ILC_ISI_Customisation.java. To extend CustomerCustomisedPane, you should create a class called C_C_CustomisedPane.java.

To deploy the customisation libraries onto your test server from within eclipse, simply run the deploy task on the Ant build script by double clicking on the task in the Ant window



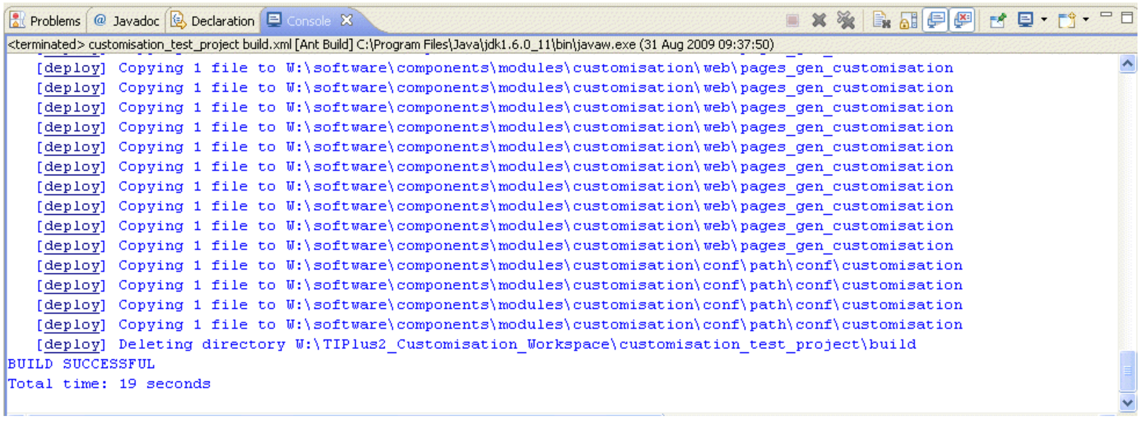
To deploy from the command line, navigate to the customisation project root folder from within a command prompt and type:

```
ant deploy
```



```
W:\TIPlus2_Customisation_Workspace\customisation_test_project>ant deploy_
```

If there are any problems with deployment check that the directory locations you are referring to in your local.properties are correct. A successful deployment will display the window illustrated below.



```
<terminated> customisation_test_project build.xml [Ant Build] C:\Program Files\Java\jdk1.6.0_11\bin\javaw.exe (31 Aug 2009 09:37:50)
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\web\pages_gen_customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\conf\path\conf\customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\conf\path\conf\customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\conf\path\conf\customisation
   [deploy] Copying 1 file to W:\software\components\modules\customisation\conf\path\conf\customisation
   [deploy] Deleting directory W:\TIPlus2_Customisation_Workspace\customisation_test_project\build
BUILD SUCCESSFUL
Total time: 19 seconds
```

You will now need to build the customised unit into the correct archive format ready for deployment, (see the *TI Plus Installation Guide*). To remove customisation, simply run the undeploy ant task.

TI Plus Database Scripts

The generate and deploy ant functions described in the previous section write two scripts for each of three target database managers.

A "Drop and create" script for all the required database changes to tables and members to a file called:

DB2	ExtData.bat
iSeries DB2	iExtData.sql
Oracle	oExtData.sql

An "Insert" script to implement version checking and to provide definitions for all extra static tables. This script must be run after the ExtData script above and also after any upgrade processing and script running. The building of the database upgrade script is described in detail in Appendix C.

DB2	ExtDefns.bat
iSeries DB2	iExtDefns.sql
Oracle	oExtDefns.sql

These SQL scripts will be written to a sub-directory of your customisation project folder called db_scripts.

It is essential that you run the ExtDefns script when you wish to deploy a customisation project. If the version stamp embedded in the generated code for the customisation project does not match the version inserted onto the database with this script, then TI will not allow startup to proceed.

DB2-Specific Instructions

The file must be used in a DB/2 command window to create empty customised tables, and has three parameters - the database name, user ID and user password.

Beware if you have run the customisation project before and subsequently entered customised data into the unit. This script deletes and recreates the customised tables. If any existing entered customised data must be retained then an update must be done rather than a create process - see Appendix C

You can run this script from your workstation only if you have DB/2 administrator rights for the target node and database unit you wish to update. You can, however, run the script on the database server itself (having signed on as an administrator user), using the same parameters.

If any existing entered customised data must be retained then an update script process must be done rather than a create process.

Chapter 6 Methods

This chapter covers the callback and exposed methods that support customisation.

TI Plus Methods

Regardless of which specific areas you are customising, there are common sets of methods that perform specific tasks. This set of methods must be set up correctly for TI Plus to interface with the object classes in order to perform the required processing. These methods fall into two main types:

- Callback methods
- Exposed methods

Callback Methods

To access information from TI Plus, exported methods are called directly from the customisation classes using a series of callback methods. The following table lists all the callback methods provided by TI Plus. Some are available on all the customisation classes; others are available on either a specific customisation class or on a selected few. All callback methods are accessible through the driver wrapper object.

Method	Customisation class
ConvertAmount	Customer, TICustomer, Event, Master
ConvertDate	Customer, TICustomer, Event, Master, Posting
ConvertFromToDBFormat	Customer, TICustomer, Event, Master, Posting
GetAccountOfficers	Customer, TICustomer, Event
GetAmountInBaseAtFXRate	Customer, TICustomer, Event, Master, Posting
GetAmountInBaseAtSpotRate	Customer, TICustomer, Event, Master, Posting
GetAmountInCurrency	Customer, TICustomer, Event, Master, Posting
GetBaseAtFXRate	Customer, TICustomer, Event, Master, Posting
GetBaseAtSpotRate	Customer, TICustomer, Event, Master, Posting
GetCountries	Customer, TICustomer, Event
GetCurrencies	Customer, TICustomer, Event
GetCurrencyAtFXRate	Customer, TICustomer, Event, Master, Posting

Method	Customisation class
GetCustomerFieldAsText	Customer, TICustomer
GetEventFieldAsText	Event, Posting
GetEventType	Posting
GetPostingFieldAsText	Posting
GetProductType	Posting
GetRawPostingFieldAsText	Posting (Netted Posting only)
GetTICustomerFieldAsText	Customer, TICustomer
GetWildCard	Customer, TICustomer, Event, Master
IsSubsidiary	Event
IsValidInputCode	Customer, TICustomer, Event
LogError	Customer, TICustomer, Event, Master, Posting

Of these methods:

- GetCountries, GetCurrencies and GetAccountOfficers are used to invoke TI Plus static data browsers
- GetEventFieldAsText, GetPostingFieldAsText, GetRawPostingFieldAsText, GetProductType and GetEventType are posting processing methods

For other methods refer to Appendix A for further details. The signatures of all the callback methods are also given in Appendix A.

To use any one of the callback methods your code should look like this:

```
getDriverWrapper().MethodName(parameters)
```

Exposed Methods

Exposed methods are included in the customisation classes to supply information to TI Plus when called by the user exits that have been introduced to support customisation. Some of the exposed methods are overridable, while others are core functional methods which are not overridable. Note that while all generated exposed methods will be visible in the generated source, upon deployment the core java will be re-generated to ensure stability. The following exposed methods are discussed in detail in this chapter and in Appendix A.

Overridable Method	Customisation class
AddExtraFields	Posting
AddExtraSettlementFields	Posting
CheckDoNettingOrBulking	Posting
postInitialiseBankAdditionalCode	Customer, TICustomer, Event, Master, Posting
initialiseUserFieldCodes	Customer, TICustomer, Event, Master, Posting

Overridable Method	**Customisation class**
NetWith	Posting
onSavingBankAdditionalCode	Customer, TICustomer, Event
UpdateMaster	Event
UpdateMasterOnRelease	Event
validatePane	Customer, TICustomer, Event
getUserFieldCodeString	Customer, TICustomer, Event, Master, Posting
getUserFieldCodeInteger	Customer, TICustomer, Event, Master, Posting
getUserFieldCodeBoolean	Customer, TICustomer, Event, Master, Posting
getUserFieldCodeAmount	Customer, TICustomer, Event, Master, Posting
getUserFieldCodeDate	Customer, TICustomer, Event, Master, Posting
putUserFieldCodeString	Customer, TICustomer, Event, Master, Posting
putUserFieldCodeInteger	Customer, TICustomer, Event, Master, Posting
putUserFieldCodeBoolean	Customer, TICustomer, Event, Master, Posting
putUserFieldCodeAmount	Customer, TICustomer, Event, Master, Posting
putUserFieldCodeDate	Customer, TICustomer, Event, Master, Posting

Final Method	**Customisation class**
CopyToAPI	Posting
GetFieldCodeAmount	Customer, TICustomer, Event, Master, Posting
GetFieldCodeAtIndex	Customer, TICustomer, Event, Master, Posting
GetFieldCodeBoolean	Customer, TICustomer, Event, Master, Posting
GetFieldCodeCount	Customer, TICustomer, Event, Master, Posting
GetFieldCodeDate	Customer, TICustomer, Event, Master, Posting
GetFieldCodeDetails	Customer, TICustomer, Event, Master, Posting
GetFieldCodeInteger	Customer, TICustomer, Event, Master, Posting
GetFieldCodeString	Customer, TICustomer, Event, Master, Posting
GetJournalCount	Customer, TICustomer
GetJournalLine	Customer, TICustomer
GetOleControl	Customer, TICustomer, Event, Master, Posting
GetExtendedString	Posting
GetValidationError	Customer, TICustomer, Event, Posting
GetValidationErrorCount	Customer, TICustomer, Event, Posting
GetValidationWarning	Customer, TICustomer, Event, Posting
GetValidationWarningCount	Customer, TICustomer, Event, Posting
Initialise	Customer, TICustomer, Event, Master, Posting
Initialise	Customer, TICustomer, Event, Master, Posting

Final Method	Customisation class
PostInitialise	Customer, TICustomer, Event, Master, Posting
InitialiseMaster	Event
Journal	Customer, TICustomer
OnSaving	Customer, TICustomer, Event
SetDefaults	Event
SetReadOnly	Customer, TICustomer, Event, Master, Posting
Validate	Customer, TICustomer, Event, Posting

These exposed methods can be grouped according to their function, as follows, and are organised within this chapter under these headings:

- Standard system level methods, that are automatically generated by the Customisation Builder
- Instance initialisation methods, so that you can initialise each extended entity instance with default values
- Pane methods, which TI Plus calls as part of the processing of panes
- Master/Event dataflow methods
- Field code methods
- Posting processing methods

Standard System-level Methods

When you create your customisation, the Customisation Builder automatically generates the following standard methods for you:

- initialise
- getWrapper
- getPane
- getDriverWrapper
- getCompWrapper (posting only)
- getApiWrapper (posting only)
- getPostingType (posting only)
- isForSnapshot (event and master only)
- getSSWrapper (master only)
- setSSWrapper (master only)

Do not change these functions – any changes made will be lost upon deployment where the generated java source will be regenerated

Instance Initialisation Methods

There are common initialisation methods for each extended entity type. TI Plus calls the Initialise() method so that you can initialise each extended entity instance with default values. The Customisation Builder generates default initialisation code, making use of the data definitions that you have specified. If you have any special initialisation to be performed it should be performed in the postInitialiseBankAdditionalCode method.

Pane Methods

Customisation classes are embedded within dialogs. In the appropriate part of the application, TI Plus displays the pane with its embedded java class, and calls the exposed methods on the customisation class at the appropriate time. TI Plus has full control over the pane.

As part of the pane processing, TI Plus provides the following calls:

- PostInitialise (which calls postInitialiseBankAdditionalCode)
- SetReadOnly
- Validate
- OnSaving

There is also a set of methods (which the object class must provide) that TI Plus calls to find out about the results from the Validate() call. The Customisation Editor generates all the above methods as required.

Below is a table with more details about the pane methods:

PostInitialiseMethod	The PostInitialisemethod is called just before the pane is displayed. This is where the pane initialisation processing should be performed. You have access to the extended entity and to TI Plus's event field mechanism. This is also where any special controls such as lists are set up. Pane events on the controls are executed in the normal way and handled by the controls.
SetReadOnly Method	The SetReadOnly method is called to tell the customisation object whether the pane is to be displayed in View or Edit mode. When the pane is displayed in view mode, all data entry controls should be disabled.
Validate Method	In this method, the data in the extended instance will be validated. In other words, the Validate method is called when TI Plus validates all the other data (standard fields) in the current context. The Validate method returns false if there is a problem with the extended entity data. If there are warnings or errors, then TI Plus makes calls to the customisation objects to get the details of the warnings and/or errors. Information from warnings and errors is not returned directly back to TI Plus: a user-defined object (called ValidationDetails - see Appendix A) and a number of common methods have been provided. Some common functions allow the set up of warning and error information, and some other common functions allow TI Plus to interrogate this validation information. At the end of the Validate method a call is made to the validatePane method. This method is overridable and can be used to add any business validation rules. The IValidationDetails interface can be used to report any errors or warnings to TI Plus following the validation. The code below illustrates how to do this:

```
@Override
public void validatePane(IValidationDetails
validationDetails) {
if (getPane().getObcr()==null ||
getPane().getObcr().equals("") &&
getPane().getIrb()==null ||
getPane().getIrb().equals("")) {
validationDetails.addError(ErrorType.MustBeEntere
d,
"One of instructions to receiving bank or object
of contract must be entered.");
}
super.validatePane(validationDetails);
}
      }
```

In this example the business validation performed ensures that the user has entered at least one of the two fields IRB and OBCR. If the user has not entered either field a validation error is reported.

OnSaving Method	The OnSaving method is called when the data from the customisation objects needs to be copied back into the extended entity instance in TI Plus. To add any functionality at this stage, the overridable method onSavingBankAdditionalCode has been provided.

Master/Event Dataflow Methods

The Master table records the current state of a transaction. The only way to change Master data is through events. All events can be initialised with the existing data in Master record. When an event is completed the data on the master record is updated from the event. This dataflow mechanism must be the same for extended masters and extended events. You are not allowed to amend Master details at any point in any method except UpdateMaster and UpdateMasterOnRelease on the event. Three special methods are provided on event controls:

- SetDefaults
- UpdateMaster
- UpdateMasterOnRelease

These are described in the table below:

SetDefaults Method	The SetDefaults method is called when the event in being initialised. In this method the Customisation builder will generate Java code to copy data from the Master extended entity into the Event extended entity if the event is not a defining event.
UpdateMaster Method	The UpdateMaster method is called when the event is released. In this method, the Customisation Editor will generate Java code to copy data from the Event extended entity into the Master extended entity.

Do not write to the extended Master entity except in the UpdateMaster method on an Event.

Event Field Code Methods

Event field codes are used by TI Plus's document and posting definitions. By accessing data using a code rather than the actual variable name, the underlying data structures can then be changed while leaving external data access untouched.

There are two aspects of the use of the event field mechanism that are particularly relevant to customisation.

First, the customisation object classes for extended Event, Master, Customer, TICustomer and Posting must provide event field information into TI Plus. TI Plus's data output mechanisms (such as documents and postings) use the event field mechanism to get information about their context. Customisation needs to provide extra field data to TI Plus using the event field mechanism, telling TI Plus what event fields it has and supplying values for those event fields when requested to do so.

Second, customisation can itself use the event fields mechanism to retrieve TI Plus's transactional data. For example, this may be to set up extra fields required on postings by calling the GetEventFieldAsText function on the extended posting exit inside TI Plus. TI Plus's system tailoring application includes an Event Field Calculator, which allows you to examine all the available event field codes for any given product and event and provides you with the necessary parameters to call the GetEventFieldAsText function.

Setting up event field codes

You may use two types of event field code within customisation objects:

Standard codes which correspond to the fields defined on the customised tables

User-defined codes, which you can add yourself. These can be set up to return calculations which may or may not involve the extra fields

The method InitialiseFieldCodes is called when an object class is first loaded. The Customisation Builder generates this method. Once the standard event field codes have been initialised, another call is made to the method initialiseUserFieldCodes to initialise your own event field codes. The method addFieldCode is used to add an event field code. All the user-defined codes should be defined in an override of the initialiseUserFieldCodes method as shown in the example below:

```
@Override
public void initialiseUserFieldCodes() {
      addFieldCode("cXYZ", "description of new field",
FieldCodeType.String);
      super.initialiseUserFieldCodes();
}
```

The first parameter to the addFieldCode method is the field code string. Note that all customisation fields should start with a lower case 'c'. The second parameter to the addFieldCode method is the string description of the field. The third parameter is a java enumeration representing a supported field code type.

Customisation event field codes consist of a lower case letter c (prefixed automatically by TI), followed by three upper case letters (that were specified in the customisation editor).

When the event field is defined it is defined as being a certain type. The types available for customisation extra fields are:

- Amount
- Boolean
- Date
- Integer
- String

The following methods return the above data types:

- GetFieldCodeAmount
- GetFieldCodeBoolean
- GetFieldCodeDate
- GetFieldCodeInteger
- GetFieldCodeString

Each event field that is defined must have code to calculate its value. The code to populate any user defined, calculated fields (those which are not stored on the database) should be placed in the following override-able methods:

- getUserFieldCodeString
- getUserFieldCodeInteger

- getUserFieldCodeBoolean
- getUserFieldCodeAmount
- getUserFieldCodeDate
- putUserFieldCodeString
- putUserFieldCodeInteger
- putUserFieldCodeBoolean
- putUserFieldCodeAmount
- putUserFieldCodeDate

Supplying event field codes to TI Plus

The customisation event field codes are used by TI Plus to tell users that the extra fields are available when documents and postings are defined using TI Plus's system tailoring application.

The following methods are generated by the Customisation Builder and are then used by TI Plus to find about the event field codes that you have defined:

- GetFieldCodeCount
- GetFieldCodeDetails
- GetFieldCodeAtIndex

Posting Processing Methods

The extra fields for postings are defined in the usual way using the Customisation Editor. However, there is no data entry facility for postings. There is only a read-only pane for displaying the values. You must provide the code that sets up the data in the extra fields.

Two methods in the Posting customisation class can be used to set up the postings extra fields:

AddExtraFields is the method you should normally use and it is called for each posting in turn

AddExtraSettlementFields should be used when the extra posting fields need to be set up with some information from settlement details

From these methods you have access to information about the current posting and information about the current context of the event using the event field mechanism.

The AddExtraFields method is used to generate raw postings or accrual postings. The code below provides several examples of its use:

```
getWrapper().setM_lccc(getDriverWrapper().GetEventFieldAsText("cAAL ",
"s", "", "E"));


---------------------------------------------
getWrapper().setM_oexpiry(getDriverWrapper().GetEventFieldAsText("cAAP
", "d", "", "E"));


---------------------------------------------
getWrapper().setP_taxapp(getDriverWrapper().GetPostingFieldAsText("PTY
", "cABF"));

---------------------------------------------
String tCcy = getDriverWrapper().GetEventFieldAsText("FOA", "v", "c",
"E");
String tAmt = getDriverWrapper().GetEventFieldAsText("FOA", "v", "a",
"E");
getWrapper().setCcy(tCCy);
```

```
if(tAmt != null && !tAmt.equals("")) {
      getWrapper().setA_money(getDriverWrapper().ConvertFromToDBFormat
(tAmt, tCcy, "T"));
}
```

In these examples M_LCCC, M_OEXPIRY, P_TAXAPP, A_MONEY and CCY are five extra fields on the Posting table, and are designed in advance (using the Customisation Editor) with full knowledge of their types, purposes and applicability to the different types of postings and events.

M_LCCC and M_OEXPIRY are set up from extra event fields by making use of event field mechanism.

P_TAXAPP is set up with a different method called GetPostingFieldAsText. Similar to GetEventFieldAsText, GetPostingFieldAsText is a different method under the event field mechanism and it is used to get access to the current posting details. From the Event Field Calculator, you should be able to tell that the code PTY (the first parameter) is used to get hold of the posting party. Because party is a complex data type in TI Plus and has many different parts (fields or data members), the second parameter (part code) in the above code of line 3 is used to access a particular part of the posting party. Before you use a part code (like cABF), make sure that you can see it from the Event Field Calculator. Note that because field code "PTY" is a party and "cABF" is prefixed with the letter c, it is indicating that an extra field in Customer and/or TICustomer is being accessed.

The last example given above differs from the others. It is dealing with a TI Plus data type of Money. Within the Customisation Editor, when you define an extra field of the data type Money (A_MONEY in the example) the Customisation Editor will automatically generate an extra field of currency code (string field) to go with it, and A_MONEY actually just represents an amount (decimal field). You should be able to find the physical name for the currency code that is related to a Money field that you have defined by examining EXTDATA.bat file, as it is always next to the Money field and is called CCY or CCY-n (where n is a one or two digit integer).

Misys recommends that you should follow the approach that has been demonstrated for setting up an extra posting field of the data type Money in your customisation code.

The AddExtraSettlementFields method can be coded in a similar way.

To help you with adding extra fields to postings, a special posting code is provided. To get this posting code for any raw posting, you can use:

```
CmnGetPostingCodeAndDescription(g_myPane, pCode, pDescription)
```

in AddExtraFields and AddExtraSettlementFields methods to return the code (pCode) and description (pDescription) for the corresponding posting. For a full list of posting codes and their descriptions search kapsh.lgc file for POSTINGTYP or PostingCode and examine the (domain) list.

Raw postings will be netted into netted postings, and accrual postings could be bulked together. The method CheckDoNettingOrBulking is used to check whether the extra posting fields are compatible for netting or bulking purposes and it can be modified to achieve some desired netting or bulking behaviours. From the generated code for CheckDoNettingOrBulking, it can be seen that netting or bulking will happen only when all the extra fields are the same. When the compatible raw postings are to be netted into a netted posting, the event fields of the netted posting will be initialised with those of the first encountered raw posting, then the method NetWith will be used to net the other raw postings to it. In the method NetWith, you can either access the event field information directly from the current raw posting by using:

```
getWrapper().set$ExtPostingComponent$(getCompWrapper().get$ExtraFieldN
ame$());

or extract more information from the raw posting itself by using:
getDriverWrapper().GetRawPostingFieldAsText(fieldCode, partCode)
```

Chapter 7 General Programming

This chapter provides some general advice on writing code to customise TI Plus.

Accuracy in Calculations

Intel 32 bit, MS and Java provides double precision arithmetic to an IEEE standard, but may not have the accuracy needed for some calculations involving rates and large amounts. Pay particular attention, therefore, to the accuracy of calculations involving rates and large amounts and also when two numbers are compared.

Accessing Extended Data

Customisation does not make connections to the database. All database activity is done by the main TI application and the customisations controls call back into TI to provide database actions.

In your code you will need to write to and read from extended entities. The extended entities are available on the TI Plus pane.

Class	Name of extended entity
Master	ExtMaster
Event	ExtEvent and ExtMaster
Customer	ExtCustomer
TICustomer	ExtTICustomerDetails
Posting	ExtPosting

To access a field on an extended entity use the wrapper object. If you wish to do so you should check the Multi-character Field Separator checkbox on the Build dialog (see page 39) of the Customisation Editor.

So to assign a value of 99 to the code field on the extended Event entity, in your Event subclass you would write:

```
        getWrapper().setCode(99);
```

Accessing Static Tables Directly

At some points during your input of data for extra fields, you may want to access some data from tables directly. This can be achieved by the CreateQuery method provided by the TI Plus run-time system - an example of utilising the CreateQuery method is demonstrated in the samples sub-directory of the customisation eclipse project root archive.

The technique shown in the sample can only be used to read some data from static tables; getting data from transaction tables should be carried out by means of the Event Fields mechanism.

Passing Information Back to TI Plus

Information is passed back to TI Plus using a combination of java return types and where necessary java objects to encapsulate multiple return parameters such as IValidationDetails to allow error and warning reporting.

Browser Callbacks

The following three data items can be selected in TI Plus:

- Currencies
- Countries
- Account Officers

This can be achieved by making the following calls in your code:

- getDriverWrapper().GetCurrencies()
- getDriverWrapper().GetCountries()
- getDriverWrapper().GetAccountOfficers()

Handling Exceptions

It is entirely up to the user as to the level of exception handling they wish to implement in their extension classes. All overridable methods from the customisation class or customised pane are always surrounded by exception handling methods from the generated java code. In this way the user can just perform the basic checked exception handling required by java to ensure compilation, or they are free to add their own exception types and throw these to the calling super classes when required.

Chapter 8 Going Live with Your Customisation

This chapter explains how to move your customisation into a live environment.

Overview

Once you have set up the additional tables needed to support customisation in your test unit database, the basic steps involved in going live with your customisation are as follows:

1. Make the extra field definitions available for viewing.
2. Add the extra fields to the TI Plus database in your live unit.
3. Build the Java into libraries
4. Copy the libraries, XML schema and HTML pages to the TI Plus Application Server using the ant script's deploy task.

The steps described in this chapter do not copy any test data onto the live unit, nor do they initialise the columns for existing masters and events. Only new transactions subsequent to going live with a customisation project will be affected

Making the Extra Fields Definitions Available for Viewing

The .cnv file produced during the Customisation Editor build can be copied to the directory where the TI Plus data model viewer is installed. It must then be renamed to

```
ex2nna.cnv
```

where *nna* is the TI Plus version that is being customised.

You will then be able to view the customer-defined fields using the data model viewer.

Adding the Extra Fields to the Live Unit

The EXTDATA.bat script (created by the build function of the Customisation) contains an SQL script to create new tables and extensions to existing tables in the database unit chosen. Run the EXTDATA.bat script (see page 39) from a DB2 Command Window. This should run to completion without requiring any additional input.

You can run this script from your workstation only if you have DB/2 administrator rights for the target node and database unit you wish to update. You can, however, run the script on the database server itself (having signed on as an administrator user), using the same parameters.

Building Customisation Libraries

The build process for the live unit will be identical as the process outlined in Chapter 7.

Deploy Customisation

As described in Chapter 7, however change the local.properties to point to the real tiplus2 software directory, instead of your test system. If you are building from a common directory for both, no change would be required.

Chapter 9 Postings in a Customised System

This chapter discusses key points relating to extending postings generated by the system.

You must have a full understanding of the different types of postings that are generated by TI Plus, and decide how each posting type is to be affected before attempting to add extra fields. For details of the posting types generated by TI Plus see the *TI Plus Business Reference Guide*.

Posting Principles

The postings generated by TI Plus are of two types:

- User-defined postings, which are set up using the system tailoring application. These are generated and released as part of event processing
- Internally generated postings, arising from charge and financing details entered for a transaction. Postings of this type are generated as events are released, and subsequently during end-of-day accruals processing

These areas are covered in more detail in the following sections.

Generation of Postings

The postings generated by an event are essentially of two types:

- Postings without any associated settlement details
- Funds movements postings, where the posting has attached settlement instructions

Where a posting is to a charge income account, SP account or an account type which is not valid for funds movement the system generates a debit or credit posting to an account for the appropriate amount and value date.

Where a posting is a charge settlement posting or is defined as being an Event Party Settlement posting it will additionally have attached settlement instructions derived from the standing settlement instructions and master settlement instructions held for the party. These postings are used to:

- Project future funds movements. Where the value date is in the future the posting details are passed to the back office on a projection basis immediately the event is released. This allows the details of future funds movements to be included in nostro positions
- Post the projection on due date. When the value date is reached the system passes the posting to the back office to be posted
- Generate pay and receive advices to be sent to the account/nostro being debited

When considering this in the context of adding extra data to postings, it may be that a posting needs to be extended to include additional information to allow for the funds movement being passed to the back office as both a projection and then subsequently as a posting.

Charge-related Postings

Postings are generated as a result of taking charges, either standard or periodic. At the simplest level, a charge results in a debit to the charge payer and a credit to a profit and loss account. Depending upon the charge type and system options defined, TI Plus may also:

- Post to tax accounts
- Generate postings to income receivable accounts for periodic charges
- Convert charge amounts to a booking currency through exchange accounts
- Split fees between different branch profit and loss accounts
- Bill charges

Consequently, you should consider each different type of charge posting and determine what extra data it may require. It may be that the same extra data is required on all postings generated by a charge; alternatively, only some of the postings generated may require extra data. For example, the charge code may be added to all postings, but the extra fields you have added to the Customer table (for example) may only be required on the charge settlement postings.

Postings Defined for Each Event

TI Plus generates postings according to the posting definitions set up against the product and event. (Postings generated by the Manual Bookkeeping event are not set up using posting definitions, but entered directly as part of each event.)

Postings are generally specified at event level, and use details from the event to generate the appropriate accounting. For example, postings can be set up for:

- The import letter of credit Issue event, to record LC issuance liability and contra liability
- The Finance Import LC event, to record the deal and to post interest received in advance
- The Repay Fin event, to record the repayment of principal and interest

The posting definitions reference event fields, which provide information held for the event or master. These event fields (including customised event field) are also available to allow additional data to be added to postings by extending the generated customised code.

Postings can also be generated for each of the following attachments entered within the event:

Foreign exchange deals	Exchange deals entered as part of settlement.
Collateral	Details of collateral recorded against the master.
Part payments	Individual payments made under a Claim Received, Documents Presented or collection order Payment event.
Disbursement	Payments under assignments, transfers and back-to-back credits.
Assignments	Details of assignments.
Advances	Finance deals with interest at maturity (for systems where TI Plus's bills financing module is not installed).
Discounts	Finance deals with interest at inception (for systems where TI Plus's bills financing module is not installed).
Repayments	The repayment of advance deals, for example trust receipt loans, import financing or outward collection financing (for systems where TI

Plus's bills financing module is not installed).

TI Plus's posting definitions reference event fields specific to these attachments, which provide information held for the attachment. These attachment-level event fields are also available (in addition to the master or event fields) to allow additional data to be added to postings by extending the generated customised code.

Financing and Charge Accruals

Accrual postings are generated as a result of the daily periodic charge accrual and bills financing accrual processes. TI Plus will for example:

- Transfer income to profit and loss on a daily basis for periodic charges
- Amortise interest received in advance for a finance deal
- Accrue interest or past due interest for a finance deal

As these postings are not associated with a specific event there is no specific event-level information. However, the event field mechanism can be used to obtain information about the master, finance interest or charge interest accrual details, which can then be added to the accrual postings by extending the generated customised code.

Netting of Postings

Once the individual postings (these are also known as raw postings) have been generated for an event, the system attempts to net postings to the same account together so as to produce netted postings. For example, TI Plus will net all charge debits to the same account, resulting in a single debit rather than a series of individual debits.

Postings will be netted together providing that the following are the same:

- Account (customer account/system parameter/charge account)
- Currency
- Value date
- Customer ID
- Reference
- User codes
- Party
- Related party
- Team

It is the netted posting that is passed to the back office upon release of the event, not the individual raw postings.

The raw postings will continue to be netted together under customisation where the extra data is the same on each posting. TI Plus compares the data in the standard posting fields and in the extra data on each posting during the netting process. If for example an extra field is added at master level which is then applied to all postings, netting will continue to work as at present. Alternatively, the generated code can be altered to achieve the required netting.

Typically where information is added to the raw posting from the attachment this will result in the raw postings not being netted, resulting in a netted posting for the raw posting amount.

For example, without customisation the system would typically net together a charge settlement posting, a discount interest posting and finance deal posting. If the same master-level extra field was added to all postings, netting would continue to net the raw postings together.

			Extra posting field
Type	**Dr/Cr**	**Posting details**	**Col 1**
Charge	Dr	Common details	Master code
Discount	Dr	Common details	Master code
Finance	Cr	Common details	Master code

However, if customisation were to add the charge code to the charge posting, the discount interest rate to the interest posting, and the term to the deal posting, this would result in three individual postings being passed to the back office.

			Extra posting fields			
Type	**Dr/Cr**	**Posting details**	**Col 1**	**Col 2**	**Col 3**	**Col 4**
Charge	Dr	Common details	Charge code			
Discount	Dr	Common details		Rate		
Finance	Cr	Common details			Start	Maturity

Netting of Funds Movements

Raw postings are also netted into netted funds movements on a similar basis, except that the settlement details are included in the netting criteria. These details are displayed in the Settlements window. This netting cannot be changed.

Netted funds movements are used by the system to generate pay and receive advices via SWIFT or Mail.

Bulking

The TI Plus system tailoring application's System|Trade Finance System Option menu option allows you to specify whether accruals are passed separately or bulked to the same accounts.

If there is no bulking the system passes individual accrual postings.

If your system is set to bulk, TI Plus bulks together accrual postings with the same characteristics. In the bulking, accrual postings are bulked together where the following are the same:

- Branch
- SP or SK
- Currency
- Transaction code
- Related customer
- Related customer - customer type
- Related customer - analysis code
- Related customer - parent country

- Related customer - team

The accrual postings will continue to be bulked together under customisation where the extra data is the same on each posting. TI Plus compares the data in the standard posting fields and in extra fields on each posting during the bulking process. The generated code can therefore be altered to achieve the required bulking. For example, the code could be extended to:

- Never bulk accrual postings
- Bulk according to different criteria (in which case the required data on the resulting bulked posting would need to be set up according the relevant business rules)

Adding Data to a Posting

Each time a raw posting or an accrual posting is generated, the system allows data to be added through extending the PostingCustomisation java class according to the type of posting, as defined above.

There are two methods which can be used to add extra data to a posting:

- Add Extra Data to Posting. This will typically be used where some general identifier is being added to a raw posting
- Add Extra Settlement Posting. This adds data to netted postings and will typically be used where attributes of the account (such as account type) need to be cross-referenced to determine what data is to be added

This is particularly important for settlement postings where the original raw posting party account may have been overridden via settlement maintenance.

Examples of the resulting postings are as follows:

Dr	Netted	Settlement A/C 2
Cr	Netted	SKP&L
Cr	Raw	SKP&L
Dr	Raw	Settlement A/C 1

(but when you are in Raw Postings you would see A/C 1, and if you were in Add to Settle you would see A/C 2 against individual postings)

The code section applicable to posting type can be enhanced to add extra event fields. At the point the posting is generated, the PostingCustomisation java class is called, allowing any extra data required to be added, with the system also providing the necessary context of master, event, posting details (including account) and attachment. The event field mechanism can be used to obtain details from the related items and/or condition the setting of the required extra fields. When a posting is being processed, the following information can be obtained:

- The posting itself, for example:
    - Whether debit or credit
    - Account plus branch, basic number, suffix, external account number and back office account number
    - Amount
    - Currency
    - Transaction code
    - SP System parameter the account represents
    - SK Charge code the account represents
- The account to which the posting is to be made:
    - Number

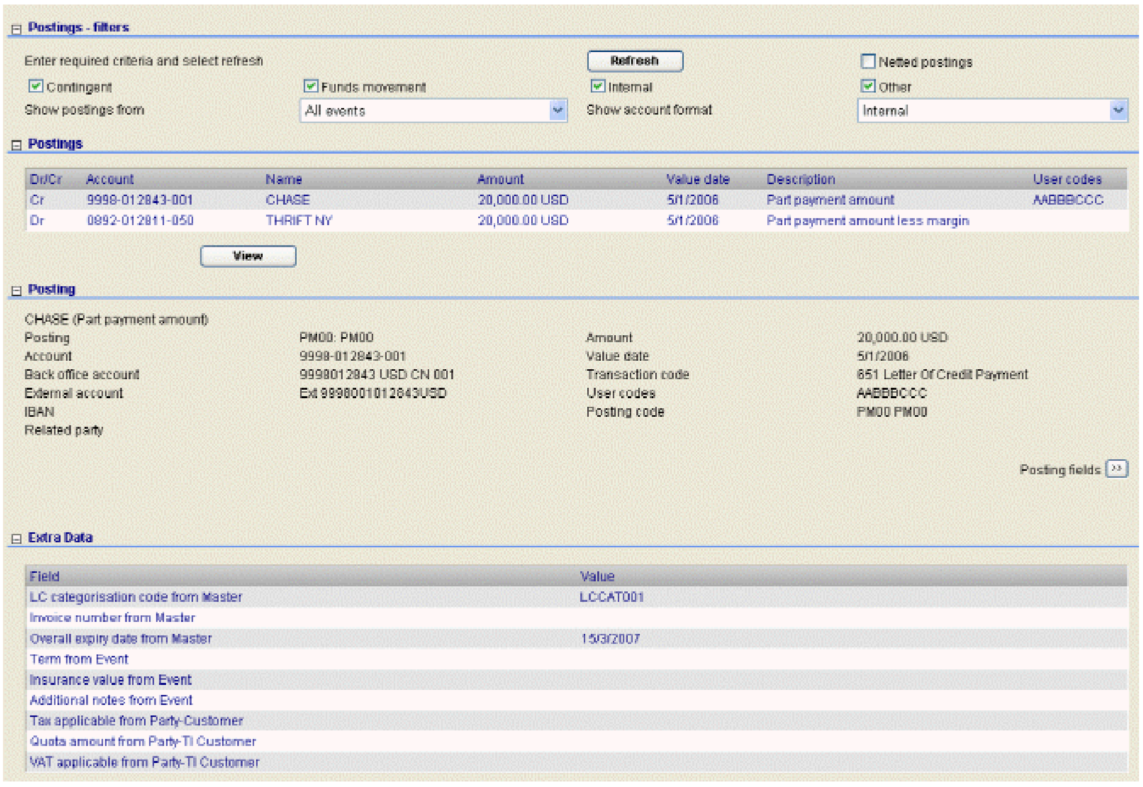
- Short name
- Contingent flag
- Internal flag
- Valid for funds movement

- The party for whom the posting was made, for example:
    - Analysis code
    - Risk country
    - Plus extra fields from the Customer table
    - Plus extra fields from the TICustomerDetails table
- The master or event, for example:
    - All standard extra fields such as issue date and expiry date
    - Plus extra fields on the Master and/or Event tables
- The attachment itself (see page 60):
    - FX deal - rate, base currency equivalent
    - Collateral code - code, description
    - Part payment - type, term, number of days

Viewing Data Added to Postings

Extra data added to postings can be seen in TI Plus when viewing postings in the Postings window.

Uncheck the Netted check box and press Refresh. Then highlight a posting from those listed in the Postings pane and press View.



Enter required criteria and select refresh

☐ Netted postings

✓ Contingent ✓ Funds movement ✓ Internal ✓ Other

Show postings from: All events

Show account format: Internal

Postings

Dr/Cr	Account	Name	Amount	Value date	Description	User codes
Cr	9998-012843-001	CHASE	20,000.00 USD	5/1/2006	Part payment amount	AABBBCCC
Dr	0892-012811-050	THRIFT NY	20,000.00 USD	5/1/2006	Part payment amount less margin	

Posting

CHASE (Part payment amount)

Posting	PM00: PM00	Amount	20,000.00 USD
Account	9998-012843-001	Value date	5/1/2006
Back office account	9998012843 USD CN 001	Transaction code	651 Letter Of Credit Payment
External account	Ext 9998001012843USD	User codes	AABBBCCC
IBAN		Posting code	PM00 PM00
Related party			

Field	Value
LC categorisation code from Master	LCCAT001
Invoice number from Master	
Overall expiry date from Master	15/3/2007
Term from Event	
Insurance value from Event	
Additional notes from Event	
Tax applicable from Party-Customer	
Quota amount from Party-TI Customer	
VAT applicable from Party-TI Customer	

Event Fields Available

The system tailoring application's Output|Browse and Field Codes menu option provides a means to view the event fields that are available for the following:

- Product/events
- Events/business areas
- Part codes
- Postings

This allows the field codes used for both standard event fields and extra event fields to be identified.

See the *TI Plus System Tailoring User Guide* for full instructions on using this menu option.

Posting Identifier

Each posting generated by the system has a unique identifier. This posting code can be used in Java to condition postings, or simply to confirm or verify that the correct posting has been identified.

A sample list is given below. A full list can be obtained by searching through the file KAPSH.LGC for the field PostingCode in the table Posting.

SP101 posting for conversion of tax amount in charge currency	=	C001
SP101 posting for tax amount in charge booking currency	=	C002
SP2.05 tax accumulation account for tax amount in charge booking currency	=	C003
SP2.05 tax accumulation account for tax amount in charge currency	=	C004
SP101 for reimbursement charge due from seller/other - original charge amount	=	C005
SP101 for reimbursement charge due from seller/other - charge amount due to buyer	=	C006
Settlement of reimbursement of charges due to buyer from another party - charge amount - due to buyer	=	C007
Settlement of tax charge amount due from buyer	=	C008
Settlement of tax charge amount due from seller	=	C009
Settlement of charge amount due from buyer	=	C010
Settlement of charge amount due from seller	=	C011
Settlement of charge amount due from other party (from an intermediary bank)	=	C012
Settlement of charge amount due from other party	=	C013
Reverse posting from SP for billed income when billed charge amount is taken	=	C02.0
Posting taken charge amount to SK P&L when billed charge amount is taken	=	C015
Reverse posting from customer billing account when billed charge amount is taken	=	C016
Settlement of charge amount due from billed customer	=	C017

Reverse posting from SP for billed income when billed charge amount is waived	=	C018
Reverse posting from SK P&L when billed charge amount is waived	=	C019
Reverse posting from customer billing account when billed charge amount is waived	=	C020
Settlement of charge amount due to another bank (OBC - Other Bank Charge)	=	C021
SP101 posting for charge amount in charge currency	=	C022
SP101 posting for charge amount in charge booking currency	=	C023
SK P&L for charge amount in charge booking currency	=	C024
SK P&L for charge amount in charge currency	=	C025
Manual posting	=	M001
User defined postings	=	U***

Chapter 10 Switching Customisation Off

This chapter explains how to switch the customisation functionality off, and how to remove customisation from a TI Plus installation.

Switching Customisation Off

Once a TI Plus unit has been customised the presence of the customisation libraries on the classpath is used to determine whether customisation is enabled.

To remove customisation from a customised TI Plus unit, simply run the undeploy ant task from the customisation projects build script, remembering to refer to the correct TI Plus zone directory in your local.properties file.

To reactivate customisation, simply run the deploy ant task again.

Removing Customisation tables from an Existing Database

The following method is for use in test environments only and is only to be used where an existing database unit has been customised and the customisation is no longer required.

In a DB2 environment:

- Open and edit the ExtData.bat script, saving it as remove.bat. Edit out all of the CREATE statements, leaving the CONNECT and DROP statements, then save it. Run this remove.bat script from a DB2 command window

In an Oracle environment:

- Open and edit the oExtData.sql script, saving it as oremove.sql. Remove all the CREATE statements,leaving just the DROP statements. Run the resulting script in the TI script executor, or by using Oracle SQLPLUS)

In an iSeries environment

- Open and edit the iExtData.sql script, saving it as iremove.sql. Remove all the CREATE statements, leaving just the DROP statements. Run the resulting script in the TI script executor

Appendix A Reference Information

This appendix provides reference information.

Public Enumerations

```
public static enum ErrorType {
        Dependancy, DoesNotExist, InvalidInput, MustBeEntered, Other
};

public static enum WarningType {
        Other
};
```

Final Methods

The following final methods are used by the TI Plus application, they provide core functionality which should always be invoked and should never be changed.

public final void **CopyToAPI**() - only found in Posting customisation

public final void **GetFieldCodeAmount**(String fieldCode)

public final void **GetFieldCodeAtIndex**(int index)

public final void **GetFieldCodeBoolean**(String fieldCode)

public final void **GetFieldCodeCount**()

public final void **GetFieldCodeDate**(String fieldCode)

public final void **GetFieldCodeDetails**(String fieldCode)

public final void **GetFieldCodeInteger**(String fieldCode)

public final void **GetFieldCodeString**(String fieldCode)

public final void **GetJournalCount**()

public final void **GetJournalLine**(int idx)

public final void **GetOleControl**()

public final void **GetExtendedString**() - only found in Posting customisation

public final void **GetValidationError**(int idx)

public final void **GetValidationErrorCount**()

public final void **GetValidationWarning**(int idx)

public final void **GetValidationWarningCount**()

public final void **initialise** (EnigmaOLEPane olePane)

public final void **Initialise**(String postingType) – for Posting customisation only

public final void **Initialise**() – for other customisations public final void **PostInitialise** ()

private final void **initialiseFieldCodes**()

public final void **InitialiseMaster**() - for Event customisations only

public final void **Journal**()

public final void **OnSaving**()

public final void **SetDefaults**() - only found in Event customisations.

public final void **SetReadOnly**(Boolean readOnly)

public final void **Validate**()

Overridable Methods

```
public void AddExtraFields() – only found in Posting customisations
public void AddExtraSettlementFields() – only found in Posting
customisations
public void CheckDoNettingOrBulking(String postingTypeString) – only
found in Posting customisations
public void NetWith(String postingTypeString) – only found in Posting
customisations
public void postInitialiseBankAdditionalCode()
protected void initialiseUserFieldCodes()
public void onSavingBankAdditionalCode()
public void UpdateMaster()
public void UpdateMasterOnRelease()
public void validatePane(IValidationDetails validationDetails)
```

Callback Functions (in PDL)

Callback methods are those methods exported by the TI Plus application. These methods can be used in your code to access data from TI Plus or perform calculations provided by the TI Plus application. These methods are exposed through the DriverWrapper objects to the relevant customisation.

PDL Function	Description
String **ConvertAmount** (String argAmount, String argCcy, String flag)	Used to convert an amount to supplied currency form from TI Plus shorthand input.
String **ConvertDate** (String argDateString)	Used to convert supplied date from TI Plus shorthand input.
String **ConvertFromToDBFormat** (String argAmount, String argCcy, String toFrom)	Used to convert an amount to or from the database format where it will be stored as minor denominations.
String **GetAccountOfficers** ()	Invoke the account officer drop-down list in TI Plus.
String **GetAmountInBaseAtFXRate**	Returns the base currency equivalent amount of

PDL Function	Description
(String argFXRateCode, String argCurrency, String argAmount, String argBuySellMid)	the amount passed in argAmount / argCurrency. The amount is converted to base currency equivalent using the fx rate code passed in argFXRateCode / argBuySellMid if passed, otherwise the conversion is via the spot rate.
String **GetAmountInBaseAtSpotRate** (String argCurrency, String argAmount)	Returns the base currency equivalent amount of the amount passed in argAmount / argCurrency using the spot rate
String **GetAmountInCurrency** (String argAmount1, String argCurrency1, String argCurrency2, String argStandardisedRate, String argFXRateCode, String argBuySellMid)	Returns the equivalent amount in currency argCurrency2 of the amount passed in argAmount / argCurrency. The amount is converted to base currency equivalent using the standardised rate passed in argStandardisedRate or the fx rate code passed in argFXRateCode / argBuySellMid, otherwise the conversion is via the spot rate.
String **GetBaseAtFXRate** (String argFXRateCode, String argCurrency, String argBuySellMid)	Returns the normalised exchange rate for the currency passed in argCurrency. The rate is determined by using the fx rate code passed in argFXRateCode / argBuySellMid to retrieve the rate previously entered via Static / FX rates
String **GetBaseAtSpotRate** (String argCurrency)	Returns the normalised spot exchange rate for the currency passed in argCurrency.
String **GetCountries** ()	Invoke the country drop-down list in TI Plus.
String **GetCurrencies** ()	Invoke the currency drop-down list in TI Plus.
String **GetCurrencyAtFXRate** (String argCurrency1, String argCurrency2, String argFXRateCode, String argBuySellMid)	Returns the normalised cross currency rate for the currencies passed in argCurrency1 and argCurrency2. The rate is converted using the fx rate code passed in argFXRateCode / argBuySellMid if passed, otherwise the conversion is via the spot rates.
String GetCustomerFieldAsText (String fieldCode)	Return a specific piece of information related to Customer field fieldCode (refer to the list of Customer/TICustomer Field Codes in this Appendix for all available Customer fields). This callback is only available for the Customer or TICustomer extensions.
String **GetEventFieldAsText** (String argEventFieldCode, String argEventFieldType, String argPart) - for event customisations	Return a specific piece of information related to a given event field code (argEventFieldCode) and part code (argPart) when applicable by making use of the event field mechanism. The second parameter, argEventFieldType, represents the event field type you are looking at and should use one of the constant names given on the list of Event Field Types in this Appendix.
String **GetEventFieldAsText** (String argEventFieldCode, String argEventFieldType, String argPart,	Similar to the same method used in event customisations, but with one more parameter (flag). When flag = M, it is getting information directly from master record; otherwise when flag =

PDL Function	Description
String flag)	E, it will try to get information from event record first then (if it fails to find) will look at the master record. This callback is only available for Event and Posting customisations.
String **GetEventType** ()	Return the event type of the current event. This callback is only available for Posting customisations.
text **GetPostingFieldAsText** (String argCode, String partCode) - for posting customisations	Similar to the method GetEventFieldAsText, but get information via posting. An example of its use is provided earlier.
String **GetProductType** ()	Return the product type of the current transaction. This callback is only available for Posting customisations.
String **GetRawPostingFieldAsText** (String argCode, String partCode)	Similar to the method GetPostingFieldAsText. This callback is only available for Event and Posting customisations.
String **GetTICustomerFieldAsText** (String fieldCode)	Return a specific piece of information related to TICustomer field fieldCode (refer to the list of Customer/TICustomer Field Codes in this Appendix for all available TICustomer fields). This callback is only available for Customer and TICustomer customisations.
String **GetWildCard** (String argCard)	Return the wild card for all in TI Plus when argCard = A (or a), or the wild card for single otherwise.
boolean IsSubsidiary ()	Test whether the current event is a subsidiary event. This callback is only available for Event customisations.
boolean **IsValidInputCode** (String tableName, String inputCode)	Examine whether the user input (variable inputCode) of a data item code in a given database table (represented by the variable tableName) is a valid one. The three tables which are made available for use in this way are the Currency, Country and Account Officer tables, where tableName is Currency, Country and Officer, respectively.
String **LogError** (String argMessage)	Displays an error message for the error passed in argMessage.

Common Functions

Common methods are provided to facilitate the implementation of customisation projects. These methods will not be used by the TI Plus application, but will be used only by the exposed methods.

Function	Description
public void **addError** (ErrorType errorType , String errorText)	Add an error to the validation details. Use the ErrorType enumeration exposed in IValidationDetails.
protected void **addFieldCode** (String fieldCode, String description, FieldCodeType type)	Add a field code definition to the field code details. Use the FieldCodeType enumeration exposed.
public void **addJournal** (String id, String type, String val)	Adds changes to Customer or TICustomer data to the standard TI Plus journal table for reporting and display
public void **addWarning** (WarningType warningType, String warningText)	Add a warning to the validation details. Use the WarningType enum exposed in the IValidationDetails interface.
public static boolean **isAmountIsReallyEmpty** (String amount, String ccy)	Examine whether the money field is empty. This method is part of the utility methods included in CustomisationHelper.
Public Function **CmnGetAmountDescription** (ByRef myPane As Object, _ ByVal strAmount As String, _ ByVal strCurrencyCode As String, _ ByRef strDescription As String) As Integer	This is used to get the description of a money amount.
Public Function **CmnGetAmountString** (ByRef myPane As Object, _ ByVal strAmount As String, _ ByVal strCurrencyCode As String, _ ByRef strConverted As String) As Integer	This is used to convert an input amount string into a valid amount string. For example, if "3t" is the input, the output will be "3,000". The exact format of the resultant amount is defined by the user in TI Plus.
Public Function **CmnGetDateString** (ByRef myPane As Object, _ ByVal strDate As String, _ ByRef strConverted As String) As Integer	Convert an input date string into a valid date string. For example, if "3W" is the input, the output will be a date three weeks from the current unit's processing date. The exact format of the resulting date is defined by the user in TI Plus.

Posting Type Codes

The following enumeration, exposed in AbstractPostingCustomisation, represents the different types of postings:

```
protected static enum PostingType {
      ADV, // Advance
      CHA, // Charge
      COL, // Collateral
      DIS, // Discounted
      EVE, // Event
      FX,  // FX
      FIN, // Financing
      FIR, // FinancingRepayment
      MAN, // Manual
      PAP, // Part Pay
      REP, // Repayment
      PCA, // Periodic Charge Accrual
      FAP  // FinancingAccrual
    };
```

Error Codes

The following enumeration, exposed in IValidationDetails represents the different types of Error codes that are supported by TI Plus:

```
public static enum ErrorType {
            Dependancy,  // there is a dependency
DoesNotExist, // does not exist
InvalidInput, // invalid input
MustBeEntered, // field must be entered
 Other
    };
```

Event Field Types

The following enumeration exposed in FieldCdoeType represents the different types of field code used by customisation.

```
public enum FieldCodeType {

      Account("a"),
      Boolean("l"),
      Branch("b"),
      Date("d"),
      Integer("i"),
      Money("v"),
      Party("p"),
      Reference("r"),
      String("s");

      private String typeString;

      public String getTypeString() { return this.typeString; }
      FieldCodeType(String typeString) {
            this.typeString = typeString;
      }
}
```

Files and Libraries used by Customisation Functionality

The files installed for customisation are of different types.

The following fixed code module is provided for inclusion in the project:

com.misys.tiplus.apps.ti.customisation-base.jar	provides definitions of fields required within all customisation projects and core functional methods used for customisation which are not dependent on the customisation project itself.

The following editor components are used by customisation functionality.

customisti.exe	The main design tool for customisation, providing the ability to define requirements, draw user interface screens and to generate Visual Basic and SQL scripts.
staticdefs.txt	An input file to the customisation editor, providing definitions of which tables can be customised, which products and events can be customised and which standard static tables can be queried within the subsequent Visual Basic project.
customiseti.hlp	A standard Windows help file for the Customisation Editor.

All other components not listed above are generated by the Customisation Editor Save function as follows:

<MajorCode><minorcode>Customisation.jsp	Various jsp pages to provide the user interface functions required by the customisation project. One page is output for each pane designed in the customisation editor.
ExtData.bat	DB2 SQL script to create the extra data tables and fields. This script is used to originally create the required database changes. It should be run subsequently as it may result in the loss of entered business data, instead an upgrade method should be followed (see separate upgrade chapter)
oExtData.sql	Oracle version of the above.
iExtData.sql	IBM iSeries version of the above.
<project-name>.txt.xml	The output from the customisation editor which is input to the Java build and deploy processes
<project-name>.cnv	A text file that when integrated with the data model viewer will show the extra data tables and fields within the viewer
<project-name>.xml	(Reserved for future use.)

Appendix B Upgrading a TI Plus Installation that has been Customised

The following steps are required when upgrading TI Plus from one release to another, even if you are not changing your extra data requirements. If you wish to add extra fields to your customisation at the same time as upgrading your version of TI Plus, we recommend you carry out the following steps first to upgrade your existing customisation, prior to carrying out the steps described in Appendix C to change your customisation.

In a customised TI Plus unit extra tables and fields are used that are not used by standard TI Plus. These tables all have table names beginning 'EXT' and are defined and set up using functions provided in the Customisation Editor.

Following the running of the standard TI Plus upgrade scripts to bring the database up to the new required level the following instructions must be carried out. These steps are required even if the customisation is not changing. It is not possible to continue to use the old customisation which must be rebuilt to function with the new release.

Misys strongly recommend that the following approach is taken.

First copy the whole customisation project directory to a new directory. Leave all the sources and objects of the live project alone. All the following steps are then applied to the files in the new directory.

Delete the customisation library (it will be regenerated later).

Rename the customisation project file.

Identify, then copy and paste to new files any hand-written code, preferably by creating a readme.txt file describing the extra code to others.

Create a copy of the live database to provide a test unit.

Open the Customisation Editor, then open the new project's .txt file.

Use the Save function.

Use the special function 'Build upgrade script' provided in the Customisation Editor. This function will compare old and new definitions of the customisation by comparing the project TXT files and create an upgrade script. It works by comparing the old customisation with the new project. This step is required even if you are not changing any of the extra data tables and fields. The reason for this is that the underlying housekeeping and control fields may have changed and the upgrade script created will take this into account in generating the script. The definition of the housekeeping fields is contained in an installed file called STATICDEFS.TXT and this function will ask you to specify the location of the previous version of this file so it can compare with the current version and generate a script of the differences.

Run the script using the function CONVTIDB (this software is part of the CSNT installation normally in the TI Plus client/server control system directory). The function must be given the target database as a parameter. The function will offer to take a database backup prior to running the script, to which you should respond 'Yes'. The function will also take an unconditional backup of the database after the script has run.

Ensure the customisation new project .txt.xml file is copied into your eclipse workspace under the project directory. Run the ant script's generate task. If any of the regenerated code from your new project does not compile with your old source code, edit any outdated references to old fields etc until the source compiles. Now run the ant script's deploy task.

Upgrading from Trade Innovation to TI Plus

TI Plus includes facilities to help you convert a Trade Innovation customisation to run on TI Plus. These are described in the section "Adding a User Interface Control to the Interface".

Your bank is expected to review the converted layout and user interface before rebuilding the existing project. In particular, since TI Plus supports only a four-column layout, your customisation interface will need to be redesigned if it uses more than four columns.

Appendix C Changing your own Extra Fields in a Subsequent Project

If you already have a customised TI Plus system and need to update it, then if the extra fields or tables have changed in any way, you will need to update the TI Plus database accordingly. This is done in the following way:

1. Build a database conversion script (see page 80).
2. Make a backup of the database.
3. Run the conversion script. If any problems occur during the running of this script, you must restore from the backup before attempting to continue.

For any modified table the following rules apply:

1. If a column name exists in the definition of the current database but not in the definition of the new database, then the column is dropped
2. If a column name exists in the definition of the new database but not in the definition of the current database, then the column is added. The column value in existing rows is set to the initial value, if one has been specified. If no initial value has been specified, then the value is set to null
3. If a column name exists in both the old and new database definitions, and the column definitions are identical, then the column and all column values are retained
4. If a column name exists in both the old and the new database definitions, and the column definitions are different, then one of three things will happen:
    - The column values are retained
    - The column values are retained if possible, but truncation may occur. An error message will be displayed, and a suitable comment inserted into the conversion script
    - No conversion is possible between the two data formats. An error message will be displayed, and a suitable comment inserted into the conversion script. The column values in the database are discarded, column is treated as if it were a new column

Where the data definitions of a column are different, then existing data (possibly truncated) will be retained for the following cases.

Original type	New types	Comments
CHAR	CHAR VARCHAR LONGVARCHAR	
VARCHAR	CHAR VARCHAR LONGVARCHAR	May truncate Column values retained
LONGVARCHAR	CHAR VARCHAR	May truncate
INTEGER	NUMERIC DECIMAL FLOAT	May truncate Column values retained
NUMERIC	INTEGER DECIMAL NUMERIC FLOAT	May truncate Column values retained
DECIMAL	INTEGER NUMERIC DECIMAL FLOAT	May truncate Column values retained
FLOAT	INTEGER NUMERIC	May truncate

Original type	New types	Comments
	DECIMAL	

Building a Database Conversion Script

The functionality described in this section will compare old and new definitions of staticdefs.txt and create an upgrade script. It works by comparing the old customisation with the newly generated project. This step is required even if you are not changing any of the extra data tables and fields. The reason for this is that the underlying housekeeping and control fields may have changed and the upgrade script created will take this into account in generating the script. The definition of the housekeeping fields is contained in an installed file called STATICDEFS.TXT and this function will ask you to specify the location of the previous version of this file so it can compare with the current version and generate a script of the differences.

You will need a database conversion script if you have a customised system that you wish to upgrade and:

- The updated system includes database changes, and
- The customised system contains customisation data that you wish to preserve

When a conversion script is run it will log this fact, with some data recording details of the conversion. Some of this data must be supplied when the script is built.

A database conversion script should be produced in the following way:

Start the Customisation Editor.

Open the file for the new customisation. (This defines the new customisation-related database structure).

From the File menu select Build Upgrade Script. This opens a dialog box with the fields shown in the following table. Enter the correct values into these fields, then press OK.

From/To	This is included in the log records for the conversion. It should identify the TI Plus version being updated
Script Name	This is the name of the customisation script
Old Customisation File	This is the file which defines the customisation that corresponds to the current database state. A full path need not be supplied if this file is in the same directory as the new customisation file
Comments	This is included in the log records for the conversion. You may add any comment required to identify the new customisation.

You can select the Build Upgrade Script menu option after loading a customisation definition and then making changes to the database definition. Any script made at this point will work correctly, but there is a danger that the log information will be misleading, if you do not subsequently save the changes to the file from which you loaded.

Misys strongly recommend that you build upgrade scripts as described above, and use a different file name when saving any further changes to the customisation.

Run the script as follows:

For DB/2	Use the function CONVTIDB (this software is part of the CSNT installation normally in the TI Plus client/server control system directory). The function must be given the target database as a parameter. The function will offer to take a database backup prior to running the script, to which you should respond 'Yes'. The function will also take an unconditional backup of the

	database after the script has run.
For ORACLE	Use the TI database script executor which will ask for a logon to the ORACLE unit name, user ID and password, then request the name of the script. For ORACLE this must be of the form oxxxxx.sql where xxxx can be anything and the initial 'o' tells the script runner that the target is ORACLE.
For iSeries	Use the TI database script executor which will ask for a logon to the i-series unit name, user ID and password, then request the name of the script. For iSeries this must be of the form ixxxxx.sql where xxxx can be anything and the initial 'i' tells the script runner that the target is iSeries.

Appendix D Invoking Chart - TI Plus to the Java Code

This document has covered all the main Visual Basic methods (exposed methods) which reside in the customisation library and wait to be called by the TI Plus application.

Initialisation

Below is a summary of initialisation of customisation in TI Plus for the different types of extra data:

Customer	**Automatic when obtaining customer details.**
Event	Automatic when obtaining the event details when the system option CustomisedDataInSeparatePane is not used. Via the Extra Data button when the system option CustomisedDataInSeparatePane is used.
Master	Automatic when obtaining the master details when the system option CustomisedDataInSeparatePane is not used. Via the Extra Data button when the system option CustomisedDataInSeparatePane is used.
TICustomerDetails	Automatic when obtaining TI customer details.
Postings	Using the Release Items|Postings link during transaction processing.
Static data fields	Using the static data maintenance application's Trade|Customised Codes menu option.

When the system option CustomisedDataInSeparatePane is used, initialisation is via the Extra Data button for masters and events:

Exit from TI Plus User Interface

There are some differences in processing between pressing OK from the Customer (or TICustomer) pane and from the event pane.

Appendix E Static Data from the Back Office to TI Plus

If you wish to import customer data from back office into TI Plus then the following example includes importing customised extra data for customers with it.

The system requires you to send all the customisation extra fields in the XML message even though they are flagged as optional in the schema. This limitation will be removed in a later release.

```
<?xml version="1.0"?>
<ServiceRequest xmlns="urn:control.services.tiplus2.misys.com"
    xmlns:c="urn:common.service.ti.apps.tiplus2.misys.com"
    xmlns:m="urn:messages.service.ti.apps.tiplus2.misys.com"
    xmlns:u="urn:custom.service.ti.apps.tiplus2.misys.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <RequestHeader>
        <Service>TI</Service>
        <Operation>Customer</Operation>
        <Credentials>
            <Name>SUPUSER</Name>
            <Password>SUPUSER1</Password>
        </Credentials>
        <NoRepair>Y</NoRepair>
        <NoOverride>Y</NoOverride>
        <TransactionControl>NONE</TransactionControl>
    </RequestHeader>
    <m:Customer>
        <m:MaintType>A</m:MaintType>
        <m:Mnemonic>PARTON</m:Mnemonic>
        <m:CustomerNumber>563457</m:CustomerNumber>
        <m:CustomerType>AA</m:CustomerType>
        <m:FullName>PARTON INDUSTRIES SA </m:FullName>
        <m:ShortName>PARTON</m:ShortName>
        <m:Reference>PAR001</m:Reference>
        <m:Branch>MBHO</m:Branch>
        <m:MailToBranch>MBHO</m:MailToBranch>
        <m:Group>ACHETE</m:Group>
        <m:AccountOfficer>TFD</m:AccountOfficer>
        <m:ResidenceCountry>FR</m:ResidenceCountry>
        <m:ParentCountry>FR</m:ParentCountry>
        <m:RiskCountry>FR</m:RiskCountry>
        <m:AnalysisCode>Q1</m:AnalysisCode>
        <m:Language>FR</m:Language>
        <m:PrincipalFXRateCode>IBANK</m:PrincipalFXRateCode>
        <m:ChargeFXRateCode>IBANK</m:ChargeFXRateCode>
        <m:Closed>N</m:Closed>
        <m:Blocked>N</m:Blocked>
        <m:Deceased>N</m:Deceased>
        <m:Inactive>N</m:Inactive>
        <m:AllowTaxExemptions>Y</m:AllowTaxExemptions>
        <m:MidasFacilityAllow>N</m:MidasFacilityAllow>
        <m:OtherDetails>
            <c:Gateway>Y</c:Gateway>
            <c:SWIFTAckRequired>N</c:SWIFTAckRequired>
        </m:OtherDetails>
        <m:CustomerExtraData>
            <u:Sort_Code>009878</u:Sort_Code>
            <u:Town_Name>Sevenoaks</u:Town_Name>
            <u:Common_System_ID>123456789</u:Common_System_ID>
            <u:BO_CIF/>
            <u:EU_AREA/>
            <u:REG_Area/>
            <u:Profit_Centre/>
            <u:GBLCI/>
            <u:GCIDTE/>
            <u:CICDE/>
            <u:CCDAYS>1</u:CCDAYS>
            <u:TCDAYS>1</u:TCDAYS>
            <u:POSTEX/>
            <u:SNDSWIFT/>
            <u:EUEXEMPT/>
            <u:REPRSENT/>
            <u:UNPDSWFT/>
            <u:UNPCHGE/>
            <u:UNPSWIFT/>
            <u:CCYUNP/>
            <u:IDA/>
            <u:IDB/>
            <u:MergeCustomer/>
            <u:IDEXT1/>
            <u:IDEXT2/>
            <u:LCRiskLoading>1.0</u:LCRiskLoading>
            <u:GtyRiskLoading>1.0</u:GtyRiskLoading>
            <u:AccountNumber/>
            <u:SICcode/>
            <u:BusinessArea/>
            <u:CHECK_LIMIT/>
        </m:CustomerExtraData>
        <m:TICustomerExtraData/>
    </m:Customer>
</ServiceRequest>
```

All the customised fields are in the urn:custom.service.ti.apps.tiplus2.misys.com namespace. A shortname of u is provided for this namespace; hence the line:

```
xmlns:u="urn:custom.service.ti.apps.tiplus2.misys.com"
```

at the head of the message.

Each extra field is identified by its logical name as defined in the customisation project.

Index