QAD Enterprise Applications
Enterprise Edition

User Guide

QAD Integrated Customization Toolkit

Introduction
QAD ICT Functions and Procedures
QAD ICT Processing
QAD ICT Restrictions and Example Customizations

70-3261-4.1.22
QAD ICT 4.1.22
December 2016

IntegratedCustomizationToolkit_UG_v04122.pdf/mat/mat

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 566-6000
http://www.qad.com

Contents

QAD ICT User Guide Change Summary

The following table summarizes significant differences between this document and previous versions.

Date/Version	Description	Reference
December 2016/QAD ICT 4.1.22	Rebranded for QAD ICT 4.1.22	--
March 2015/QAD ICT 4.1.20	Added description of Compile field in File Maintenance	page 55
	Added description of ICT Setup By Function Report	page 69
	Added description of ICT Setup By Table Report	page 70
	Added description of how wildcards can be used to select functions	page 78
	Other minor changes	
April 2014/QAD ICT 4.1.18	Minor updates	throughout
October 2013/QAD ICT 4.1.16	Minor updates	throughout
	Updated information on Frame & Field Properties Maintenance	page 29
	Added information to UI Triggers and Functions Restrictions table	page 82
	Added topic regarding compliance with SOX	page 83
	Provided additional information In Adding Custom Fields to a Standard Frame section	page 90
October 2012/QAD ICT 4.1	Numerous minor updates	throughout
	Added new appendix with function examples for QAD 2011 EE and higher	page 107
May 2012/QAD ICT 4.0	First formal release of document	

Chapter 1

Introduction

This chapter provides a brief overview of QAD Integrated Customization Toolkit (QAD ICT).

Overview

The QAD Integrated Customization Toolkit (QAD ICT) allows customizations to be designed and developed in a non-intrusive way, eliminating or limiting required changes to the original standard applications. As a result, custom development is faster to create and reduces support costs, while retaining high quality standards. In addition, QAD ICT provides flexibility to reuse customized functionality in other areas of your business and increases portability, making it easier to migrate to new releases of QAD Enterprise Applications.

Typically, you encounter customization issues in the following areas:

- System interface. Adding simple and complex validations, making whole frames or selected values invisible for selected users, setting whole frames or selected fields as read-only for selected users, defining initial/default values for selected fields, adding new frames to the existing maintenance programs, adding new fields to already existing frames.
- Database schema. Adding “shadow” tables.
- Processing. Adding new processing, changing how the system currently processes data.
- New queries and reports.

Customizations are ordinarily connected with changes to source code. The obvious disadvantages of that are:

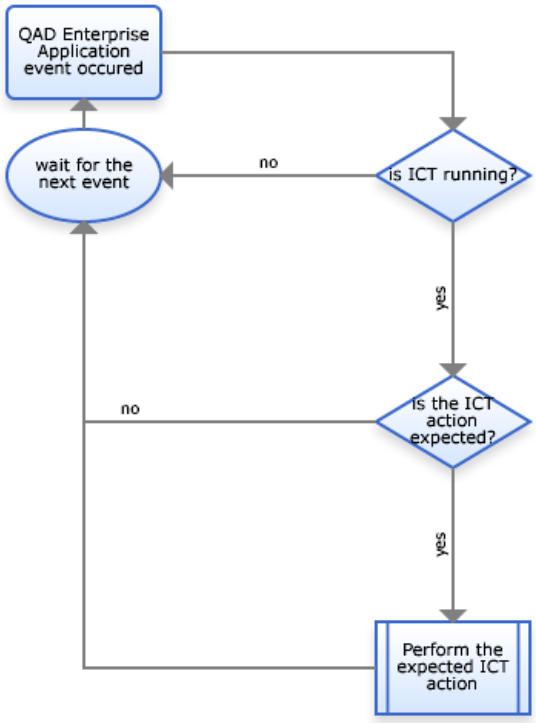
- Permanency. Changes are usually permanent; it is impossible to switch them on and off.
- Access to source code. Necessity of having the full set of system source code. (The standard QAD installation includes only source code for reports and inquiries.)
- Difficulty with installing new patches. With intrusive customizations, the process of installing new patches becomes difficult and time consuming.
- Expensive upgrades. Extra work time is necessary to upgrade customizations to the higher system version or service pack.
- Many other disadvantages.

The QAD ICT solution helps you avoid such problems. Customizations:

- Are much simpler
- Avoid the need to have source codes necessity
- Are significantly faster
- Give you absolute control (ICT Debugger and Reports)
- Are non-intrusive and system version/service pack independent
- Repeatedly cost less

Main Processing Flow

The following figure summarizes the main flow of ICT processing.



Chapter 2

QAD ICT Functions and Procedures

This chapter provides function and procedure examples for Standard Edition and earlier versions of Enterprise Edition. Examples for Enterprise Edition 2011 and higher are provided in Appendix A on page 107.

Overview

This chapter describes how the developer can use the functions and procedures delivered with QAD ICT. These functions and procedures can be used in the same way as any other Progress functions such as string, logical, and so on.

Each function or procedure is defined in an include file called `icdef.i`. To be able to use the function or procedure in a custom-developed program, the definition of the include file has to be specified in the custom procedure, as in the following example:

```
/*$FileName - Trigger Link Custom Procedure                    */
/* Generated by QAD ICT source codes generator                 */
/*V8:ConvertMode=Maintenance                                   */

{mfdeclre.i}
{icdef.i}
```

Important Function examples presented in this chapter cannot be used in QAD Enterprise Edition 2011 and higher. You can find examples for those versions in Appendix A on page 107.

Register Functions

Use register functions to register a definition or value in QAD ICT.

registerTrigger(definition)

Use this function when you define a new trigger in QAD ICT. When the new trigger is registered, the definition is visible in QAD ICT Reporting. To keep track of all triggers defined in the system, the developer has to register all definitions using the *registerTrigger* function. One parameter must be passed in the function: a string specifying the type of trigger and type of table.

Example

```
{mfdeclre.i}
{icdef.i}

registerTrigger(“FIND pt_mstr”).
on find of pt_mstr do:
    disable triggers for dump of cp_mstr.
    define variable temp_value as character.
    …
end.
```

registerRecord(identifier,rowid)

Use this function to register a new record in QAD ICT. When the record is found (for example, using a find statement) and registered using the *registerRecord* function, the developer can retrieve that record in any other program using the *getRecord* function (see page 9). Parameters that should be passed in the function are identifier (a string that specifies the name of the table) and rowid (rowid to the record that should be registered).

Example

```
{mfdeclre.i}
{icdef.i}

registerTrigger(“FIND pt_mstr”).
on find of pt_mstr do:
    registerRecord(“pt_mstr”,rowid(pt_mstr)).
```

```
end.
```

registerBuffer(identifier,handle)

Use this function to register the currently processed buffer. For example, when pt_mstr is available, you can register it in QAD ICT and later retrieve it for further use with the *getBuffer* function.

Example

```
{mfdeclre.i}
{icdef.i}

find first pt_mstr no-lock no-error.
if available(pt_mstr) then
    registerBuffer(“pt_mstr”, buffer pt_mstr:handle).
```

Note A buffer pointed by a handler may have been released (function release). The buffer is available for the *registerBuffer* function; however, the returned handler is empty.

registerValue(variable_name,character_value)

Use this function to register a new value in QAD ICT. When a value is calculated (stored in a variable) and registered using the *registerValue* function, the developer can retrieve that value in any other program using the *getValue* function (see page 9). Parameters that should be passed in the function are variable (a string that specifies the name of the variable that stores the value) and character value (a string with the value that should be registered).

Example

```
{mfdeclre.i}
{icdef.i}

define variable lvc_variable as character no-undo.
assign lvc_variable = “value that should be stored”.

registerValue(“lvc_variable”, lvc_variable).
```

registerPersistent(program,handle)

Use this function to register a new persistent procedure in QAD ICT. When the procedure is executed (using, for example, `gprun.i`) the developer can register that procedure as a persistent one (all internal procedures or triggers defined in the procedure will be available during the session). Parameters that should be passed in the function are program (a string that specifies the name of the program) and handle (handle to the procedure that should be set as persistent).

Example

```
{mfdeclre.i}
{icdef.i}

define variable lvhHandle as handle no-undo.
{gprun.i ““icuipers.p”” “persistent set lvhHandle”}

registerPersistent(“icuipers.p”, lvhHandle).
```

registerSuper(program,handle)

Use this function to register a new super program in QAD ICT. The developer can register that program as a super program (all internal procedures or triggers defined in the program will be available during the session). Parameters that should be passed in the function are program (a string that specifies the name of the program) and handle (handle to the program that should be set as persistent). The difference between *registerPersistent* and *registerSuper* is that when using *registerPersistent* the developer who wants to run an internal procedure from a persistent one must specify the handler to the procedure where the internal one is located. Using a *registerSuper*, the system will execute the one that was loaded last.

Example

```
{mfdeclre.i}
{icdef.i}

…

if not (qad_key5 = "" or qad_key5 = gsvICTTaskID) then next.

if session:add-super-procedure(getProcHandle(qad_key2, false)) then do:
    if existsInSuper("HandleEvent") then
        registerSuper(qad_key2, getProcHandle(qad_key2, false)).
end.
```

registerWidget(program,widget,handle)

Use this function to register a widget in QAD ICT. When a new widget is created or retrieved during processing, the developer can register that widget. Parameters that should be passed in the function are program (a string that specifies the name of the program), widget (the name of the widget), and handle (handle to the widget that should be registered).

Example

```
{mfdeclre.i}
{icdef.i}

create fill-in lvwWdg
assign
    name                = entry(3,ttic_key2)
    frame               = lvhFrameHdl
    data-type           = ttic_charfld[2]
    format              = ttic_charfld[4]
    col                 = FCOL(ttic_charfld[5],ttic_charfld[6])
    row                 = FROW(lvhFrameHdl,ttic_charfld[5],ttic_charfld[6])
    sensitive           = no
    .

registerWidget (entry(1,ttic_key2),
                lvwWdg:name,
                lvwWdg:handle).
```

registerShadow(identifier,rowid)

Use this function to register a shadow table in QAD ICT. When the record of a shadow table is found (for example, using a find statement) and registered using the *registerShadow* function, the developer can retrieve that record in any other program using the *getShadow* function (see page 10). Parameters that should be passed in the function are identifier (a string that specifies the name of the table) and rowid (rowid to the record that should be registered).

Example

```
{mfdeclre.i}
{icdef.i}

registerTrigger(“FIND xxpt_mstr”).
on find of xxpt_mstr do:
    registerShadow(“xxpt_mstr”,rowid(xxpt_mstr)).
end.
```

Retrieve Functions

Use retrieve functions to retrieve a value or record that was previously registered in QAD ICT.

getRecord(identifier)

Use this function to retrieve a record that was previously registered using the *registerRecord* function. One output parameter is passed back by the function: a rowid to the record. The input parameter for the function should be identifier (a string that determines which record should be retrieved).

Example

```
{mfdeclre.i}
{icdef.i}

define variable i_rowid as rowid.

assign i_rowid = getRecord(“pt_mstr”).
for first pt_mstr where rowid(pt_mstr) = i_rowid exclusive-lock:
end.
```

getBuffer(identifier)

Use this function to retrieve the handle to the previously registered buffer. For example, when the pt_mstr buffer was registered previously, you can retrieve the handle to it using the *getBuffer* function.

Example

```
{mfdeclre.i}
{icdef.i}

define variable iphBuffer as handle no-undo.
define temp-table ttpt_mstr no-undo like pt_mstr.

iphBuffer = getBuffer(“pt_mstr”).
create ttpt_mstr.
Buffer ttpt_mstr:buffer-copy(iphBuffer).
```

getValue(variable_name)

Use this function to retrieve a value that was previously registered using the *registerValue* function. One output parameter is passed back by the function: a character value (value of the variable). The input parameter for the function should be variable (a string that determines the variable name from which the value should be retrieved).

Example

```
{mfdeclre.i}
{icdef.i}
```

```
define variable lvc_variable as character no-undo.
registerValue(“my_value”, “test”).
…
lvc_variable = getValue(“my_value”).
```

getBufferValue(buffer,field,index)

Use this function to retrieve a particular value from the buffer. One output parameter is passed back by the function: a character value (value of the variable). The input parameters for the function should be buffer, field (a string that determines the field name from which the value should be retrieved), and index (expression representing a subscript, for fields that have extents). For a field that has no extents, the index should equal 0 (zero).

Example

```
{mfdeclre.i}
{icdef.i}
find first pt_mstr no-lock no-error.
registerBuffer("pt_mstr", buffer pt_mstr:handle).
display getBufferValue("pt_mstr", "pt_um", 0).
```

getShadow(identifier)

Use this function to retrieve the record of a shadow table that was previously registered using the *registerShadow* function. One output parameter is passed back by the function: a rowid to the record. The input parameter for the function should be the identifier (a string that determines which record should be retrieved).

Example

```
{mfdeclre.i}
{icdef.i}
define variable i_rowid as rowid.
assign i_rowid = getRecord(“xxpt_mstr”).
for first xxpt_mstr where rowid(xxpt_mstr) = i_rowid exclusive-lock:
end.
```

Screen (Widget Walker) Manipulation Functions and Procedures

The developer can use one function and one procedure to manipulate the screen.

getFieldValue(variable,frame,property)

Use this function to retrieve information from the screen. One output parameter is passed back by the function: a character value. In that value, the function can pass back the value, label, or a handle to the field or frame, depending on the input parameters.

The input parameters for the function can be:

- Variable. String that determines the name of the field that is displayed on the screen.
- Frame. Name of the frame in which the field is displayed.

- Property. The string that determines what kind of information should be retrieved. The property can be SCREEN-VALUE, LABEL, or HANDLE. (Using a handle, you can retrieve all information for the field; it is very flexible.)

Example

```
{mfdeclre.i}
{icdef.i}

define variable i_handle as widget-handle no-undo.
define variable lvc_value as character no-undo.

assign i_handle = widget-handle(getFieldValue(“pt_part”,”a”,”handle”)).
assign lvc_value – i_handle:format.
```

setFieldValue(variable,frame,screen-value)

Use this procedure to set the value in a field that is currently displayed on the screen. The value is only set on the screen—not in the database.

The input parameters for the procedure should be:

- Variable. String that determines the name of the field that is displayed on the screen.
- Frame. Name of the frame in which the field is displayed.
- Screen-value. The string that determines the value that should be specified in the field.

Example

```
{mfdeclre.i}
{icdef.i}

run setFieldValue(“pt_um”, “a”, “kg”).
```

Note When the field that is displayed on the screen was previously added using Frame & Field Properties Maintenance, the developer must define the first parameter as table_name.field_name to set the value in it. In the example shown above, if pt_um is added using Frame & Field Properties Maintenance, the execution of the function should contain the first parameter as:

```
run setFieldValue(“pt_mstr.pt_um”, “a”, “kg”).
```

Additional Functions and Procedures

firstTime(identifier)

This function allows you to execute the block of custom code only once for each record. The input parameter for the function/procedure should be the identifier (string value) for the currently processed record. The output parameter is a logical value (true or false) that indicates if the *saveFirstTime* procedure was executed for the current record.

Note When using *firstTime* for custom pop-up frames, developers should be aware that if there is a validation error somewhere in standard code, the pop-up frame will not appear again and previously filled-in values will not be stored.

saveFirstTime(identifier)

This procedure allows you to save the event when some block of custom code was executed for the currently processed record. To use this procedure for a particular record, the proper table must have been added to Trigger Link Maintenance (90.1.1).

Example

```
{mfdeclre.i}
{icdef.i}

if firstTime(“pt_mstr”) then do:

    display pt_um.
    saveFirstTime(“pt_mstr”).
end.
```

enableCallerFrame(which_fields)

This function has been deprecated. It is included for backward compatibility only.

disableCallerFrame(*|which_fields)

This function has been deprecated. It is included for backward compatibility only.

popup_frame(name)

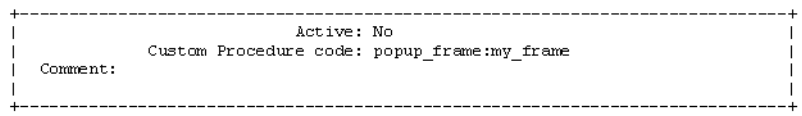
This procedure displays a pop-up frame defined in Pop-up Frame Maintenance.

Example

The *popup_frame* procedure can be executed from the custom code:

```
{mfdeclre.i}
{icdef.i}

run popup_frame(“my_frame”).
```

The *popup_frame* procedure can be also executed from UI Trigger Maintenance or Program Hook Maintenance (“popup_frame:” is a fixed text).



Chapter 3

QAD ICT Processing

This chapter describes the programs used to set up and perform QAD ICT processing. Information is structured by menu location.

Overview

The key role of the customization toolkit is processing based on memory-resident customization parameters. QAD ITC's role is to monitor all events that occur in the system and react by performing additional actions if requested. If QAD ICT is installed, the process starts with the QAD Enterprise Applications session and resides persistently in the system memory. To make all analyses faster, the QAD ICT process loads all defined customizations parameters to its memory. After that time, if an application event occurs, QAD ICT analyzes it, determines if any action is required, and reacts as needed. Two kinds of events can be defined: interface events and database events.

Interface Events

Host-based and client/server applications typically interact with database data directly within the user interface. QAD ICT can react to interface events generated by QAD Enterprise Applications by:

- Adding dynamically generated fields to the standard frames
- Hiding visible frames and fields
- Displaying editable frames and fields as read-only
- Changing the format, width, or position of standard fields
- Setting default values for selected fields
- Imposing new validations on existing fields
- Adding new messages to the application that include information about the process
- Running additional internal or external 4GL programs
- Performing other actions

Database Events

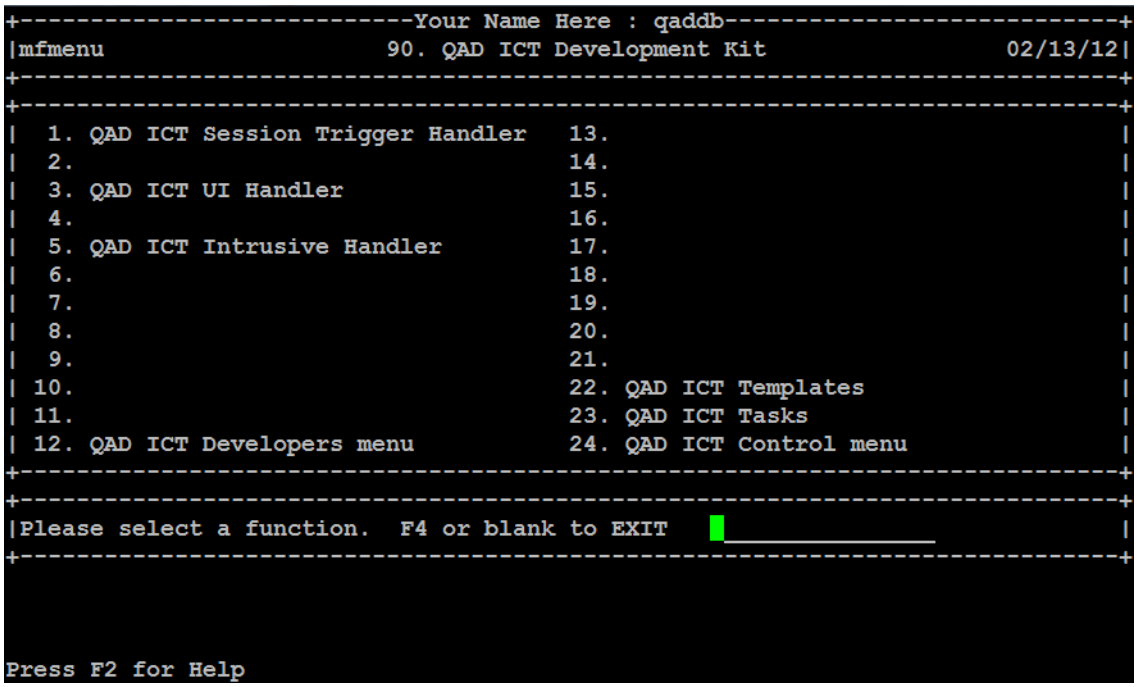
Customizations created by QAD ICT can subscribe to particular events in the database. The following actions can be executed in response to database events:

- Registering the record ROWID
- Finding the shadow table if it exists
- Automatically deleting associated shadow tables records when a standard table record is deleted
- Running additional internal or external 4GL programs when necessary
- Performing other actions

QAD ICT Developers Menu

The QAD ICT Developers Menu lets users create their own customizations. Two versions of the Developers Menu are provided in the character UI:

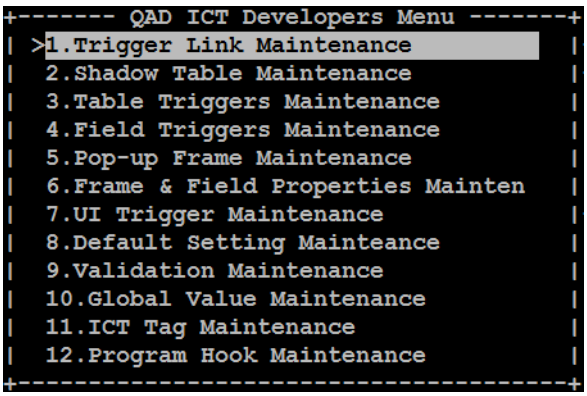
- A standard menu (in the most installations) is provided on menu 90.



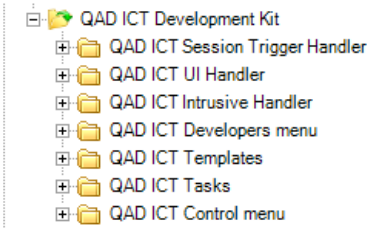
- The Developers Menu can be called directly from anywhere in the application by pressing Ctrl+the selected key (menu accelerator). Key settings such as the program name and the frame and field from which the Developers Menu was called are copied to the QAD ICT maintenance programs and used as defaults.

  Note The menu accelerator (or shortcut) is user configurable. See “Changing the ICT Development Menu Accelerator” on page 17 for details.

Note The menu accelerator is available in character and GUI versions. It is unavailable in the QAD .NET UI/Desktop interface, but you can create a favorites menu that includes the QAD ICT Developers Menu entries.



In .NET UI, the QAD ICT Developers Menu is installed in the QAD ICT Development Kit folder.



This chapter includes descriptions of the menu functions.

Scenario Examples

Scenario 1. Character UI and GUI only

1. From the QAD Enterprise Applications menu, select the program that is to be customized or that includes items to be customized.
2. Move the cursor to the frame or field that needs to be changed.
3. Use the menu accelerator. The Developer Menu should appear.
4. Select an ICT Development Menu option. If you are customizing the programs, frames, fields, or user interface events, all parameters related to the customized program, frame, and field are copied to the selected maintenance program. You do not need to re-enter them.
5. Define required settings.
6. Exit from the ICT option.
7. Exit from the ICT Development menu.
8. Answer Yes or No when prompted to rebuild ICT processing.

Note Each customization to be considered by the ICT processing must be activated and the QAD ICT processing has to be restarted.

Note After leaving the ICT option and customized program by pressing ENDKEY, the transaction you started to let you access the customized frame or field will be undone—**except for the defined ICT settings.**

Scenario 2. All UIs

1. Select an ICT option from the 90 menu (Character UI and GUI) or QAD ICT Development Kit in .NET UI/Desktop.
2. Fill in all mandatory fields and optional fields (if necessary).
3. Exit from the ICT option.
4. Restart ICT processing to load your customization.

Note Each customization to be considered by the ICT processing must be activated and the QAD ICT processing has to be restarted.

Restarting ICT Processing

To restart ICT processing in character and GUI, you can use Reload QAD ICT in ICT Control Table (90.24.24), run the procedure called `icreload.p`, or log off and log on again to the QAD application.

Note `icreload.p` works only in Standard Edition. In Enterprise Edition, you cannot run procedures that are not on the menu.

In .NET UI/Desktop, you must restart the Connection Manager.

Changing the ICT Development Menu Accelerator

1. Set the ICT Menu Accelerator CTRL field in ICT Control Table to the letter you want to use.
2. If `icacc`*`X`*`.p` does not exist for the selected letter, create an `icacc`*`X`*`.p` procedure (*X* is your selected letter; for example, `icaccw.p` for "w"). The `icacc`*`X`*`.p` procedure should be created as follows:
   - {`icacc2.i CTRL-X`} for QAD Enterprise Applications Standard Edition and Enterprise Edition earlier than EE 2011.
   - {`us/ic/icacc2.i CTRL-X`} for QAD Enterprise Applications Enterprise Edition 2011 and higher.
3. Compile the procedure.

   Note ICT provides `icacco.p`, `icaccp.p`, and `icaccr.p` procedures with ICT installation.
4. Reload ICT processing.

QAD ICT Session Trigger Handler Menu

Use the programs on the QAD ICT Session Trigger Handler (90.1) menu to set up session trigger processing.

Trigger Link Maintenance (90.1.1)

Trigger Link Maintenance allows QAD ICT programmers to select a QAD Enterprise Applications database table and add it to the list of tables controlled by QAD ICT. After that, each record event that is found will automatically:

- Be registered. QAD ICT registers the ROWID of each found record and makes it available (on request; see "getRecord(identifier)" on page 9).
- Be available for custom procedures. QAD ICT programmers can handle the executed event with their custom procedures.
- Link master and shadow records. The shadow record is automatically searched when related to this master table; the main and shadow table records are both registered.

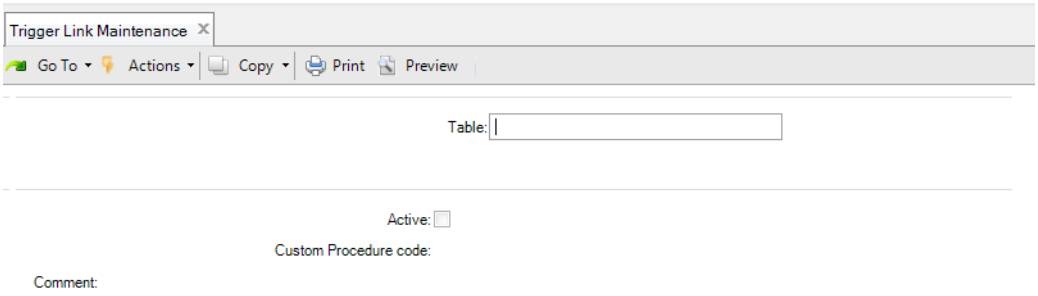


Table. Name of table on which trigger link is defined. Mandatory.

Active. A logical value that defines whether the trigger link is active. Yes means that the trigger link is active. Mandatory.

Custom Procedure code. For backward compatibility only. In QAD ICT 2.0 and higher, Subscribers Maintenance should be used.

A field to enter a procedure name that will be executed when the trigger link is fired. This additional custom procedure code must be defined in any of the (custom) QAD ICT persistent programs or can be a normal program that is not defined as persistent. Optional.

Comment. Space for additional information, such as the reason for activating the trigger link. Optional.

Using the Program

1. Run Trigger Link Maintenance using the standard QAD Enterprise Applications system menu or QAD ICT Development menu.
2. Select an already-supported table to change its parameters, or enter the name of a new table.

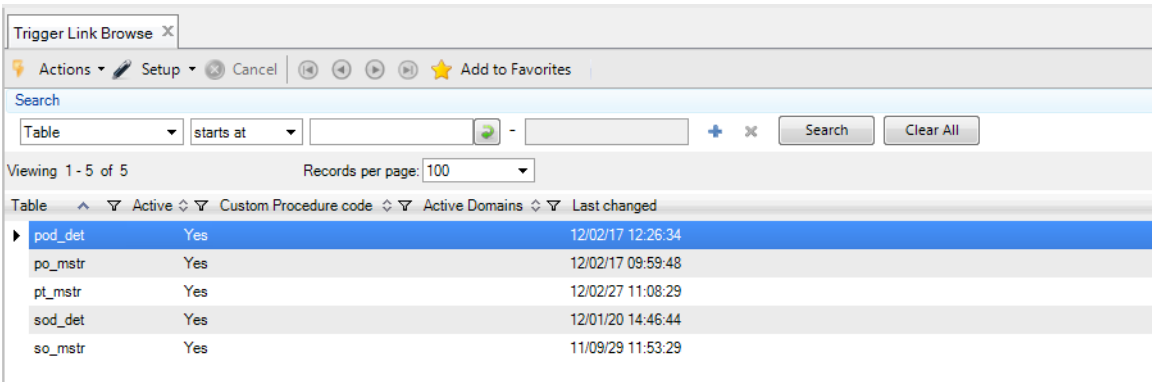
   Note The table must exist in the database dictionary of the connected database. Otherwise, the system displays an error message.
3. Make the trigger link active/inactive.
4. Exit the Trigger Link Maintenance program.
5. Respond to the prompt to rebuild link and consistency triggers.

   Note During the rebuild process, source code for new triggers is automatically generated, placed in a session working directory, and compiled into the working directory keeping the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`. To make your customization visible for other users, you should move all generated source and r-code to the central environment.

   See “QAD ICT Tasks” on page 52 for details about how to manage your customizations.
6. Restart QAD ICT to load your customization.

Trigger Link Browse (90.1.2)

Trigger Link Browse is a standard browse for trigger links defined in Trigger Link Maintenance.

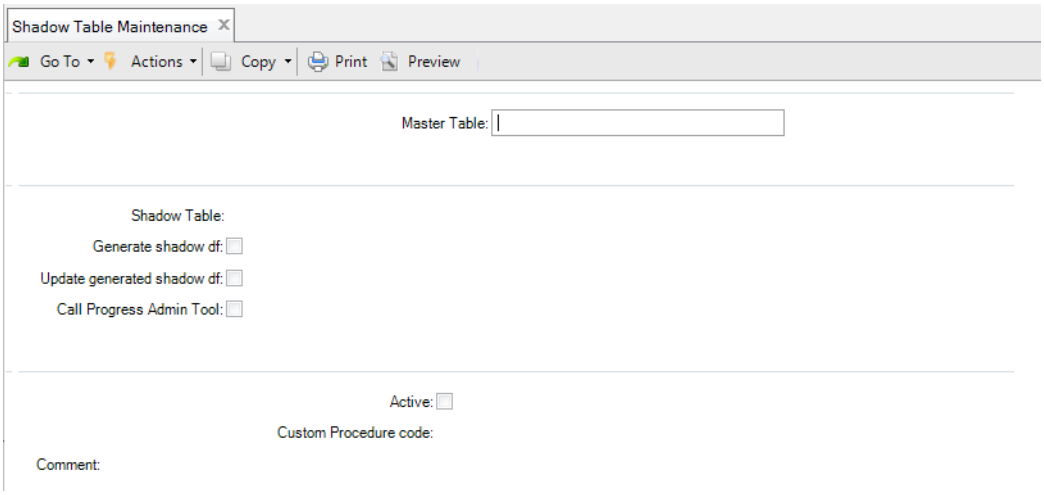


Shadow Table Maintenance (90.1.4)

Shadow Table Maintenance allows QAD ICT programmers to register a new *shadow table* in the system. A shadow table expands the functionality of the existing table. The registered shadow table is:

- Automatically defined. A `.df` file is automatically generated. It includes a unique primary index that maps to the main table.
- Supported by delete. The shadow table record is automatically deleted when the associated main record is removed.
- Registered and available for the developer's own procedures. QAD ICT registers the ROWID of a found record and makes it available (on request; see "getRecord(identifier)" on page 9) for QAD ICT processing and for programmers.

  Note This happens only when a trigger link for the main table has been added.



Master Table. Name of the main (existing) table. Mandatory.

Shadow Table. Name of the shadow table. Mandatory.

Generate shadow df. Determines if QAD ICT generates a `.df` file of the shadow table. The generated file includes a unique primary index that maps to the main table. Optional.

Update generated shadow df. Specify whether you want to edit the shadow table database definition generated by QAD ICT. Optional.

Note This field is not updatable in .NET UI/Desktop.

Call Progress Admin Tool. Specify whether you want to start the Progress Admin Tool, which lets you to load the generated shadow database definition. Optional.

Note This field is not updatable in .NET UI/Desktop.

Active. Determines if the consistency link is active or not. Mandatory.

Custom Procedure code. For backward compatibility only. In QAD ICT 2.0 and higher, Subscribers Maintenance should be used.

Name of the procedure that should be executed before any record (the main one or the shadow one) is removed. The custom procedure code must be defined in any of the custom QAD ICT persistent programs, or it can be a normal program that is not defined as persistent. Optional.

Comment. Additional information, such as the reason for activating the record. Optional.

Using the Program

1. Run Shadow Table Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Select an already-supported master table or enter the name of a new table to be extended by the shadow table.

   Note The table must exist in the database dictionary of the connected database. Otherwise, the system displays an error message.
3. Define the shadow table name.
4. Set Generate shadow df to Yes.
5. Set Update generated shadow df to Yes.

   Note If you work in .NET UI/Desktop, this field is not updatable. If you want to edit the generated `.df` file, you must do it manually.
6. Set Call Progress Admin Tool to Yes.

   Note If you work in .NET UI/Desktop, this field is not updatable. You must load the `.df` file manually.

   Note Before you press Go and call the Progress Admin Tool, make sure that no users are connected to the QAD Enterprise Applications system. Changing the database schema with users connected to it may be impossible.

   If Generate shadow .df is Yes, the system automatically generates a proper `.df` file. If Update generated shadow df flag is Yes, the system opens the Progress Editor with the loaded `.df` file. You can modify the file and save your changes. Close the Progress Editor. If Call Progress Admin Tool is Yes, the system starts the Progress Admin Tool, from which you can load the generated `.df` file.
7. Make the consistency link active/inactive.
8. Add, update, or delete a comment.
9. Exit the Shadow Table Maintenance program.

10 Respond to the prompt to rebuild link and consistency triggers.

Note During the rebuild process, source code for new triggers is automatically generated, placed in a session working directory, and compiled into the working directory keeping the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`. To make your customization visible for other users, you should move all generated source and r-code to the central environment.

See “QAD ICT Tasks” on page 52 for details about how to manage your customizations.

11 Restart ICT to load your customization.

Shadow Table Browse (90.1.5)

Shadow Table Browse is a standard browse for shadow tables defined in Shadow Table Maintenance.

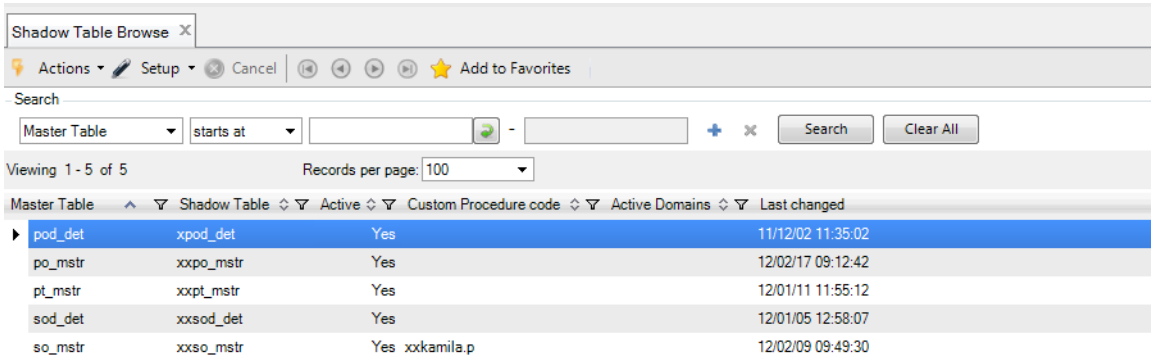


Table Triggers Maintenance (90.1.13)

Table Triggers Maintenance allows users to activate triggers on the database events WRITE, CREATE, DELETE for particular tables. Activated events can be handled with custom programs defined in subscribers. See “Subscribers Maintenance (90.1.22)” on page 25.

Moreover, each record from the table that has active proper events will be available on request with the *getRecord()* function after it is deleted, created, or written.

Note The ON FIND trigger can be activated using Trigger Link Maintenance.

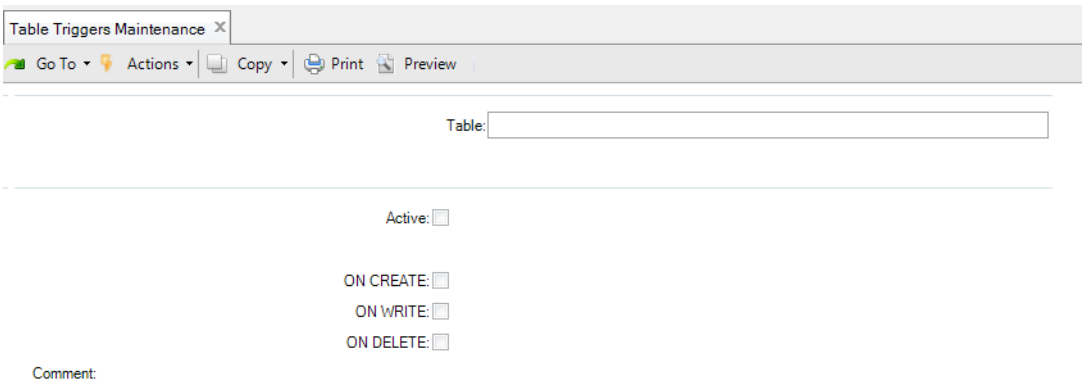


Table. Name of the table. Mandatory.

Active. Determines if events that were set for this table should be active or not. Mandatory.

ON CREATE. Specify whether event ON CREATE of table should be followed. Optional.

ON WRITE. Specify whether event ON WRITE of table should be followed. Optional.

ON DELETE. Specify whether event ON DELETE of table should be followed. Optional.

Comment. Additional information such as the purpose of the activation. Optional.

Using the Program

1. Run Table Triggers Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Select an already-supported table or enter the name of a new table.

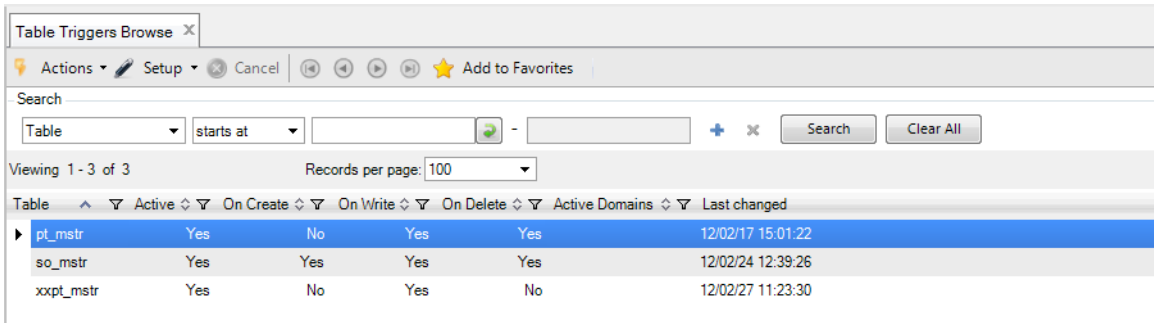
   Note The table must exist in the database dictionary of the connected database. Otherwise, the system displays an error message.
3. Make the events for the chosen table active/inactive.
4. Decide which events should be followed for the chosen table. Enter Yes next to events you want to activate.
5. Add, edit, or delete the comment.
6. Respond to the prompt to rebuild link and consistency triggers.

   Note During the rebuild process, source code for new triggers is automatically generated, placed in a session working directory, and compiled into the working directory keeping the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`. To make your customization visible for other users, you should move all generated source and r-code to the central environment.

   See “QAD ICT Tasks” on page 52 for details about how to manage your customizations.
7. Restart ICT to load your customization.

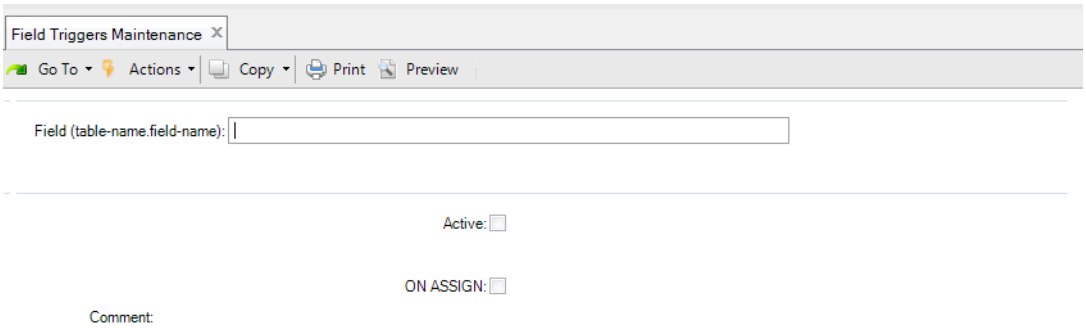
Table Triggers Browse (90.1.14)

Table Triggers Browse is a standard browse for triggers on tables defined in Table Triggers Maintenance.



Field Triggers Maintenance (90.1.16)

Field Triggers Maintenance enables users to follow an ON ASSIGN event on the specified field of the table. Activated events can be handled with custom programs defined in subscribers. See “Subscribers Maintenance (90.1.22)” on page 25.



Field. Table field, defined as table_name.field_name. Mandatory.

Note The dot symbol (.) is required between the table and field names.

Active. Specify whether events set for this field are active or not. Mandatory.

ON ASSIGN. Specify whether the ON ASSIGN event for this field should be followed. Optional.

Comment. Enter additional information such as the reason for activating the trigger. Optional.

Using the Program

1. Run Field Triggers Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Select an already-supported table and field or enter the name of a new table and field.

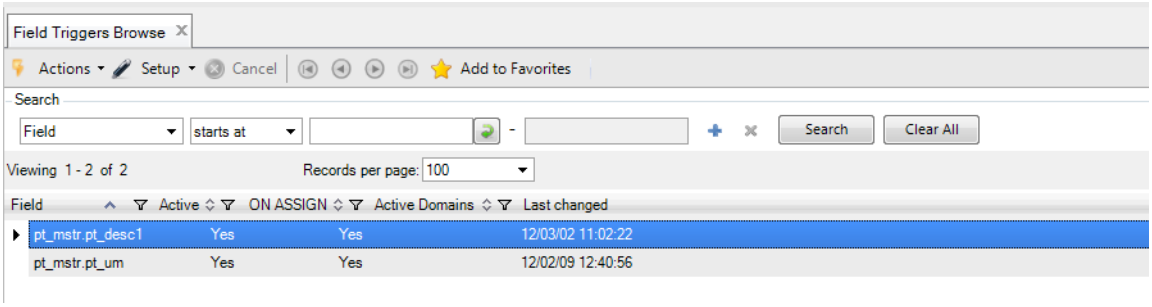
   Note The field must exist in the database dictionary of the connected database. Otherwise, the system displays an error message.
3. Make the events for the chosen table active/inactive.
4. Decide if you want to activate an ON ASSIGN event for the chosen field.
5. Add, edit, or delete the comment.
6. Respond to the prompt to rebuild link and consistency triggers.

   Note During the rebuild process, source code for new triggers is automatically generated, placed in a session working directory, and compiled into the working directory keeping the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`. To make your customization visible for other users, you should move all generated source and r-code to the central environment.

   See “QAD ICT Tasks” on page 52 for details about how to manage your customizations.
7. Restart ICT to load your customization.

Field Triggers Browse (90.1.17)

Field Triggers Browse is a standard browse for triggers on fields defined in Field Triggers Maintenance.



Simplified Audit (90.1.19)

The Simplified Audit function allows the user to perform an audit on selected tables or particular fields. Each change done by users is registered so that the administrator can keep the track of:

- Old and new values
- Name of the user who made the change
- Date and time when the change was made

Changes are stored in the standard QAD audit table. Use Master Table Audit Trail Report (36.17.2) to list the changed data.

Note In some QAD Enterprise Applications versions, Master Table Audit Detail Report (36.17.2) is not available. In this case, the user must prepare a browse for the aud_det table.

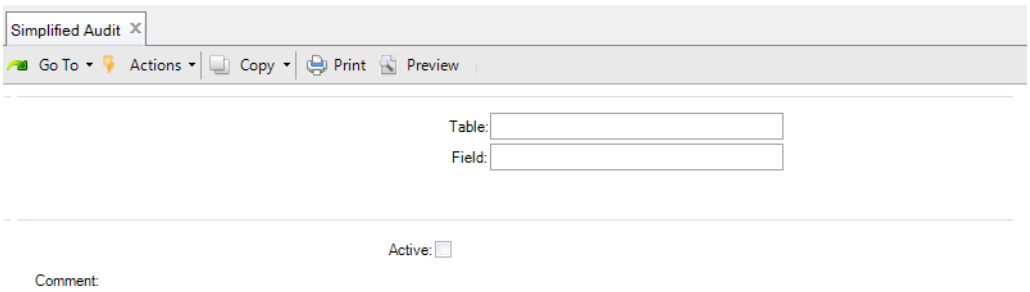


Table. Name of the table. Mandatory.

Field. Name of the field. Mandatory.

Active. Specify whether audit is performed for the defined field. Mandatory.

Comment. Additional information, such as the reason for activating the trigger. Optional.

Using the Program

1. Run Simplified Audit using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Select an already-supported table and field or enter the name of a new table and field.

Note The field must exist in the database dictionary of the connected database. Otherwise, the system displays an error message.

Note Remember that an ON WRITE trigger for the table should be defined and activated to enable tracking the changes. To track deletion, an ON DELETE trigger should be active. See “Table Triggers Maintenance (90.1.13)” on page 21 for details.

3 Make the audit for the chosen field active/inactive.

4 Add, edit, or delete the comment.

5 Restart ICT to load your customization.

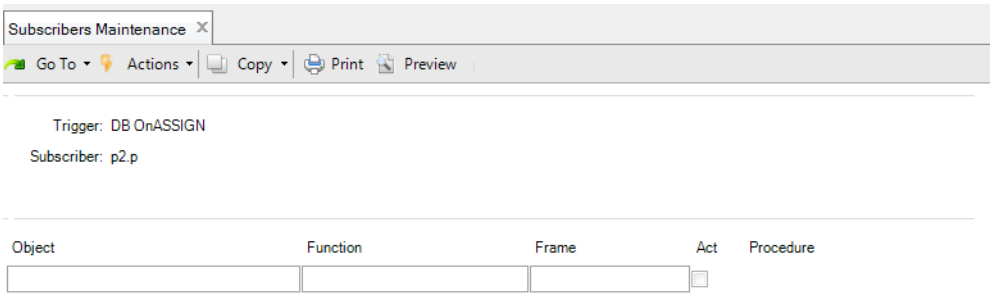
Subscribers Maintenance (90.1.22)

In the second release of QAD ICT, the publishing/subscribing mechanism was added. After defining a DB trigger, the developer could subscribe that trigger and execute external procedures inside the subscriber procedure. That was possible because for each definition of the trigger, a publishing line was added. The process of publishing and subscribing a trigger was not entirely automatic. The developer had to create the subscriber procedure, develop the Progress code inside, and compile it; only the publishing mechanism was automated.

Subscribers Maintenance allows QAD ICT programmers to do the same actions, but the process is completely automated. The program is designed to work in character mode. Although the program will not raise an error when executed in a .NET UI environment, some of the functionality does not work.

In the function, each developer can select a QAD Enterprise Applications database trigger and define the subscriber procedure in which that trigger should be subscribed. The system automatically creates that file, adds the subscriber code needed, and compiles the procedure. Based on input from the user, the system also creates the external procedure (which can be customized by the developer) and compiles it.

Note Beginning with ICT version 4.1.20, you can disable procedure usage in “Domains by tasks,” “Tasks by domains,” and “Activate/Deactivate Task”—the current setup is validated at runtime. Note that the subscriber defined as a super procedure is always started, regardless of the domain setup. The subscriber then decides whether the custom code should be executed (based on the current setup).



Trigger. The type of the trigger (DB OnASSIGN, DB OnCREATE, DB OnDELETE, DB OnFIND, DB OnWRITE). Mandatory.

Subscriber. Name of the subscriber procedure. Mandatory.

Object. Name of a field (in the form table_name.field_name) for ON ASSIGN trigger, or name of the table for other types of triggers. Mandatory.

Function. Name of the function that causes execution of the trigger. Optional.

Frame. Name of the frame that causes execution of the trigger. Optional.

Act (Active). Indicates whether a custom program will be fired when the chosen type of trigger is fired on the specified table:

- In the specified frame in the specified function (when both function and frame are not empty) or
- In the specified function (when frame is empty) or
- Anywhere (when both function and frame are empty)

This field is mandatory.

Procedure. Name of the custom procedure that should be executed when the chosen trigger is fired under the specified condition. Mandatory.

Using the Program

1. Run Subscribers Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Select the type of trigger and choose an existing subscriber, or enter the name of a new one.
3. Define the object (field name for ON ASSIGN trigger or table name for other triggers) and/or function and/or frame.
4. Specify whether the custom procedure should be active.
5. Enter the name of a custom procedure.
6. If the procedure does not exist, you are prompted to create it from a template. Answer Yes to create a template of the procedure.
7. You are prompted to edit your procedure. Answer Yes to open Progress Editor with the loaded procedure. Customize the template and create your own procedure. Save it and close Progress Editor.
8. You are prompted to compile your procedure. Answer Yes to compile it automatically or No if you want to compile it manually.

   Note When you answer Yes to these questions, the system creates the files, prompts you to edit them, and compiles them to the task directory keeping the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.
9. You are informed that your subscriber definition was changed and prompted to regenerate it. Answer Yes to allow QAD ICT to regenerate the subscriber code.
10. You are prompted to edit subscriber code. Remember that subscriber code is generated automatically and each manual change will be erased by QAD ICT during the next regeneration.

11 Allow ICT to compile the subscriber procedure to the current task directory. R-code is stored in the standard r-code directory structure, `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.

12 Make sure that a subscriber was added to a pool of super programs. See “Super Program Maintenance (90.24.13)” on page 76.

13 Restart ICT to load your customization.

QAD ICT UI Handler

This section describes programs on the QAD ICT UI Handler menu (90.3).

Group Functionality

Most of programs in the UI Handler section use the group functionality. The group functionality is also enabled for program hooks; see “Program Hook Maintenance (90.5.13)” on page 46. The group field in the maintenance functions determines for which user groups the customization should be enabled.

Multiple groups should be separated by a comma. You can also define the groups that should *not* use the customization using “!” before the group name. For example, !group1,* means that only group1 will not be able to see the customization. An asterisk (*) indicates all groups, and cannot be used as a mask (for example, A* does NOT mean all groups beginning with A). By default (empty value), customization is available for all groups.

Note Groups with the “!” prefix should be listed before other groups.

Note Be aware that users can belong to many groups. For example, to access users who belong to group1 but not group2, you can use definition !group2,group1

Note Be aware that in case of two identical sets of parameters defined for different groups, when the user running customized code is a member of two defined groups, the procedure (or other kind of a customization) will be fired twice.

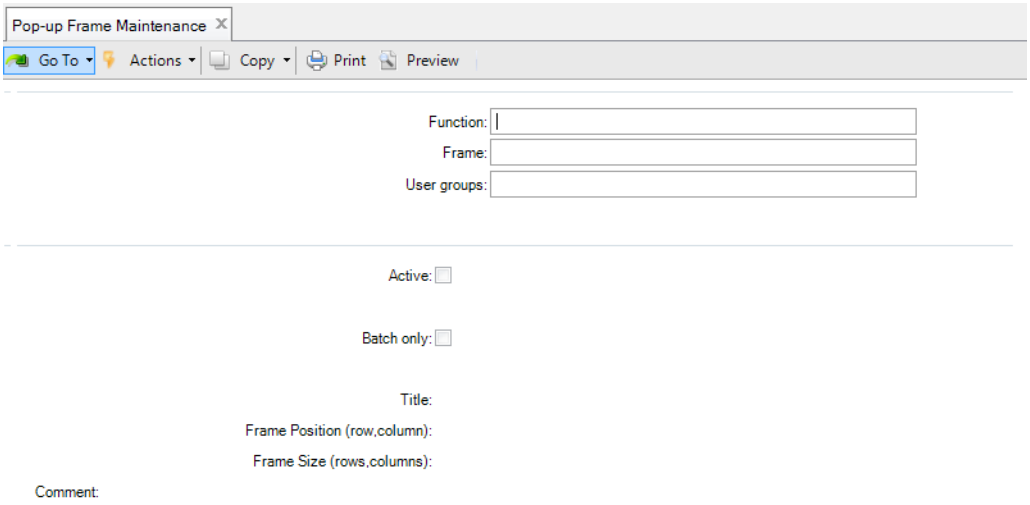
Beginning with QAD ICT 4.0, you can add more than one UI trigger of the same type for a particular triple: frame, field, and program. To use this functionality, add a UI trigger as usual and define the first group on the group lists as #name, where ‘#’ (hash) is a fixed mark and ‘name’ is a string of characters. Order of processing (when there is more than one UI trigger defined for a triple) is determined by the alphabetical order of string values after the hash mark. For example, the UI trigger for group #aa will be fired before the one for #bb and for #10 before #9 but after #09. An entry with an empty “group” field will be fired before other entries.

Pop-up Frame Maintenance (90.3.1)

Pop-up Frame Maintenance allows you to add a custom frame

- Into a standard QAD Enterprise Applications function
- Into a custom procedure generated by Program Generator (starting with QAD ICT 4.0)

A pop-up frame can be called by the *popup_frame* function in UI Trigger Maintenance or Program Hook Maintenance.



Function. Specifies the name of the maintenance function where the change should take place.

Frame. Name of the custom pop-up frame.

User groups. Name of the user groups for which the customization should be enabled. See “Group Functionality” on page 27. Optional.

Active. Specify whether the pop-up frame should be active or not. Mandatory.

Batch only. Specify whether the frame should be enabled in batch mode only. Optional.

Title. Title of the frame. If the value is defined as a translation term, the title will be translated to a user-defined language. Mandatory.

Frame position (row,column). Coordinates of the upper left corner of the frame. Mandatory.

Frame size (row,column). Size of the frame – number of rows (height) and number of columns (width). Mandatory.

Note A frame with at least one field should be higher than 2 rows.

Comment. Additional information, such as the reason for activation. Optional.

Using the Program

1. Run Pop-up Frame Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Specify the name of the maintenance function where the change should take place.
3. Specify the name of the custom frame.
4. Specify the user groups for which the frame should be active.
5. Make the setting active/inactive.
6. Specify if the frame should be active only in Batch mode (used for CIM loading purposes).

7 Set the title for the custom frame.

8 Set the position for the custom frame by specifying the row and the column of the left upper corner of the frame.

9 Set the size for the custom frame by specifying the rows (height) and the columns (width).

10 Enter an optional comment.

11 Use copying functionality if needed.

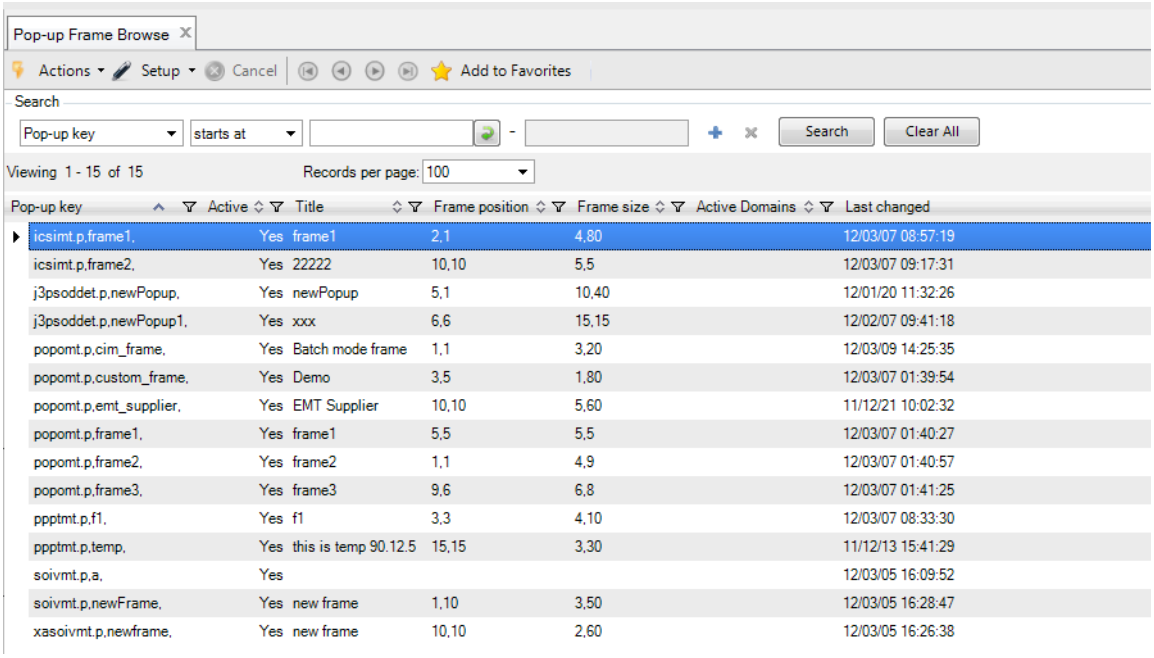
12 Exit Pop-up Frames Maintenance.

13 Restart ICT processing.

Use Frame & Fields Properties Maintenance to add fields to the new frame, and execute it using the *popup_frame* function.

Pop-up Frame Browse (90.3.2)

Pop-up Frame Browse is a standard browse for pop-up frames defined in Pop-up Frame Maintenance.



Pop-up key	Active	Title	Frame position	Frame size	Active Domains	Last changed
icsimt.p,frame1,	Yes	frame1	2,1	4,80		12/03/07 08:57:19
icsimt.p,frame2,	Yes	22222	10,10	5,5		12/03/07 09:17:31
j3psoddet.p,newPopup,	Yes	newPopup	5,1	10,40		12/01/20 11:32:26
j3psoddet.p,newPopup1,	Yes	xxx	6,6	15,15		12/02/07 09:41:18
popomt.p,cim_frame,	Yes	Batch mode frame	1,1	3,20		12/03/09 14:25:35
popomt.p,custom_frame,	Yes	Demo	3,5	1,80		12/03/07 01:39:54
popomt.p,emt_supplier,	Yes	EMT Supplier	10,10	5,60		11/12/21 10:02:32
popomt.p,frame1,	Yes	frame1	5,5	5,5		12/03/07 01:40:27
popomt.p,frame2,	Yes	frame2	1,1	4,9		12/03/07 01:40:57
popomt.p,frame3,	Yes	frame3	9,6	6,8		12/03/07 01:41:25
ppptmt.p,f1,	Yes	f1	3,3	4,10		12/03/07 08:33:30
ppptmt.p,temp,	Yes	this is temp 90.12.5	15,15	3,30		11/12/13 15:41:29
soivmt.p,a,	Yes					12/03/05 16:09:52
soivmt.p,newFrame,	Yes	new frame	1,10	3,50		12/03/05 16:28:47
xasoivmt.p,newframe,	Yes	new frame	10,10	2,60		12/03/05 16:26:38

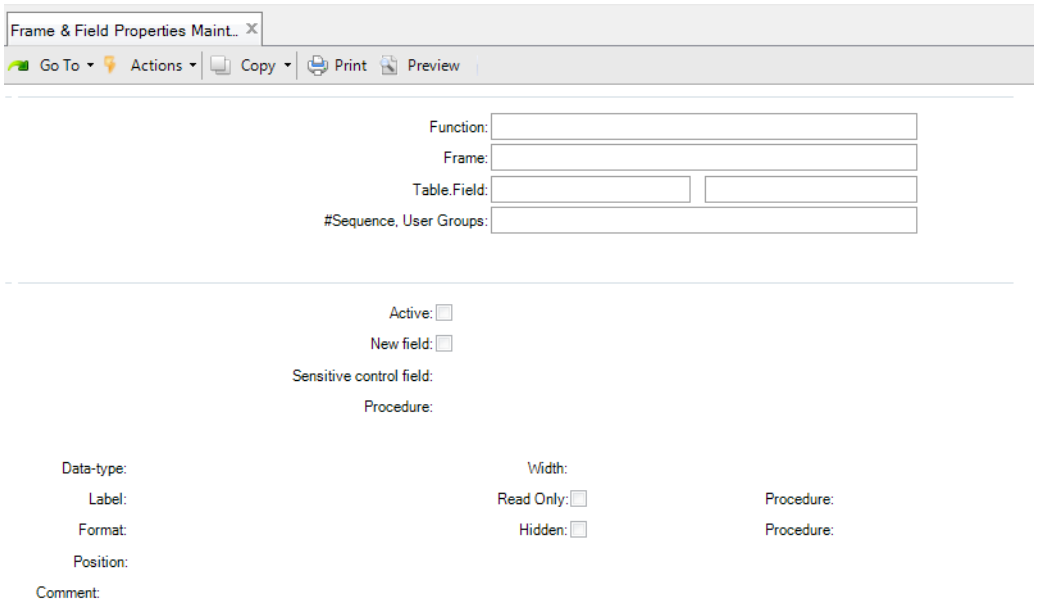
Frame & Field Properties Maintenance (90.3.4)

Frame & Field Properties Maintenance allows QAD ICT programmers to change the frames and fields parameters for specified user groups:

- Make them read-only (also conditionally)
- Hide them (also conditionally)
- Change the format
- Change the position in the frame

- Change the width
- Also provides a way of adding new fields to standard and custom frames, without changing the source code (also conditionally)

Note It is recommended that you use this program in terminal mode if you want to use conditional add/read-only/hidden functions.



Function. Name of the program that should be customized. Mandatory.

Frame. Name of the frame in which a new field should be added. (This can be a custom frame as well.) Mandatory.

Table.Field. Name of the table and field to be added (one text field for the table name and a second for the field name). New fields added as Table.Field will be automatically updated and stored in the database only if an ON FIND trigger for the proper table is defined and active in Trigger Link Maintenance. Mandatory.

Note For existing fields, do not enter the table name, only the field name.

Note You can enter only the field name while adding a new field to the frame. It this case, the new field is treated as a variable. To update the value on the screen or save the value in the database, the appropriate custom procedures should be created.

User Groups. Name of the user groups that can use this customization. See “Group Functionality” on page 27. Optional.

Active. Specify whether customization should be active. Mandatory.

New field. Specify whether the field should be added to the frame. (This is similar to Action 3 (Add) in Frame & Field Properties Maintenance in QAD ICT version 3.xx.) Mandatory.

Sensitive control field. Specify when/whether the field should be enabled/visible. It also defines the tab order. Valid entries are specified in the following table . Mandatory if New Field is Yes.

Entry	Description
Name of a field from the same frame	The dynamically created field will behave the same as the specified field from the same frame (that is, will be sensitive as long as the specified field is sensitive).
‘sensitive’	The dynamically created field will be sensitive as long as its frame is sensitive.
‘disabled’	The dynamically created field will be read only.

By default, dynamically added fields are processed in alphabetical order. To define your own order of processing (tab order), you can use Sensitive control field.

Procedure. The procedure name that returns the logical value: yes if the field should be added to the screen; no otherwise. To ensure correct tab order, enter a field name in “Sensitive control field” when the procedure is defined. “Sensitive control field” is only used to ensure own order of processing. Optional.

Data-type. Data type specified in Progress; for example, character. Mandatory if New Field is Yes.

Label. Label for the added field. If the value is defined as a translation term, the label will be translated to the user-defined language. Optional.

Format. Field format. Mandatory if New Field is Yes.

Position. Position in the frame defined as row, column. Mandatory if New Field is Yes.

Width. Width of the field. If width is less than the number of characters defined by the format, the field will be scrollable to enable users to enter more data than defined in the width value (but no more than defined by the Format value). Optional.

Read only. User can make an existing field read only (disabled). Optional.

Procedure. A procedure name that returns a logical value: yes if the field should be read-only; no otherwise. If the procedure is defined, the value in the Read Only field is ignored. Optional.

Hidden. User can make an existing field hidden (invisible). Optional.

Procedure. A procedure name that returns a logical value: yes if the field should be hidden; no otherwise. If the procedure is defined, the value in the Hidden field is ignored. Optional.

Comment. Additional information. Optional.

Using the Program

1. Run Frame & Field Properties Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the procedure and the frame names.

3 To add a new field that will automatically store data in the database and be populated with the current value from the database, enter the table name and field name. To add a new field that will be treated as a local variable or to modify the settings of the existing field, enter only the field name.

4 Decide if the modification should be active.

5 If you want to add a new field, set New Field to Yes. Otherwise, set it to No.

6 If you want to add a new field, set the Sensitive control field for it. Additionally, you can define the procedure, which dynamically determines if the field should be sensitive. If you are modifying an existing field, Sensitive control field should be disabled.

7 For existing fields, you can change width, format, position in the frame, and set the field as read only or hidden (and also define the procedure that makes the file read-only/hidden based on a specified condition). For a new field, you can also edit the data type and label. You cannot hide a new field.

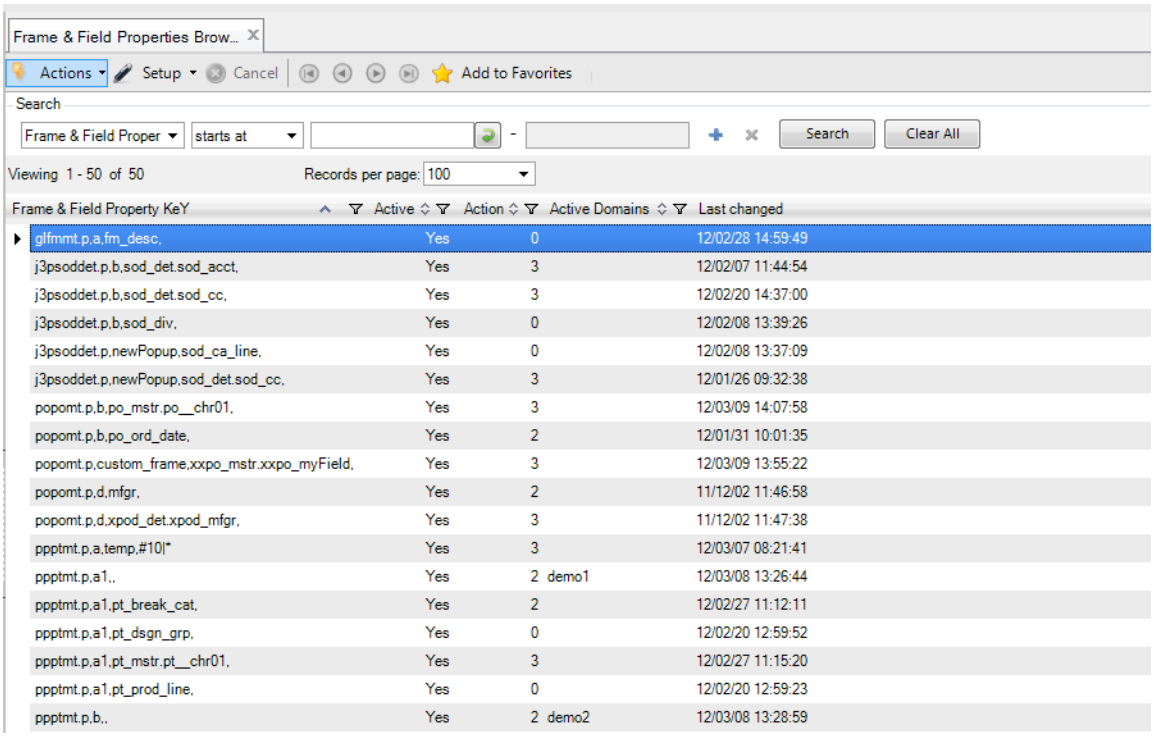
8 Add, change, or delete the comment.

9 If you entered any procedure name, ICT prompts you to create/edit/compile the procedure.

10 Restart ICT processing.

Frame & Field Properties Browse (90.3.5)

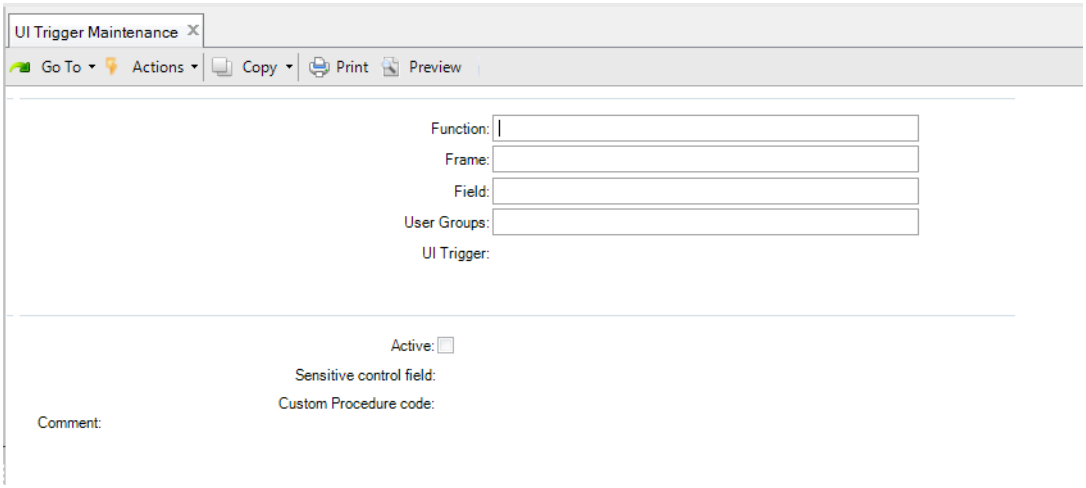
Frame & Field Properties Browse is a standard browse for settings defined in Frame & Field Properties Maintenance.



UI Trigger Maintenance (90.3.7)

UI Trigger Maintenance allows QAD ICT programmers to run any 4GL program—external or internally residing in super procedures—in response to a user interface event. QAD ICT supports the following user interface triggers:

Event	Object	Description
Entry	Field	When the system cursor is placed on a selected field
Entry	Frame	When the cursor is placed on a given frame area
Leave	Field	When the system cursor leaves a selected field because the user presses the Tab, Enter, or Go key
Go	Field	When the system cursor leaves a selected field because the user presses the Go key
Go	Frame	When the cursor leaves a given frame area



Function. Name of the program that should be customized. Mandatory.

Frame. Name of the frame. Mandatory.

Field. Name of the field. Optional.

User Groups. Name of user groups for which this customization is enabled. See “Group Functionality” on page 27. Optional.

UI Trigger. Type of the UI trigger (LEAVE, GO, ENTRY). Mandatory.

Active. Determines whether the UI trigger should be active. Mandatory.

Sensitive control field. Extra condition – the UI trigger will be fired only if the field pointed as Sensitive control field is sensitive. This helps to avoid multiple executions of the trigger; for example, when that specific frame name is linked with more than one frame in the maintenance program. Optional.

Custom Procedure code. Name of the procedure that should be executed on the defined event. You can also call a *popup_frame* directly by “popup_frame:name” where “name” is a name of the *popup_frame* defined in Pop-up Frame Maintenance. Mandatory.

Comment. Additional information, such as the reason for activation. Optional.

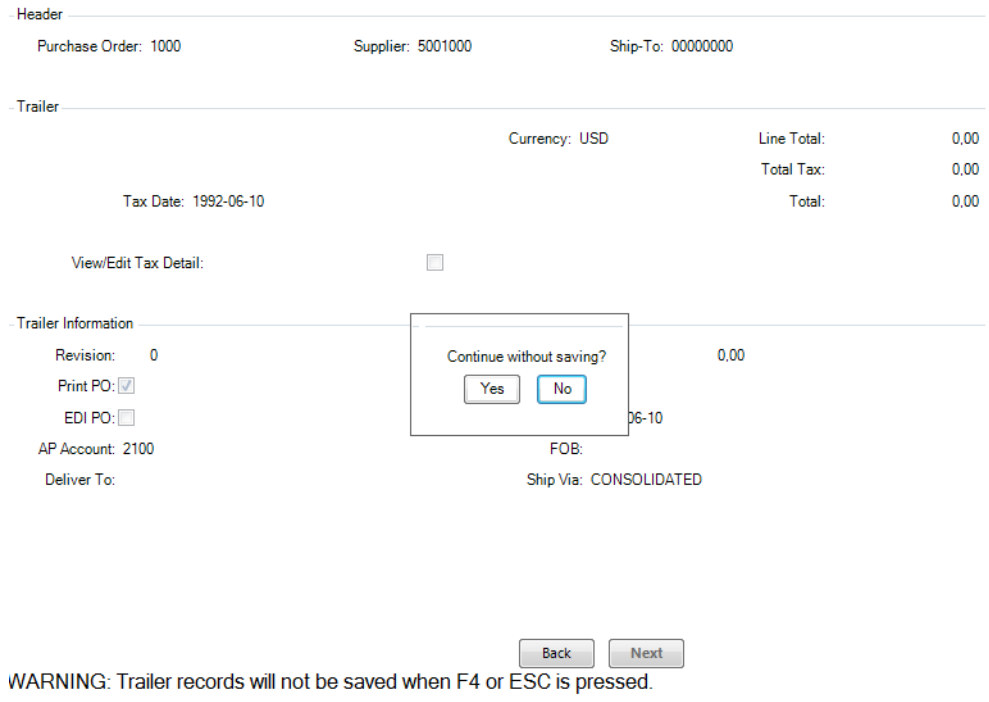
Using the Program

1. Run UI Trigger Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the name of the function that should be customized.
3. Enter the name of the frame and, optionally, the field.
4. Choose the event (GO, ENTRY, LEAVE). Note UI Triggers restrictions (see “UI Triggers and Functions Restrictions” on page 82).
5. Decide if customization should be active/inactive. Define the Sensitive control field if necessary.
6. Enter the name of the procedure that should be executed or use the fixed-phase “popup_frame” to run the defined pop-up frame directly.
7. If the specified procedure does not exist, allow ICT to create it from a template. Edit the procedure and add proper logic. Then allow ICT to compile the created procedure.

   Note To see your customization working, all your generated source and r-code should be moved to the central environment by closing the ICT task.
8. Restart ICT to load your customization.

Defining a UI Trigger on a Message Field

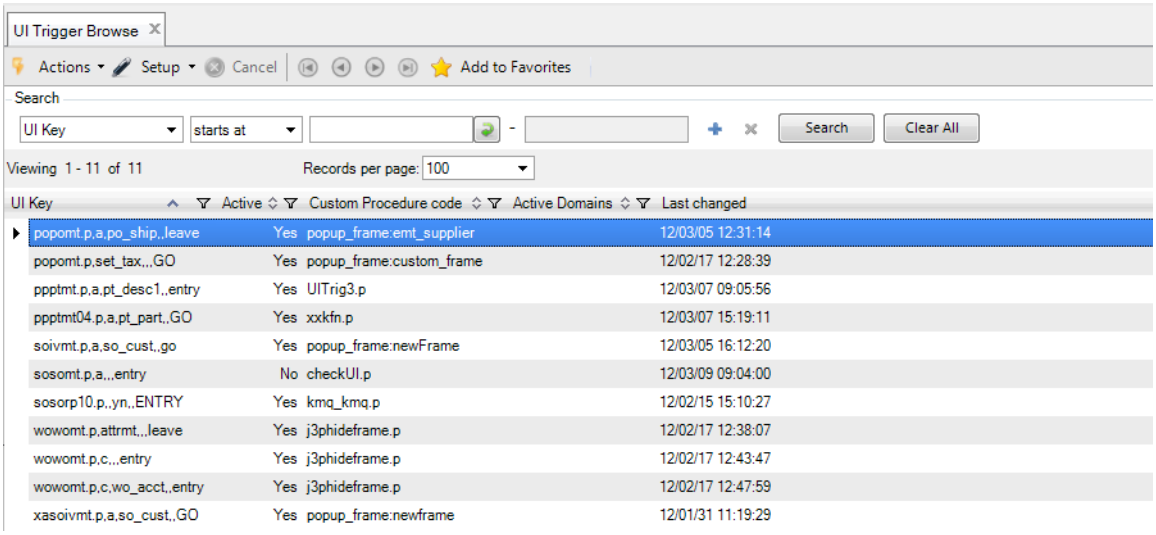
An example of a message field is shown below:



To set up a UI Trigger on that field, the user has to leave the frame field blank (the frame is not specified for those fields). This is the only case when frame can be blank; for other cases, the frame should be defined.

UI Trigger Browse (90.3.8)

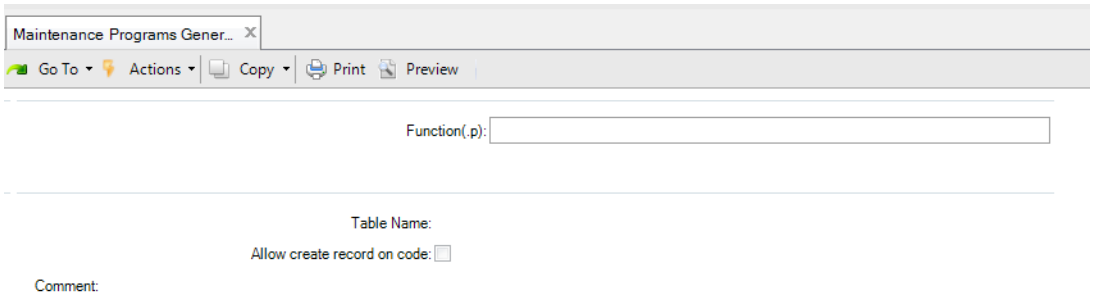
UI Trigger Browse is a standard browse for UI triggers defined in UI Trigger Maintenance.



Maintenance Programs Generator (90.3.10)

Maintenance Programs Generator allows users to create their own maintenance programs for every table, without having to write the Progress procedure on their own. Users can modify the procedure manually or add the custom fields using Frame & Fields Properties Maintenance.

The automatically generated maintenance program consists of two frames: frame a with index fields and frame b, which is originally empty and can be modified by the user.



Function(.p). Name of the function (with extension .p) that should be created. Mandatory.

Table Name. Name of the table for which the maintenance program should be created. Mandatory.

Allow create record on code. Determines if maintenance program should allow the user to add new records to the selected table. Optional.

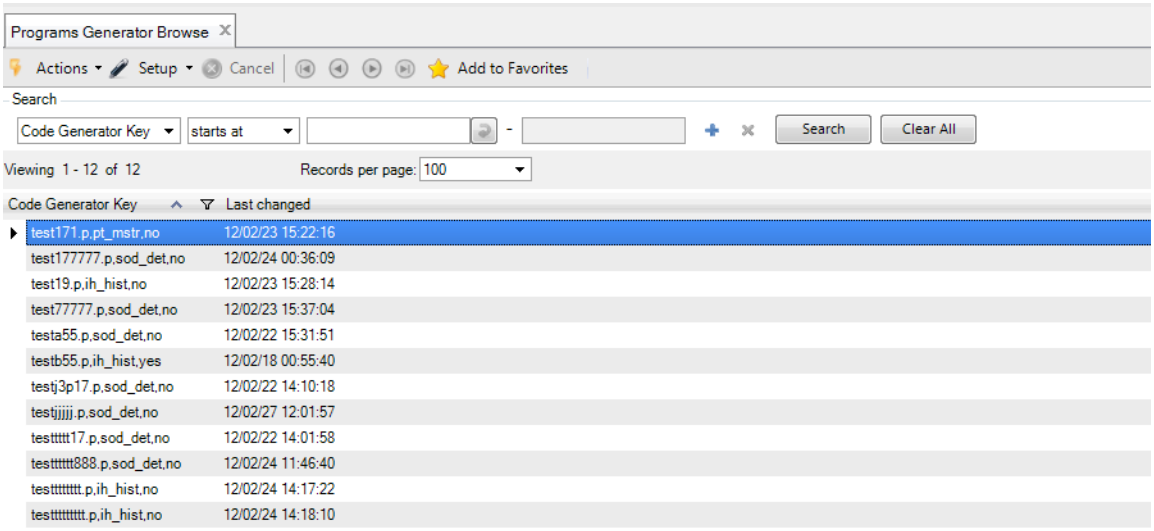
Comment. Additional information. Optional.

Using the Program

1. Run Maintenance Programs Generator using the standard QAD Enterprise Applications system menu.
2. Enter the name of a custom procedure that should be created.
3. Enter the name of the table that should be supported in the maintenance program.
4. Decide if the addition of a new record should be allowed in the maintenance program.
5. Add a comment (optional).
6. Allow ICT to create the maintenance program form template.
7. Edit the new program to make additional changes (or, if you do not want to change the source code you can add some logic, for example, using Frame & Fields Properties Maintenance).
8. Allow ICT to compile the code. ICT will compile the code to the task directory keeping the standard r-codes directory structure `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.
9. To enable a field addition in Frame & Field Properties Maintenance, add the program to the menu using Menu System Maintenance (36.4.4) and Program Information Maintenance (36.3.21.1) or the equivalent program, depending on your QAD Enterprise Applications version.
10. To enable CIM processing in a generated maintenance program, add a pop-up frame in Popup Frame Maintenance (90.3.1). The function should be defined as the generated maintenance program name, and the frame must be named “b.” Set the Batch only flag to Yes.

Programs Generator Browse (90.3.11)

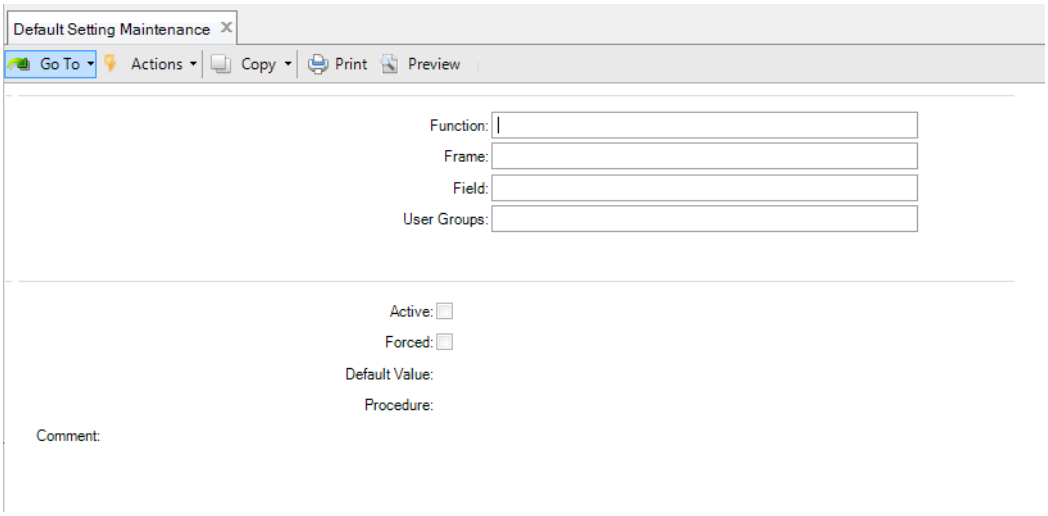
Programs Generator Browse is a standard browse for programs generated by Maintenance Programs Generator.



Default Setting Maintenance (90.3.13)

Default Setting Maintenance allows QAD ICT programmers to define default values for selected fields. The defined value may:

- Be used when the selected field is blank
- Replace the existing value (parameter Forced must be set to Yes)



Function. Name of the program that should be customized. Mandatory.

Frame. Name of the frame. Mandatory.

Field. Name of the field. Mandatory.

User Groups. Name of the user groups for which this customization should be active. See “Group Functionality” on page 27. Optional.

Active. Determines whether the customization should be active. Mandatory.

Forced. Determines whether the default value has to be forced (in cases where the field already has a non-blank value). Optional.

Default Value. Defines the default value for the field. Mandatory.

Default value for the field can be:

- TODAY phrase, which is evaluated to the current date
- TERM LABEL, which will be translated if proper translation is provided in Label Master Maintenance
- GLOBAL VALUE, defined in Global Value Maintenance

Procedure. Name of the procedure that will return (as an output parameter) a default value for the field. The procedure can be a custom procedure or can be defined as a procedure in a super program. Mandatory.

Note Only one field—Default value or Procedure—should be filled in.

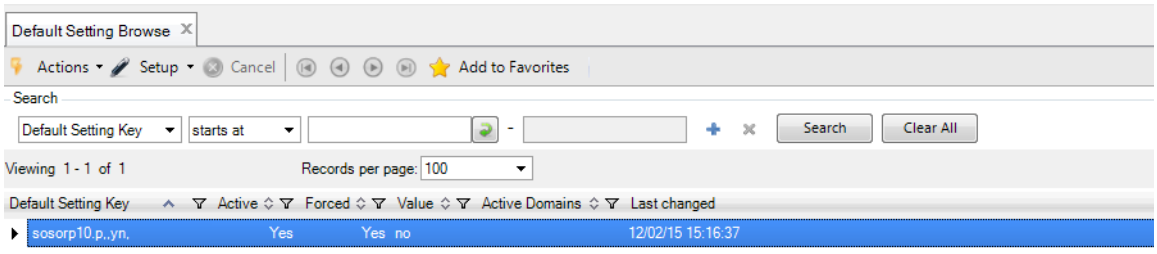
Comment. Additional information. Optional

Using the Program

1. Run Default Setting Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the name of the function that should be customized.
3. Enter the name of the field and its frame.
4. Define the user group for which this modification should be active or inactive.
5. Define if the default value should be forced for every field, even if its value is not blank.
6. Define the default value or enter the name of the procedure that will return the value.
7. Add a comment (optionally).
8. If the procedure option was chosen, allow ICT to create the procedure form template, edit and modify the procedure, and let ICT compile it to the task directory keeping the standard r-codes directory structure `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.
9. Restart the ICT processing.

Default Setting Browse (90.3.14)

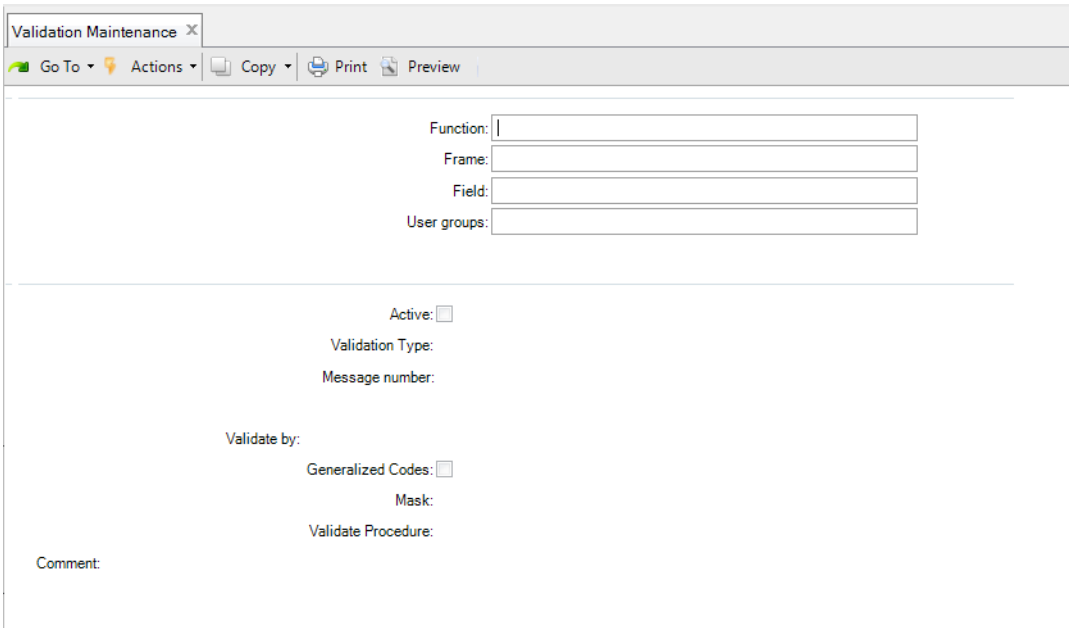
Default Setting Browse is a standard browse for settings defined in Default Setting Maintenance.



Validation Maintenance (90.3.16)

Validation Maintenance allows QAD ICT programmers to define validation on any standard, customized, or dynamically generated field within QAD Enterprise Applications. The displayed message information depends on the defined message number. It can be:

- A simple message
- A warning
- An error message



Function. Name of the program that should be customized. Mandatory.

Frame. Name of the frame. Mandatory.

Field. Name of the field. Mandatory.

User Groups. Name of the user groups for which this customization should be enabled. See “Group Functionality” on page 27. Optional.

Active. Determines whether the customization should be active. Mandatory.

Validation Type. Indicates the validation type; allowed values are: 1=Message, 2=Warning, 3= Error. Mandatory.

Message number. Number of the message defined in Message Maintenance. Mandatory.

Note If the message number is 0 (zero), no message will be displayed by QAD ICT. The user can manage the displayed messages as part of a procedure (only when validation against the procedure is used).

Validate by: You can choose only one type of validation (Generalized Codes, Mask, or Procedure), as described below. Mandatory.

Generalized Codes: Validation against generalized codes.

Mask:. Validation against mask (user can use: “*” and “.” signs).

Note Examples of valid masks and their meanings are listed below:

Mask	Meaning
A*	The word must start with “A”
.A*	The second letter of the word must be “A”
*A	The last letter of the word must be “A”
..	The word must contain exactly 2 characters
.*	The value of the field cannot be empty

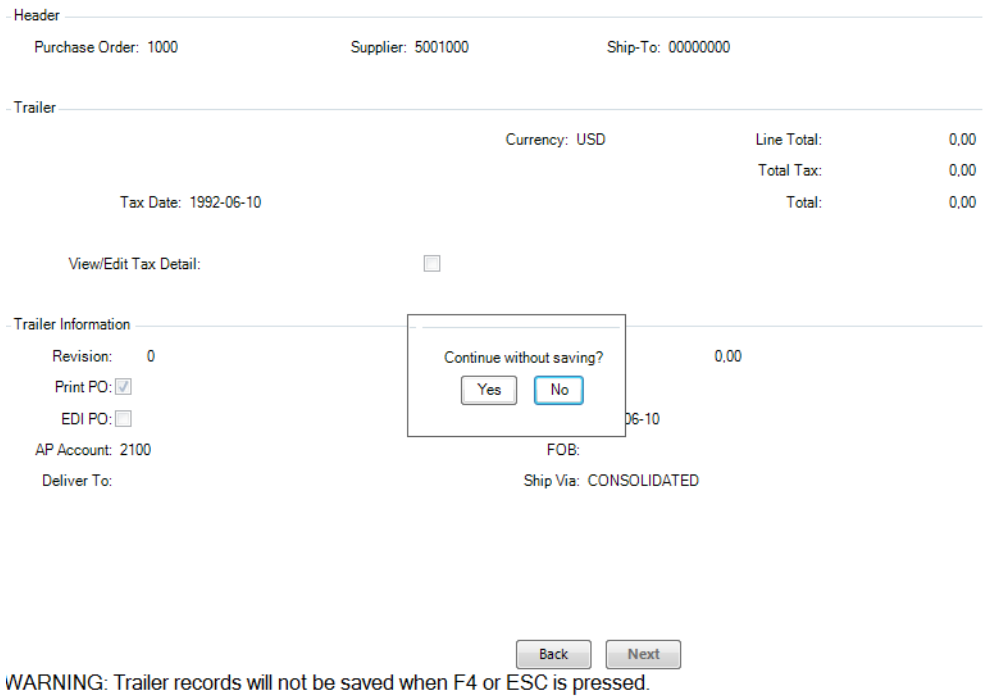
Procedure. Name of procedure that will return the logical value if the new value is valid. The procedure can be a custom procedure or can be defined as a procedure in a super program.

Comment. Additional information. Optional.

Using the Program

1. Run Validation Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the name of the function that should be customized.
3. Enter the name of the field and its frame.
4. Define the user group for which this modification should be active or inactive.
5. Define the type and number of the message that should be displayed in case of validation failure.
6. Choose the validation method: generalized codes, mask, or procedure.
7. If the procedure option was chosen, allow ICT to create the procedure form template, edit and modify the procedure, and let ICT compile it to the task directory keeping the standard r-codes directory structure `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.
8. Restart the ICT processing.

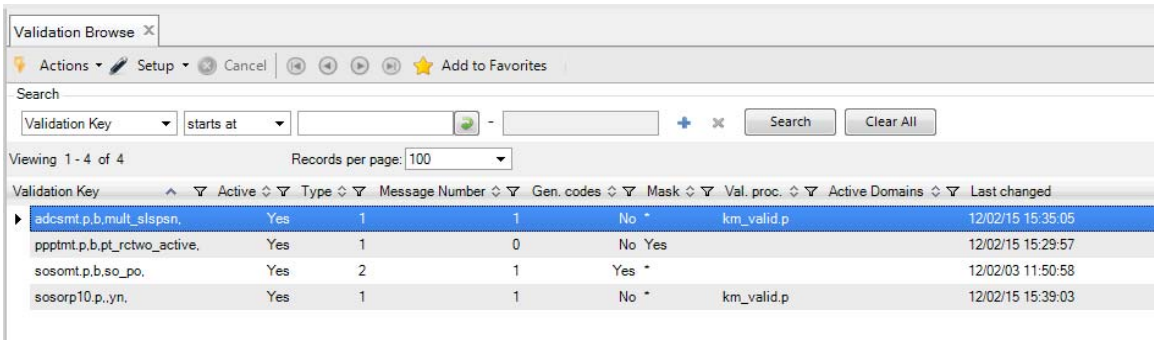
Note The validation can be set up on the fields that are updated by the user in the message area at the bottom of the screen. (In the .NET UI environment, fields are displayed in the pop-up window where the user has to click on the button.) An example is shown below:



To set up a validation on a field, the user has to leave the frame field blank (the frame is not specified for those fields).

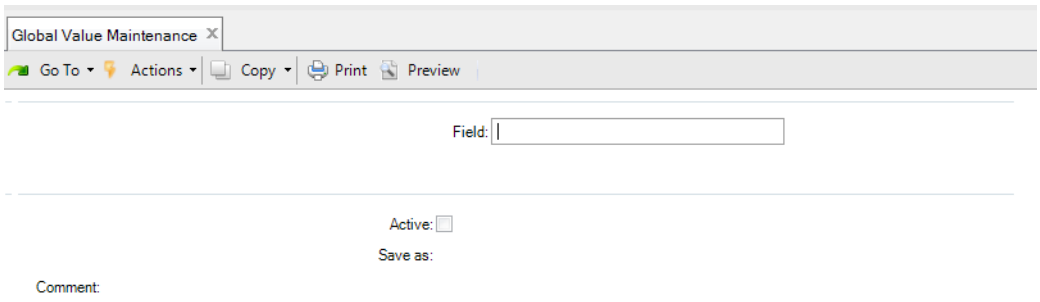
Validation Browse (90.3.17)

Validation Browse is a standard browse for settings defined in Validation Maintenance.



Global Value Maintenance (90.3.19)

Global Value Maintenance allows users to define the list of fields that are automatically registered as global values. The field value is registered as a global value when the user working with the QAD Applications function goes through the frame where the specified field is edited. The value registered as a global value can be used in Default Settings Maintenance (90.3.13) to specify the default value for the particular field.



Field. Specify the field from which the value should be stored as a global value. Mandatory.

Active. Determines whether the global value should be active. Mandatory.

Save as. Name of the global value. Mandatory.

Comment. Additional information. Optional.

Using the Program

1. Run Global Value Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the name of the field from which the value should be stored as a global value.
3. Define if the setting should be active.

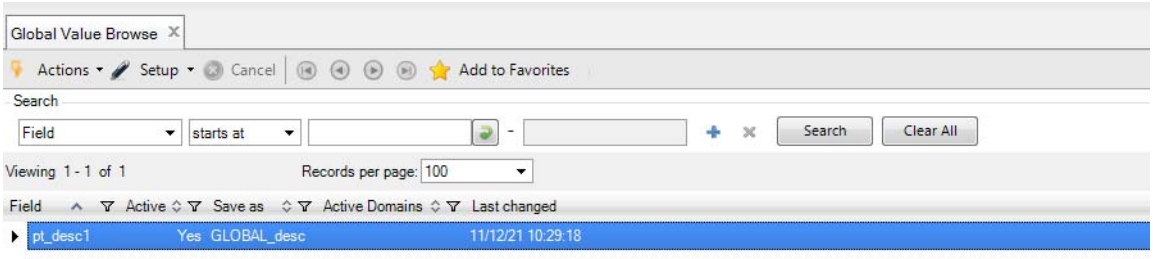
4 Use the Save as field to define the name of the global value where the specified field value should be stored. The name should be defined as “global_xxx” where “xxx” is variable text.

5 Enter a comment for the change (optionally).

6 Restart the ICT processing.

Global Value Browse (90.3.20)

Global Value Browse is a standard browse for settings defined in Global Value Maintenance.



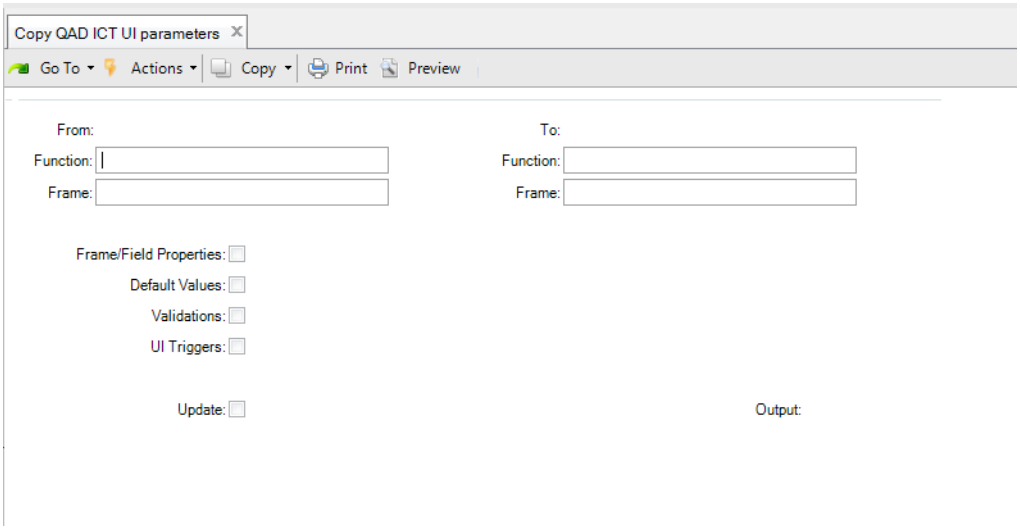
Copy QAD ICT UI Parameters (90.3.22)

Copy QAD ICT UI Parameters allows users to copy the QAD ICT UI settings from the specified functions/frames and assign them to different functions/frames.

Settings that can be copied are:

- Frame/Field Properties
- Default Values
- Validations
- UI Triggers

The copied entries are linked with the currently selected task.



From. Name of the source function and frame. Mandatory.

To. Name of the destination function and frame. Mandatory.

Frame/Field Properties. Determines whether Frame & Field properties settings should be copied. Optional.

Default Values. Determines whether Default Values settings should be copied. Optional.

Validations. Determines whether Validations settings should be copied. Optional.

UI Triggers. Determines whether UI Triggers settings should be copied. Optional.

Update. If Yes, the function will copy the settings. If No, the function will only print a report that can be used for listing purposes. Mandatory.

Output. Output device that should print the report. Mandatory.

Using the Program

1. Run Copy QAD ICT UI Parameters using the standard QAD Enterprise Applications system menu.
2. Define the function and frame in the From and To sections.

Note The names of the frames can be different in separate functions. If that occurs, the function will change the frame name to a new one.

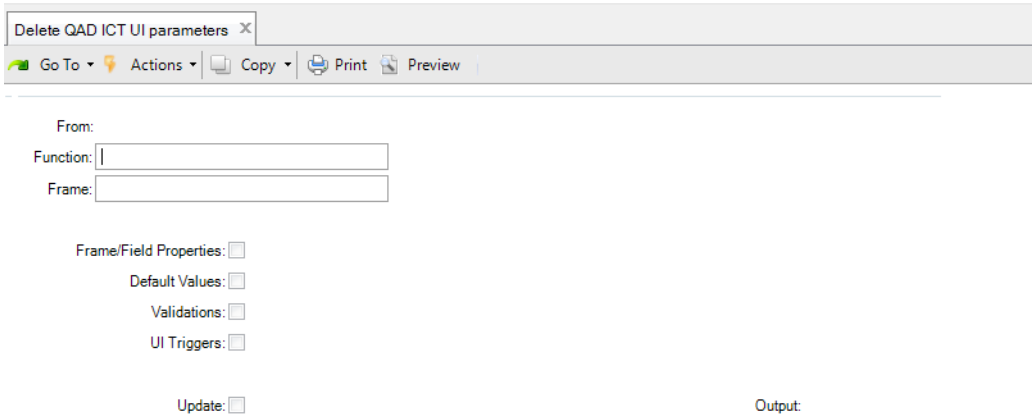
3. Specify which QAD ICT settings should be copied by setting the flags.
4. Set the Update flag. If Yes, the function will copy the settings. If No, the function will only print a report that can be used for listing purposes.
5. Choose the output device that should print the final report.
6. If Update was set to Yes, restart the ICT processing to see your copied customizations.

Delete QAD ICT UI Parameters (90.3.23)

Delete QAD ICT UI Parameters allows users to delete the QAD ICT UI settings from the specified functions/frames.

Settings that can be deleted are:

- Frame/Field Properties
- Default Values
- Validations
- UI Triggers



From. Name of the source function and frame. Mandatory.

Frame/Field Properties. Determines whether Frame & Field properties settings should be deleted. Optional.

Default Values. Determines whether Default Values settings should be deleted. Optional.

Validations. Determines whether Validations settings should be deleted. Optional.

UI Triggers. Determines whether UI Triggers settings should be deleted. Optional.

Update. If Yes the function will delete the settings. If No the function will only print a report that can be used for listing purposes. Mandatory.

Output. Output device that should print the report. Mandatory.

Using the Program

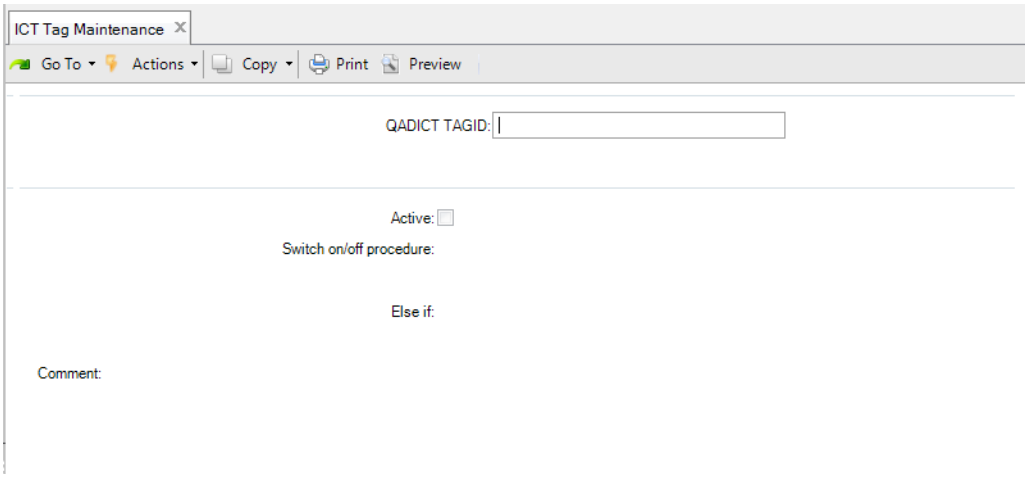
1. Run Delete QAD ICT UI Parameters using the standard QAD Enterprise Applications system menu.
2. Define the function and frame in the From section.
3. Specify which QAD ICT settings should be deleted by setting the flags.
4. Set the Update flag. If Yes, the function will delete the settings. If No, the function will only print a report that can be used for listing purposes.
5. Choose the output device that should print the final report.
6. If Update was set to Yes, restart the ICT processing to see your deleted customizations.

QAD ICT Intrusive Handler

This section describes the programs on the QAD ICT Intrusive Handler menu (90.5).

ICT Tag Maintenance (90.5.1)

Intrusive QAD ICT Tag Maintenance allows QAD ICT programmers to determine which lines from the source code will be executed. An additional use of the program is to specify which procedure should be executed instead of the code that was commented out using ICT tags. The programmer also can specify the procedure that can check whether to comment out the source code and run another procedure in its place.



QADICT TAGID. Unique ID of a tag. Mandatory.

Active. Determines if the defined tag should be active and Switch on/off procedure should be evaluated. Mandatory.

Switch on/off procedure. Its output parameter determines if the block of code marked with TAGID should be commented out (output set to Yes). Mandatory.

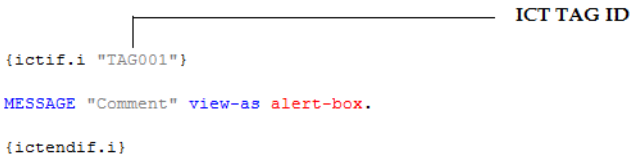
Else if. Procedure that should be executed instead of the block of code that was commented out. Optional.

Comment. Additional information such as the purpose of the tag. Optional.

Using the Program

1. Run ICT Tag Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter a unique ICT Tag ID or choose an existing one.
3. Choose if the tag should be active.
4. Define the Switch on/off procedure.
5. Define the alternative procedure (Else if) – optional.
6. Add, modify, or delete the comment.

Note You need to create Switch on/off and Else if procedures yourself. Remember to add a tag into the standard code. See the following example.



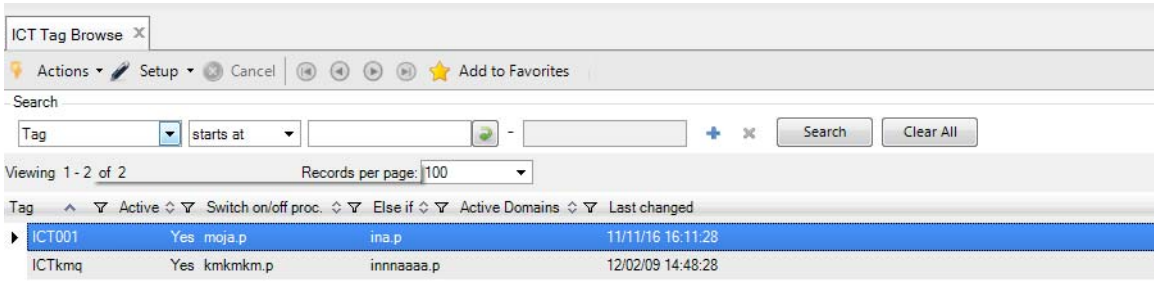
The tags in the example mark the code that should be commented out when tag TAG001 is active and the Switch on/off procedure returns “true.”

You can also use a tag to execute the external procedure using {ictrun.i “TAG ID”}. The external procedure will be executed (the one in elseif section) only if the tag is active and Switch on/off procedure returned “true.”

The block of code between includes will be commented out when ICT tag TAG001 is active and its Switch on/off procedure returns a positive output value.

ICT Tag Browse (90.5.2)

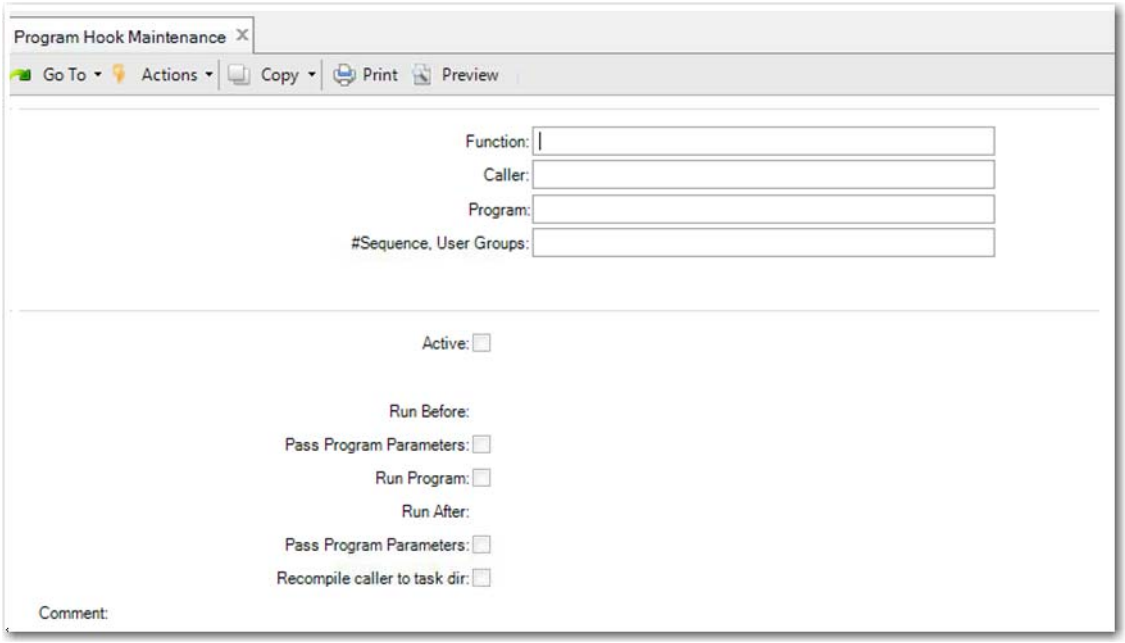
ICT Tag Browse is a standard browse for ICT Tags defined in ICT Tag Maintenance.



Program Hook Maintenance (90.5.13)

Program Hook Maintenance allows QAD ICT programmers to:

- Switch on/off the execution of the program. The program specified will be excluded from the execution list.
- Execute the custom procedure before or after the specified procedure.



Function. Name of a menu function where the change should take place. Mandatory.

Caller. The name of the program that is on the execution list when running the function specified in the Function field. Mandatory.

Program. The procedure that is executed by the Caller. Mandatory.

#Sequence, User Groups. Name of user groups for which this customization should be enabled. See “Group Functionality” on page 27. Optional.

Active. Makes program hook active/inactive. Mandatory.

Run Before. Definition of the procedure that should be executed before the one defined in the Program field. You can call a *popup_frame* directly by “popup_frame:name” where “name” is a name of the *popup_frame* defined in Pop-up Frame Maintenance. Optional.

Pass Program Parameters. Determines whether the system passes standard input/output parameters (passed by caller to program) to your defined Run Before procedure.

Run Program. Determines whether the procedure specified in the Program field should be executed. Mandatory.

Run After. Definition of the procedure that should be executed after the one defined in the Program field. You can call a *popup_frame* directly by “popup_frame:name” where “name” is a name of the *popup_frame* defined in Pop-up Frame Maintenance. Optional.

Pass Program Parameters. Determines whether the system passes standard input/output parameters (passed by caller to program) to your defined Run After procedure.

Recompile caller to task dir. Indicates whether the caller should be compiled automatically by ICT. The procedure will be compiled to your current task directory, and r-code will be moved to the central environment after you close the task.

Comment. Additional information. Optional.

Using the Program

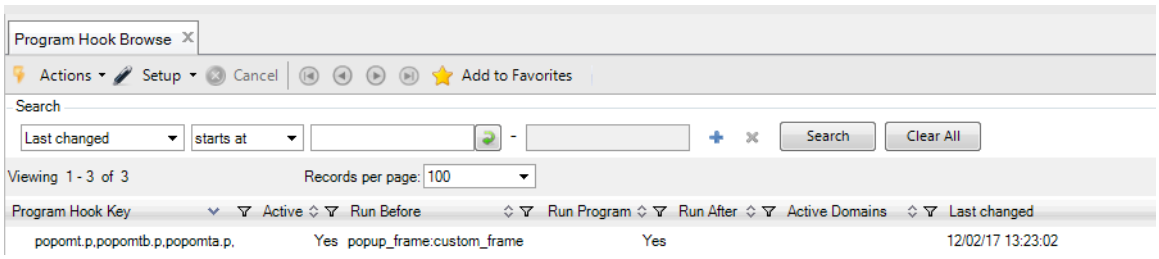
1. Run Program Hook Maintenance using the standard QAD Enterprise Applications system menu or ICT Development menu.
2. Enter the names of function, caller, and program.

   Note The Function field and Caller program can be the same. That means that the procedure specified in the Program field is the second on the execution list.
3. Set program hook active/inactive.
4. Use the Run Before field to define the procedure that should be executed before the procedure specified in the Program field. Decide if you want to pass standard parameters to your custom procedure.
5. Use the Run Program field to determine whether the procedure specified in the Program field should be executed.
6. Use the Run After field to define the procedure that should be executed after the procedure specified in the Program field. Decide if you want to pass standard parameters to your custom procedure.
7. Select “Recompile caller to task dir” to allow ICT to recompile the caller automatically. The procedure will be compiled to your current task directory, and r-code will be moved to the central environment after you close the task. Otherwise, compile the caller procedure manually.

   Note The compiled file is not permanently linked with the task. If you move the task to another environment, remember to compile your caller procedure manually or move the compiled caller from the original to the destination environment.
8. Add, change, or delete the comment.
9. Exit Program Hook Maintenance.
10. Restart ICT to load your customization.

Program Hook Browse (90.5.14)

Program Hook Browse is a standard browse for program hooks defined in Program Hook Maintenance.

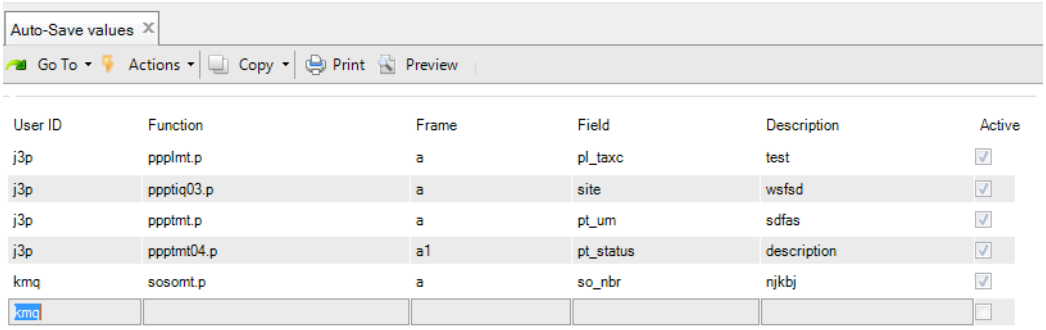


QAD ICT User Functions

This section describes a program that defines user-specific defaults.

Auto-Save Values (90.3.3)

Auto-Save Values enables the specified user to set the “default value” visible only to her or him and based on the last entered value of the selected field in the selected maintenance program. The latest entered value (indicated by function, frame, and field) is saved and used to fill this field automatically the next time the field is visible on the screen. Note that the value will be forced and can overwrite the existing one.



User ID. Identifier of the user who wants to have the default value set. Mandatory.

Function. Name of the program from the QAD menu. Mandatory.

Frame. Name of the frame. Mandatory.

Field. Name of the field in the chosen function and frame. Mandatory.

Description. Additional information. Optional.

Active. Determines whether the customization should be active. Mandatory.

Using the Program

1. Run Auto-Save Values using the standard QAD Enterprise Applications system menu.
2. Define the user for whom the customization should be defined. Only one user can be defined in each row.
3. Define the function, frame, and field.
4. Define whether the customization should be active.
5. Restart ICT processing.

QAD ICT Templates

QAD ICT Templates (90.22) allows developers to create a template for the programs shown in the following subsections.

Define the File Name for the procedure you want to create. The procedure will be saved automatically in the working directory and then edited in Progress Editor. You can customize the template. After you close the Progress Editor, ICT prompts you to compile the file. If the compilation is needed, ICT will compile the file to the working directory, keeping the standard r-codes directory structure `$LANGUAGE_DIR/$TWO_LETTERS_DIR`.

Events Subscriber (Super Program)

```
/*$FileName - QAD ICT Super Program                                    */
/* Generated by QAD ICT source codes generator                         */
/*V8:ConvertMode=Maintenance                                           */

{mfdeclre.i}
{icdef.i}

/*************************************************************************
* Subscribing OnAnyEvent                                                 *
*************************************************************************/
registerTrigger("SUBSCRIBE OnAnyEvent anywhere in $FileName").
subscribe to "OnAnyEvent" anywhere.
/*************************************************************************/
PROCEDURE OnAnyEvent.
    define input parameter ipcEventName as character no-undo.
    define input parameter ipcObjectName as character no-undo.
    define input parameter ipcParam as character no-undo.
    /* ipcEventName  - the event which has occured
                       UI Triggers: GO,ENTRY,LEAVE
                       DB Triggers: CREATE,WRITE,FIND,ASSIGN,DELETE

       ipcObjectName - the name of object which executed the trigger
                       For UI Triggers - widget name
                       For DB Triggers - DB Table or Field name

       ipcParam      - For UI Triggers - menu program + "," +
                                         frame name   + "," +
                                         widget name
                       For DB Triggers - RowID to the record converted to
                                         character value */

    {icdbg.i &T1='"OnAnyUIEvent"'
             &V1=ipcEventName
             &V2=ipcObjectName
             &V3=ipcParam
    }.

/*example*************************************************************
*    if ipcEventName  = "WRITE" and
*       ipcObjectName = "TR_HIST" then do:
*       {gprun.i ""program.p"" '(to-rowid(ipcParam)'}
*    end.
*********************************************************************/

END PROCEDURE. /*OnAnyEvent*/
```

Trigger Link Custom Procedure

```
/*$FileName  - Trigger  Link  Custom  Procedure                          */
/* Generated  by  QAD  ICT  source  codes  generator                     */
/*V8:ConvertMode=Maintenance                                            */

{mfdeclre.i}
{icdef.i}

define  input  parameter  iprRowid  as  rowid  no-undo.

/* iprRowid  - rowid  of  the  record  processed  by:
               - trigger  link       (on  find)
               - shadow  table  link  (on  delete)                       */
```

Default Value Calculation

```
/*$FileName  - Default  Value  Calculation  Program                      */
/* Generated  by  QAD  ICT  source  codes  generator                     */
/*V8:ConvertMode=Maintenance                                            */

{mfdeclre.i}
{icdef.i}

define  output  parameter  opcDefaultValue  as  character  no-undo.

/* opcDefaultValue  - output  parameter  which  will  be  passed  down  to
                      the  field  as  a  default  value                  */

/*example*************************************************************
*          retriving  default  value  for  pt_site  field
*              from  generalized  codes
*     ************************************************************
*
*  find  first  code_mstr  no-lock  where
*               code_fldname  = "DEFAULT_VALUES"  and
*               code_value    = "pt_site"
*               no-error.
*
*  opcDefaultValue  = if  available  code_mstr
*                     then  code_cmmt
*                     else  "".
*********************************************************************/
```

Validation Procedure

```
/*$FileName  - Validation  procedure                                    */
/* Generated  by  QAD  ICT  source  codes  generator                    */
/*V8:ConvertMode=Maintenance                                            */

{mfdeclre.i}
{icdef.i}

define  input  parameter  ipcScreenValue  as  character  no-undo.
define  output  parameter  oplIsValid     as  logical    no-undo.


/* ipcScreenValue  - screen-value  of  field  which  is  validated
   oplIsValid      - output  parameter  which  pass  down  an  information
                     if  field  screen-value  is  valid.
                     YES  - value  is  valid
                     NO   - value  is  invalid                        */

/*example*************************************************************
*          checking  if  the  sales  order  with  typed  SO  number  exists
*    ***************************************************************
*
*   oplIsValid  =  can-find(so_mstr  where  so_nbr  =  ipcScreenValue).
*
*********************************************************************/
```

QAD ICT Tasks

This section describes programs on the QAD ICT Tasks menu (90.23).

You can create a task and link further data changes to it. The following data is monitored: ICT settings (qad_wkfl table), labels (lbl_mstr and lbld_det tables), browses, and so on. Data changes related to ICT (qad_wkfl table) are invisible to other users until the task is closed.

Note You cannot edit an existing entry if it is linked with another ICT WIP task. Close the original ICT task first. You can then edit all entries linked with the task in another ICT task.

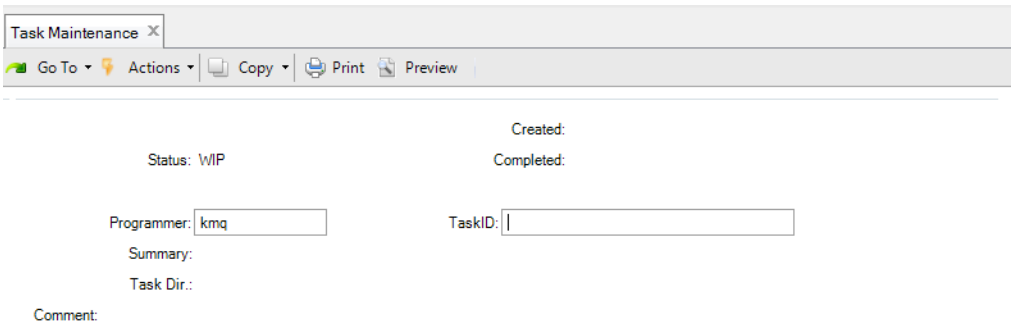
Beginning with ICT version 4.0, the functionality of the tasks has been extended. There are three new parameters in the ICT Control Table (90.24.24):

- **Task base directory,** used to indicate where task sub-directories are created. Task directory will be obtained by concatenating the base directory and the task ID. The new or modified custom procedures will be kept in the task directory and compiled there as long as the task is open. Once the task is closed, all such files will be moved automatically to the central directory (r-code to global_user_lang_dir + first two letters subdirectory, source code files to the first source subdirectory) and thus become visible to all users.
- **Central directory,** used to indicate the directory where the customer customizations are stored (subscriber, UI triggered, validation, other non ICT-related procedures). This directory will usually be the first one on the Propath.
- **Source sub-directories** define a comma-separated list of source subdirectories used to extend the Propath. The first subdirectory will be the target subdirectory for customized source files in the central directory.

Task Maintenance (90.23.1)

Use Task Maintenance to create a new or select some existing task. After a task is selected, the Propath is extended to include the task directory (at the beginning) and other subdirectories required for compilation. All changes made with the task selected will be invisible outside the task until the task is closed.

Note You can choose and create only your own task; otherwise, a “No user permission” warning will be displayed.



Status. Status of the task: WIP (work in process) or Closed. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the task. Mandatory.

Summary. A short description of the task. Mandatory.

Task Dir. The main directory for the task source code and other files. Mandatory.

Comment. Additional information. Optional.

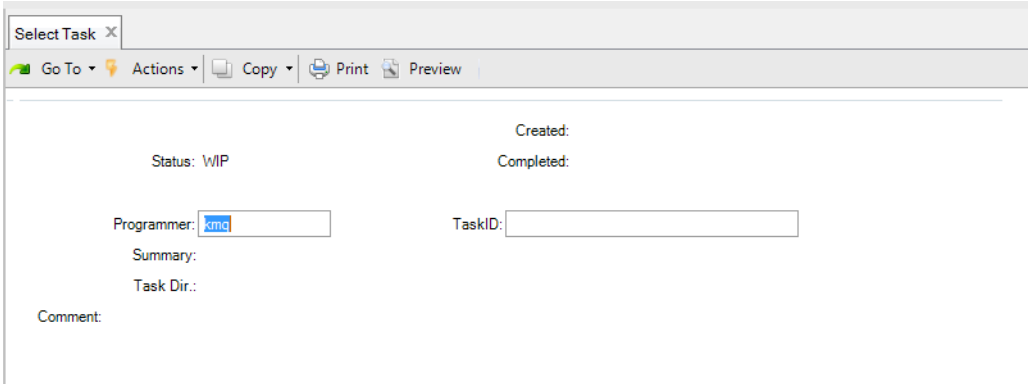
Using the Program

1. Run Task Maintenance using the standard QAD Enterprise Applications system menu.
2. Enter the name of the task you want to create or select an existing one.
3. Enter or change the summary of the task and the task directory.
4. Add/modify the comment.
5. In .NET UI, the information that Session Propath has been reset will be displayed. Confirm.

Select Task (90.23.4)

This function is similar to Task Maintenance; however, the task-related data are not enabled for modification. After selection, the Propath is extended with the current task directory.

Note You can choose only your own task; otherwise, a “No user permission” warning will be displayed.



Status. Status of the task: WIP (work in process) or Closed. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the task. Mandatory.

Summary. A short description of the task. Read only.

Task Dir. The main directory for the task source code and other files. Read only.

Comment. Additional information. Read only.

Using the Program

1. Run Select Task using the standard QAD Enterprise Applications system menu.
2. Find the task you want to select (you can select your tasks only).
3. In .NET UI, the information that Session Propath has been reset will be displayed. Confirm.

Task Reset (90.23.5)

Task Reset was developed to switch off the QAD ICT task under which the developer is working. After opening the function, the task that is currently active will be automatically reset. (Use Task Maintenance to reactivate the task.) The Propath modified after selection of the task will be changed to the original (system) one.

```
ictaskre.p ict4 a              90.23.5 Task Reset                    03/20/12
+-------------------------------------------------------------------------+
|                      TaskID: 1919test                                   |
+-------------------------------------------------------------------------+
         +----------------------Current PROPATH-----------------------+
         | Session Propath has been re-set                           |
         |+---------------------------------------------------------+|
         ||  /qad/custom/92b/qict92b11/work                         ||
         ||  /qad/custom/92b/qict92b11/dev                          ||
         ||  /qad/custom/92b/qict92b11/dev/src                      ||
         ||  /qad/custom/92b/qict92b11/dev/xrc                      ||
         ||  /qad/custom/92b/qict92b11/test                         ||
         ||  /qad/custom/92b/qict92b11/test/src                     ||
         ||  /qad/custom/92b/qict92b11/test/xrc                     ||
         ||  /qad/custom/92b/qict92b11/stage                        ||
         |+---------------------------------------------------------+|
         +-----------------------------------------------------------+
```

TaskID. Name of the currently chosen task. Read only.

Using the Program

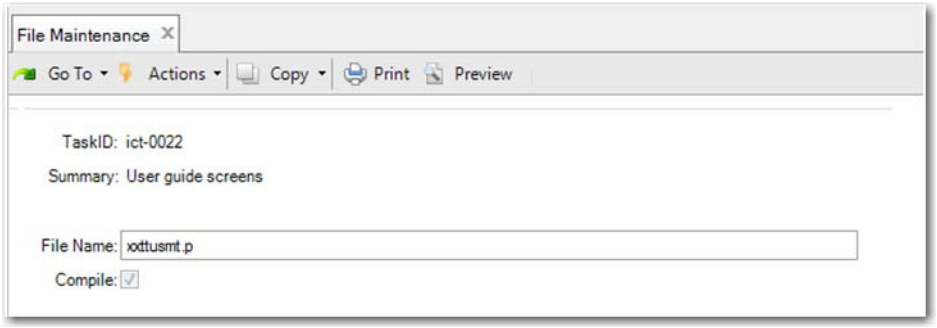
1 Run Task Reset using the standard QAD Enterprise Applications system menu.

Note This program is available in .NET UI. However, it does not show the current Propath.

2 Confirm the information about resetting the Propath.

File Maintenance (90.23.8)

This function allows the user to add/delete files linked with the current task.



TaskID. Identifier of the current task. Read only.

Summary. The summary of the current task. Read only.

File name. Name of the file. Can be entered directly or chosen from the existing files using the up and down arrows. Mandatory.

Compile. Specify whether the file should be compiled by ICT (Progress procedure file) or not (other types of files). Mandatory.

Using the Program

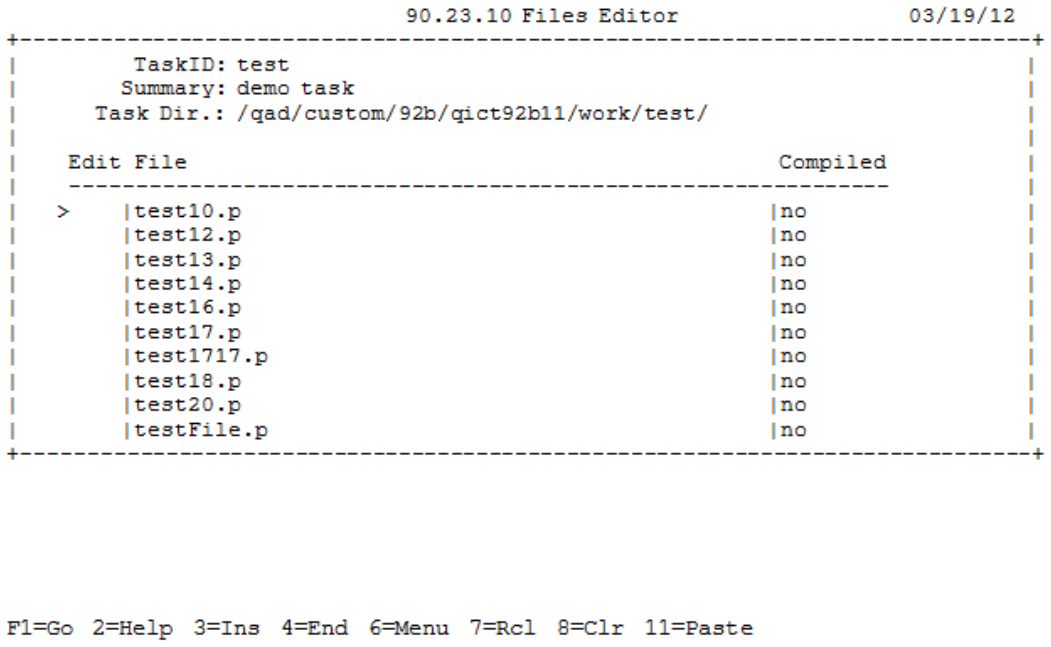
1 Run File Maintenance using the standard QAD Enterprise Applications system menu.

2 Enter the name of a new file or an existing one, or select it using the up and down arrows.

3 To delete an existing file, use the Delete button.

Files Editor (90.23.10)

Use Files Editor to select and open files linked with the current task in the Progress Editor.



TaskID. Identifier of the currently selected task. Read only.

Summary. Summary of the currently selected task. Read only.

Task Dir. Current task directory. Read only.

Edit. Files you want to edit should be marked with an asterisk (*). Mandatory.

File. File name of the files linked with the current task. Read only.

Compiled. Flag that indicates whether the file is compiled with the current version of the source procedure (if the r-code is created after the last modification of the procedure). Read only.

Using the Program

1 Run Files Editor using the standard QAD Enterprise Applications system menu.

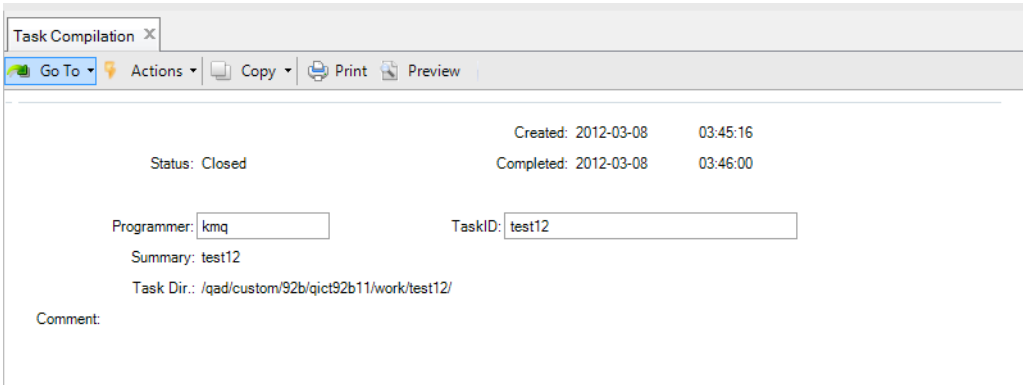
Note This program is not available in .NET UI.

2 Select the files you want to edit with the space bar (selected file is marked with an asterisk).

3 Apply your choice with F1. Progress Editor should be opened.

Task Compilation (90.23.11)

Task Compilation allows users to compile any closed task or the currently selected one. Closed tasks are compiled into the central directory. The current task should be compiled into the local directory.



Status. Status of the task: WIP (work in process) or Closed. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the task. Mandatory.

Summary. A short description of the task. Read only.

Task Dir. The main directory for the task source code and others. Read only.

Comment. Additional information. Read only.

Using the Program

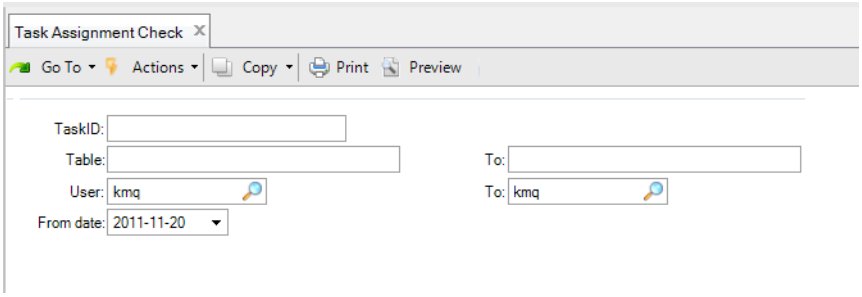
1 Run Task Compilation using the standard QAD Enterprise Applications system menu.

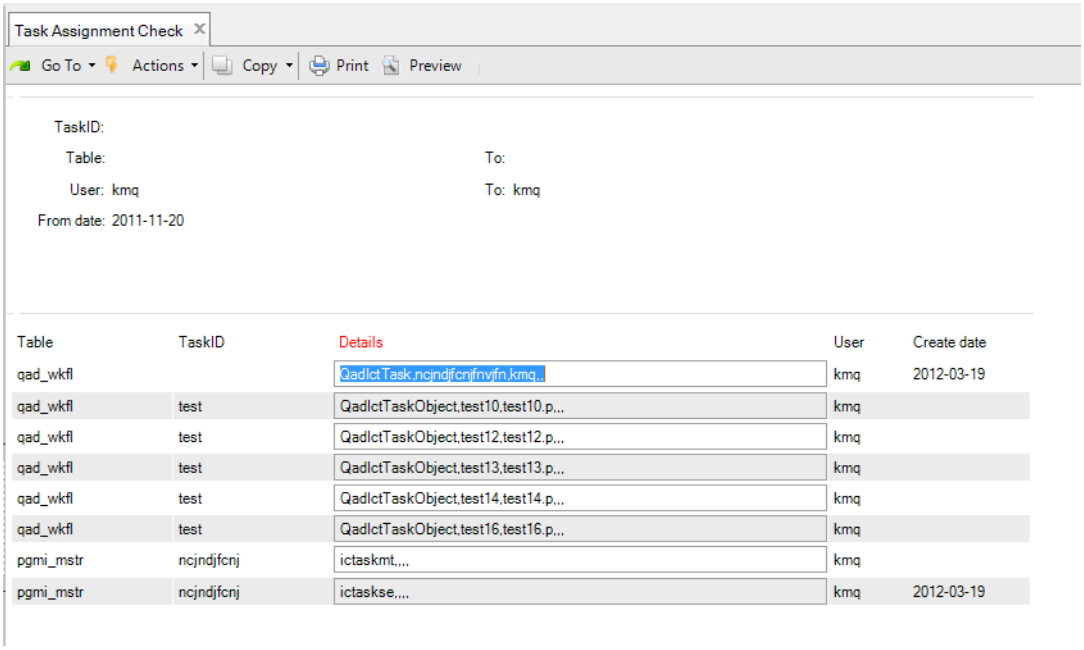
2 Choose any closed task or the current WIP task. You can compile only your task. If you compile a closed task, the source code is taken from the central directory, and all files are compiled to the central directory.

3 Apply your choice with the F1 key. The files will be compiled, and output from the compilation will be displayed on the screen.

Task Assignment Check (90.23.13)

Task Assignment Check was developed to provide an overview of changes done under a particular QAD ICT task. Specify the selection criteria in the first frame; the next frame displays all the changes.





TaskID. Task identifier. Optional.

Table, To. Range of the tables. Optional.

User, To. Range of the user logins. Optional.

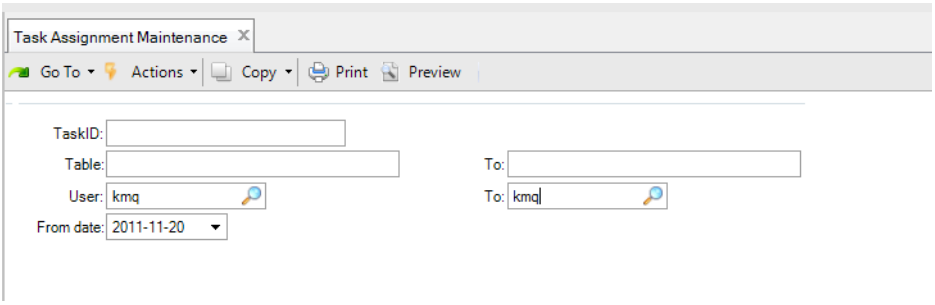
From date. Range of dates, from “from date” to today. Optional.

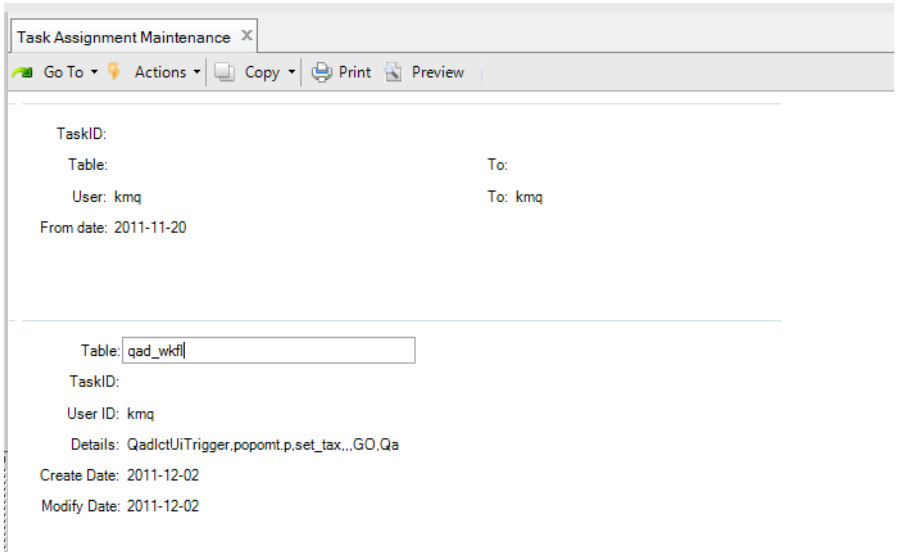
Using the Program

1. Run Task Assignment Check using the standard QAD Enterprise Applications system menu.
2. Specify the selection criteria.
3. Display the records that satisfy the defined condition.

Task Assignment Maintenance (90.23.14)

Use this function to delete or change assignments to the selected task. If there is a change that was registered under a specific task, you can delete this assignment so that the change will not be included in the file when the records are dumped. You can also change the assignment so that the changes are dumped under a different task.





TaskID. Task identifier. Optional.

Table, To. Range of the tables. Optional.

User, To. Range of the user logins. Optional.

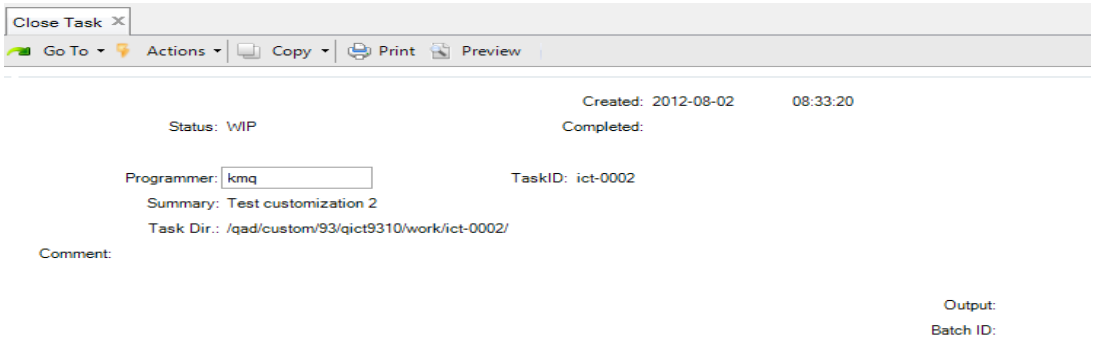
From date. Range of dates, from “from date” to today. Optional

Using the Program

1. Run Task Assignment Maintenance using the standard QAD Enterprise Applications system menu.
2. Specify the selection criteria.
3. You can display a single record from the set of records that satisfies the criteria using the up and down arrows. Then you can delete or modify each entry.

Close Task (90.23.16)

Close Task allows the user to close the current task. The task will be closed if all the related files have been compiled successfully. Then the files (source and r-code) linked to the task will be moved into the proper subdirectories of the central directory and the task directory deleted. Finally, QAD ICT-related changes should be made visible to all users.



Status. Status of the task: WIP (work in process) or Closed. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the currently selected task. Read only.

Summary. A short description of the task. Read only.

Task Dir. The main directory for the task source code and others. Read only.

Comment. Additional information. Read only.

Using the Program

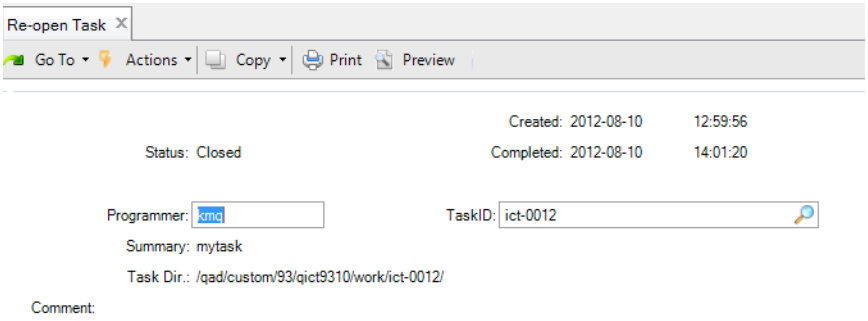
1. Run Close Task using the standard QAD Enterprise Applications system menu.

   Note A task you want to close must be selected; otherwise, the system displays “Task not selected.”

2. Confirm that the displayed task is the one you want to close and press F1 or Enter.
3. Answer Yes to the prompt about closing the task.
4. Choose the output device where the result should be displayed, printed, or saved.
5. The system displays, prints, or saves the result of closing the task. The task directory is deleted.

Re-open Task (90.23.17)

Re-open Task allows the user to reopen any closed task. The reopen task is automatically selected.



Status. Status of the task: WIP (work in process) or Closed. Only a closed task can be reopened. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the currently selected task. Mandatory.

Summary. A short description of the task. Read only.

Task Dir. The main directory for the task source code and others. Read only.

Comment. Additional information. Read only.

Using the Program

1. Run Re-open Task using the standard QAD Enterprise Applications system menu.

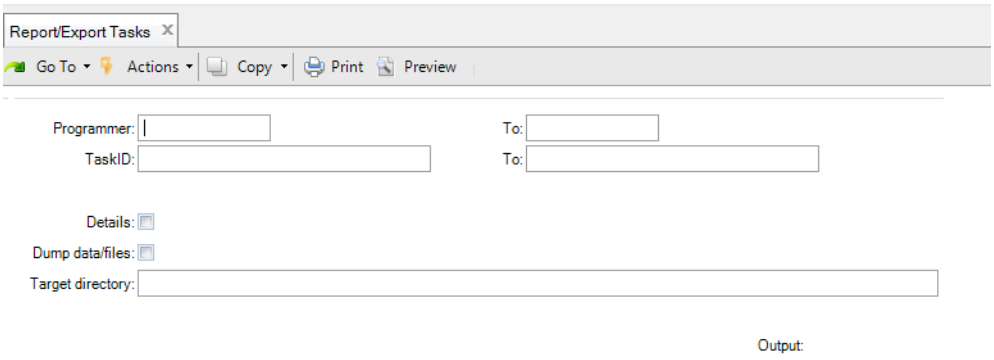
   Note No task can be selected; otherwise, the system displays “Task name is selected. Reset first.”

2. Choose the closed task (you can choose any of the tasks marked as closed—not only yours).

3. Confirm the selection. The reopened task has the WIP status, and the Propath is set to the current task directory.

Report/Export Tasks (90.23.18)

Report/Export Tasks reports details and contents of specified tasks and optionally dumps all related data into a `.dmp` file and moves the files to the specified directory. That way, it is easier to move customizations between different systems.



Programmer, To. Range of logins of programmers. Optional.

TaskID, To. Range of task identifiers. Optional.

Details. Determines how detailed the report should be. Optional.

No: Only task data is displayed.

Yes: The report also includes related data.

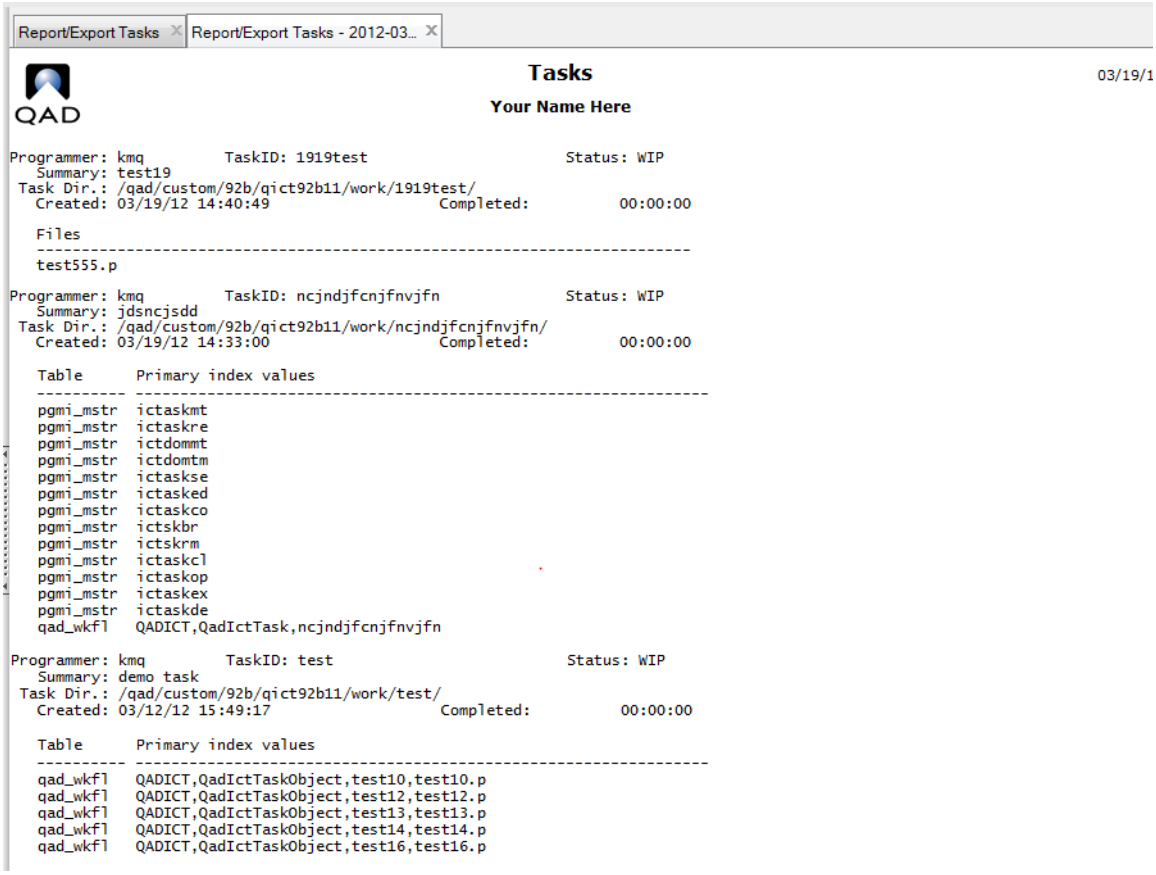
Dump data/files. Determines if the data that satisfies the specified condition should be saved in the `.dmp` file. Optional.

Target directory. A directory where the `.dmp` file should be saved (if created). Must point to an existing directory if Dump data/files is Yes. Mandatory if Dump data/files is Yes.

Using the Program

1. Run Report/Export Tasks using the standard QAD Enterprise Applications system menu.
2. Specify the selection criteria.
3. Define if you want to generate a detailed report or only include task summary data.
4. Define if data should be dumped to the file. If Yes, also specify the directory where the `.dmp` file should be saved.

Example of a Detailed Report



Report/Export Tasks | Report/Export Tasks - 2012-03...

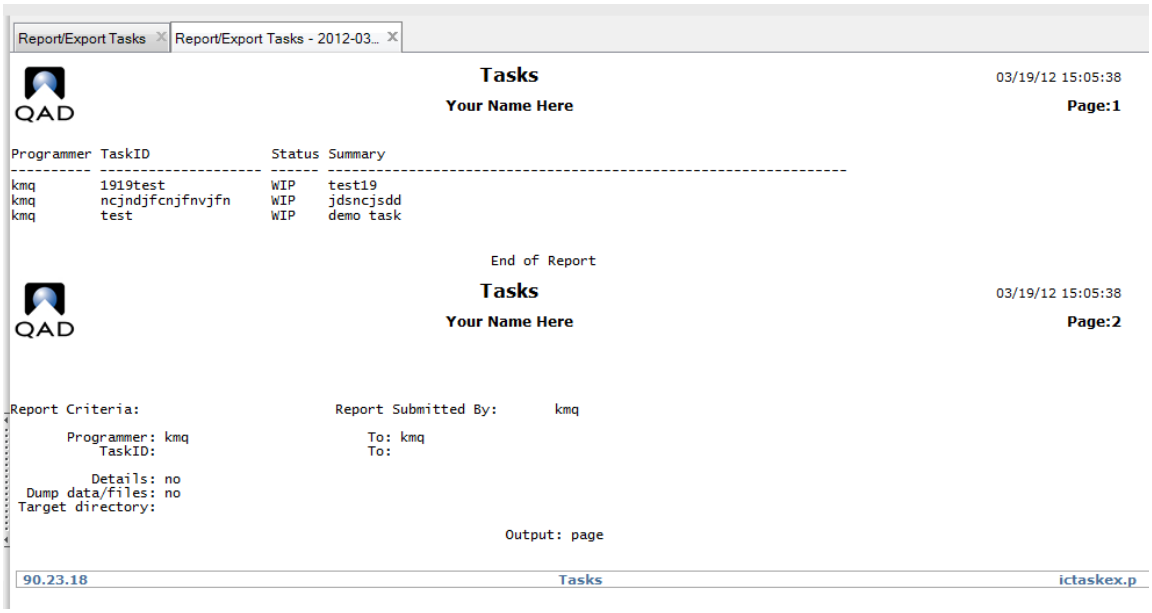
QAD

Tasks 03/19/1

Your Name Here

```
Programmer: kmq         TaskID: 1919test                    Status: WIP
   Summary: test19
 Task Dir.: /qad/custom/92b/qict92b11/work/1919test/
   Created: 03/19/12 14:40:49                 Completed:          00:00:00

   Files
   ------------------------------------------------------------------------
   test555.p

Programmer: kmq         TaskID: ncjndjfcnjfnvjfn            Status: WIP
   Summary: jdsncjsdd
 Task Dir.: /qad/custom/92b/qict92b11/work/ncjndjfcnjfnvjfn/
   Created: 03/19/12 14:33:00                 Completed:          00:00:00

   Table      Primary index values
   ---------- ------------------------------------------------------------
   pgmi_mstr  ictaskmt
   pgmi_mstr  ictaskre
   pgmi_mstr  ictdommt
   pgmi_mstr  ictdomtm
   pgmi_mstr  ictasksе
   pgmi_mstr  ictasked
   pgmi_mstr  ictaskco
   pgmi_mstr  ictskbr
   pgmi_mstr  ictskrm
   pgmi_mstr  ictaskcl
   pgmi_mstr  ictaskop
   pgmi_mstr  ictaskex
   pgmi_mstr  ictaskde
   qad_wkfl   QADICT,QadIctTask,ncjndjfcnjfnvjfn

Programmer: kmq         TaskID: test                        Status: WIP
   Summary: demo task
 Task Dir.: /qad/custom/92b/qict92b11/work/test/
   Created: 03/12/12 15:49:17                 Completed:          00:00:00

   Table      Primary index values
   ---------- ------------------------------------------------------------
   qad_wkfl   QADICT,QadIctTaskObject,test10,test10.p
   qad_wkfl   QADICT,QadIctTaskObject,test12,test12.p
   qad_wkfl   QADICT,QadIctTaskObject,test13,test13.p
   qad_wkfl   QADICT,QadIctTaskObject,test14,test14.p
   qad_wkfl   QADICT,QadIctTaskObject,test16,test16.p
```

Example of a Non-Detailed Report



Report/Export Tasks | Report/Export Tasks - 2012-03...

QAD

Tasks 03/19/12 15:05:38

Your Name Here **Page:1**

```
Programmer TaskID              Status Summary
---------- ------------------- ------ ------------------------------------------------------------------------
kmq        1919test            WIP    test19
kmq        ncjndjfcnjfnvjfn    WIP    jdsncjsdd
kmq        test                WIP    demo task

                                End of Report
```

QAD

Tasks 03/19/12 15:05:38

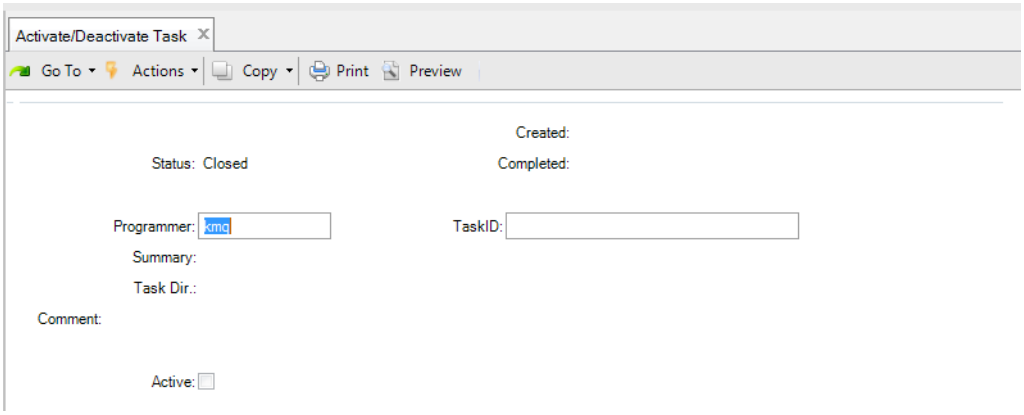
Your Name Here **Page:2**

```
Report Criteria:                    Report Submitted By:      kmq

      Programmer: kmq                   To: kmq
          TaskID:                       To:

         Details: no
 Dump data/files: no
Target directory:

                                    Output: page
```

90.23.18 Tasks ictaskex.p

Activate/Deactivate Task (90.23.19)

Activate/Deactivate Task allows the user to activate or deactivate modifications connected with a closed task. All of the customizations are automatically activated/deactivated in the associated maintenance programs.



Status. Status of the task. Read only.

Created. Date of the creation of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Completed. Date of closing of the task. Read only.

Note For tasks migrated from versions of ICT earlier than 4.0, this field will be empty.

Programmer. Programmer login. Mandatory.

TaskID. Name of the currently selected task. Mandatory.

Summary. A short description of the task. Read only.

Task Dir. The main directory for the task source code and others. Read only.

Comment. Additional information. Read only.

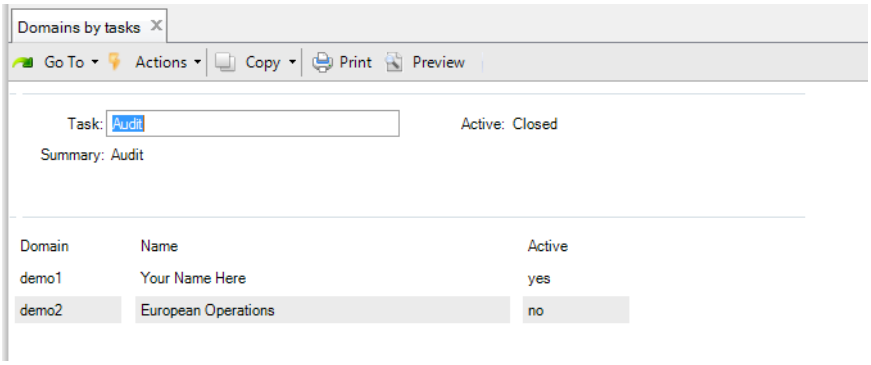
Active. Determines whether the customizations related to the selected task should be active. Mandatory.

Using the Program

1. Run Activate/Deactivate Tasks using the standard QAD Enterprise Applications system menu.
2. Choose the task you want activate or deactivate. Only closed tasks are allowed.
3. Decide if the customizations related to the selected task should be active.

Domains by Tasks (90.23.22)

Domains by Tasks lets you list all domains for each task and decide in which domain the task should be active. You can modify the access domain only for closed tasks.



Task. Task identifier. Mandatory.

Active. Task status. Only closed tasks can be modified. Read only.

Summary. A summary of the task. Read only.

Domain. The domain name. Read only.

Name. The domain description. Read only.

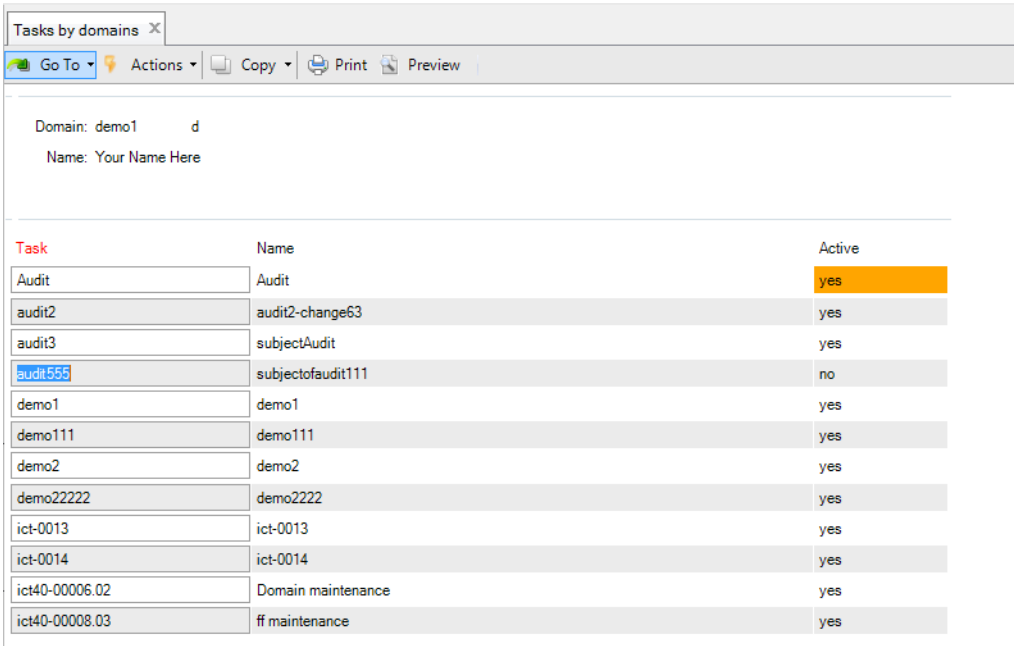
Active. Determines if the selected task should be active in the specified domain. Mandatory.

Using the Program

1. Run Domains by Tasks using the standard QAD Enterprise Applications system menu.
2. Select the task (all tasks are listed but only the closed one can be selected).
3. Decide in which domain the changes linked with the task should be enabled.

Tasks by Domains (90.23.23)

Tasks by Domains lets you list all closed tasks and decide which task should be active in the selected domain.



Domain. The domain name. Mandatory.

Name. The domain description. Read only.

Task. Task identifier. Read only.

Summary. Name of the task. Read only.

Active. Determines if the task should be active in the selected domain. Mandatory.

Using the Program

1. Run Tasks by Domains using the standard QAD Enterprise Applications system menu.
2. Select the domain.
3. Choose which tasks should be enabled in the selected domain.

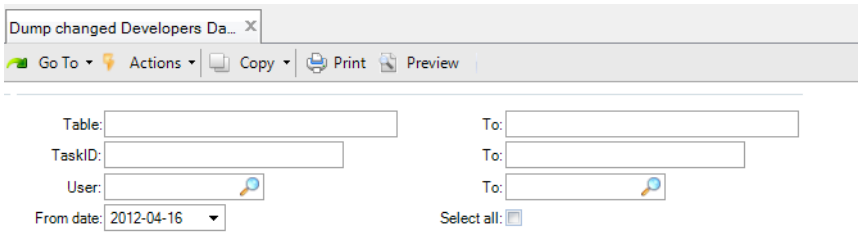
QAD ICT Control Menu

This section describes programs on the QAD ICT Control Menu (90.24).

Dump Changed Developers Data (90.24.1)

Dump Changed Developers Data allows users to dump the modified data into a file. The system registers the changes made by the user only when the Active change triggers flag is set to Yes in ICT Control Table (90.24.24).

Note This program is not available in .NET UI.



Table, To. Selection criteria for the tables. Optional.

TaskID, To. Selection criteria for the task IDs. Optional.

User, To. Selection criteria for the users who changed the data. Optional.

From date. Selection criteria for the date of data modification (period from date – today). Optional.

Select all. Determines whether all records that satisfy selection criteria are selected. If it is set to No, only records that have not yet been exported to a `.dmp` file are selected. Optional.

Using the Program

1. Run Dump Changed Developers Data using the standard QAD Enterprise Applications system menu.
2. Define the selection criteria.

   Note If you choose the Select all option, all records are selected by default. Otherwise, only records that have not yet been exported to a `.dmp` file are selected.
3. Mark the records that should be dumped (records can be marked/unmarked by pressing the space bar).
4. Define the delivery number (Delivery nbr). The system uses this number as a file name.

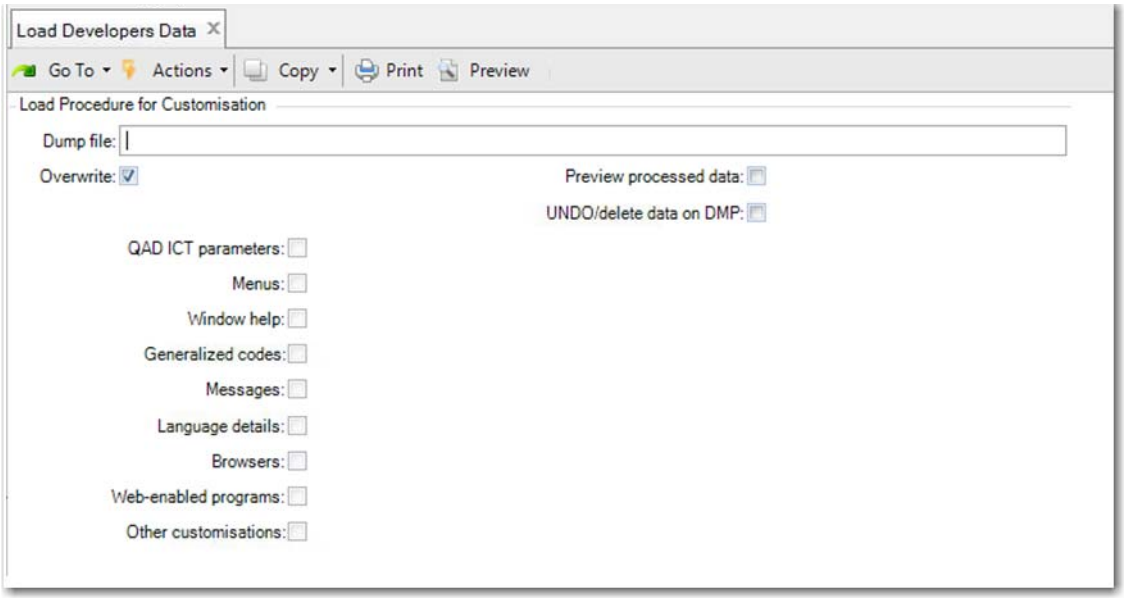
   Note ICT will add the `.dmp` file extension automatically; do not add it manually to the delivery number.

Load Developers Data (90.24.2)

Load Developers Data allows users to load the modification data from a file to QAD Enterprise Applications. Beginning with ICT 4.0, users can also delete the data from QAD Enterprise Applications.

Note The file code page and session code page must be the same. For that reason, in some versions of QAD Enterprise Applications it is impossible to load data generated by the Character UI into the .NET UI and vice versa.

Note Beginning with ICT 4.1, the `.dmp` file header contains information about the ICT version where it was generated. Every time you load a `.dmp` file from a version earlier than 4.0, remember to run conversion procedures delivered with your ICT 4.x installation (`ictcvn35.p` and `ictcnv34.p`).



Dump File. Dump file from which data should be loaded, or that defines the data to be deleted. Mandatory.

Note The full name of the file is required (name + `.dmp` extension).

Overwrite. Added in ICT 4.1. Determines whether existing records should be overwritten if there are duplicate names. Optional.

Preview processed data? Determines if data from the `.dmp` file should be previewed before loading. Optional.

UNDO/delete data on DMP. Determines if data should be deleted from the system. If this flag is set to Yes, ICT will remove the data from QAD Enterprise Applications, otherwise the data from the file will be loaded into the system. Mandatory.

Using the Program

1. Run Load Developers Data using the standard QAD Enterprise Applications system menu.
2. Enter the name of the `.dmp` file (with extension) that should be loaded or that points to the data to be deleted.

3 Decide whether the data from the DMP file should be loaded into the system (UNDO/delete data on DMP set to No). Otherwise (UNDO/delete data on DMP set to Yes), all data that is listed in the file will be deleted from the system.

4 Decide if you want to load or delete all data in the file or only selected data. If you want to choose data that should be loaded/deleted, set Preview processed data? to Yes.

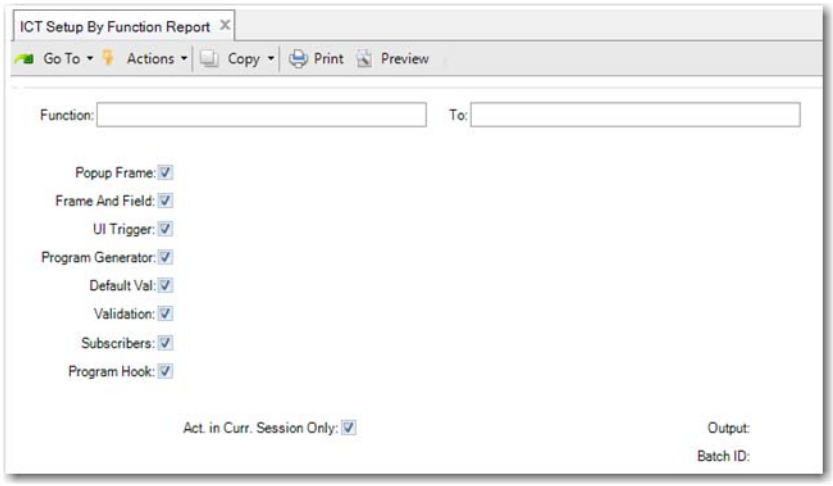
5 Decide if you want to overwrite existing records if ICT finds duplicates.

6 If Preview processed data? was Yes, choose the proper data to load/delete (using the space bar).

7 Answer Yes to the warning “Load system data?” Information about the deleting/loading process will be displayed.

ICT Setup By Function Report (90.24.8)

This report allows you to display ICT setup related to specified functions (execnames). You can select what types of customizations should be listed.



Function/To. Determines the range of functions (execnames) included in the report. Optional.

Popup Frame. Determines whether added custom frames should be included in the report. Optional.

Frame And Field. Determines whether changes in the frames and fields should be included in the report. Optional.

UI Trigger. Determines whether UI triggers should be included in the report. Optional.

Program Generator. Determines whether program generators should be included in the report. Optional.

Default Val. Determines whether default values for fields should be included in the report. Optional.

Validation. Determines whether validations should be included in the report. Optional.

Subscribers. Determines whether subscribers should be included in the report. Optional.

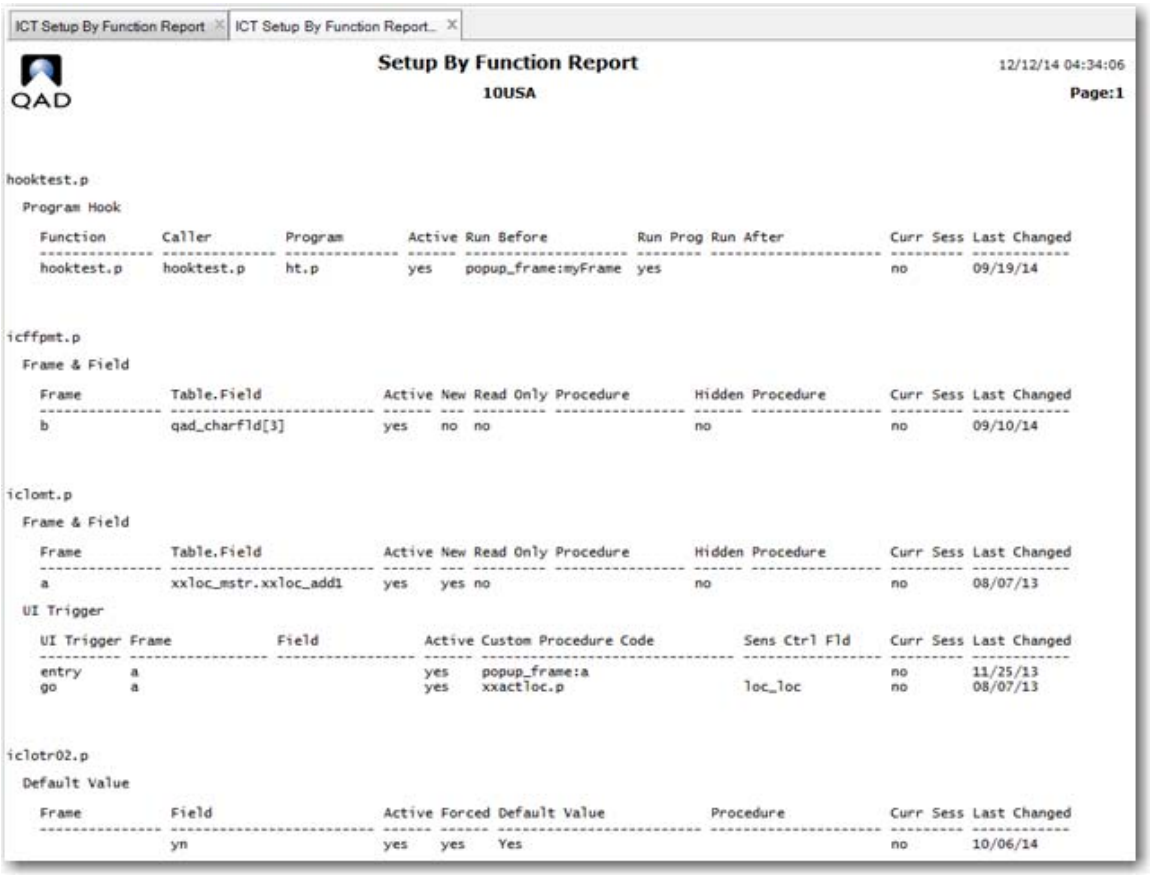
Program Hook. Determines whether program hooks should be included in the report. Optional.

Act. In Curr. Session Only. Determines whether only setup active in the current session should be included in the report. Optional.

Output. Output destination for the report. Mandatory.

Batch ID. Batch identifier. Optional.

Sample Output from Report



ICT Setup By Function Report | ICT Setup By Function Report...

QAD

Setup By Function Report 12/12/14 04:34:06

10USA Page:1

hooktest.p

Program Hook

Function	Caller	Program	Active	Run Before	Run Prog	Run After	Curr Sess	Last Changed
hooktest.p	hooktest.p	ht.p	yes	popup_frame:myFrame	yes		no	09/19/14

icffpmt.p

Frame & Field

Frame	Table.Field	Active	New	Read Only	Procedure	Hidden	Procedure	Curr Sess	Last Changed
b	qad_charfld[3]	yes	no	no		no		no	09/10/14

iclomt.p

Frame & Field

Frame	Table.Field	Active	New	Read Only	Procedure	Hidden	Procedure	Curr Sess	Last Changed
a	xxloc_mstr.xxloc_add1	yes	yes	no		no		no	08/07/13

UI Trigger

UI Trigger	Frame	Field	Active	Custom Procedure Code	Sens Ctrl Fld	Curr Sess	Last Changed
entry	a		yes	popup_frame:a		no	11/25/13
go	a		yes	xxactloc.p	loc_loc	no	08/07/13

iclotr02.p

Default Value

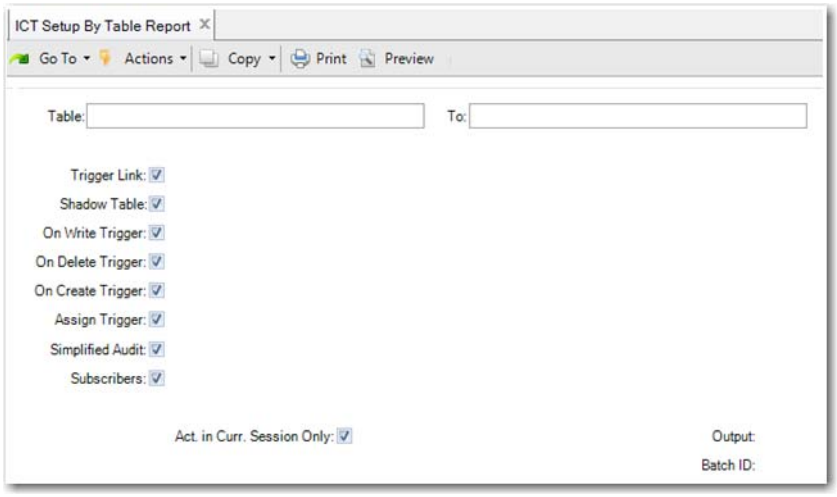
Frame	Field	Active	Forced	Default Value	Procedure	Curr Sess	Last Changed
	yn	yes	yes	Yes		no	10/06/14

Using the Program

1. Run the report using the standard QAD Enterprise Applications system menu.
2. Specify a range of execnames, select the types of ICT setup that you want to list, and select the output.
3. Analyze the report.
4. Close the report and exit ICT Setup By Function Report.

ICT Setup By Table Report (90.24.9)

This report allows you to display ICT setup related to a specified range of table names. You can select what types of customization should be listed.



Table/To. Determines the range of table names included in the report. Optional.

Trigger Link . Determines whether trigger links should be included in the report. Optional.

Shadow Table . Determines whether shadow tables should be included in the report. Optional.

On Write Trigger. Determines whether on write triggers should be included in the report. Optional.

On Delete Trigger. Determines whether on delete triggers should be included in the report. Optional.

On Create Trigger. Determines whether on create triggers should be included in the report. Optional.

Assign Trigger. Determines whether on assign triggers should be included in the report. Optional.

Simplified Audit. Determines whether simplified audits should be included in the report. Optional.

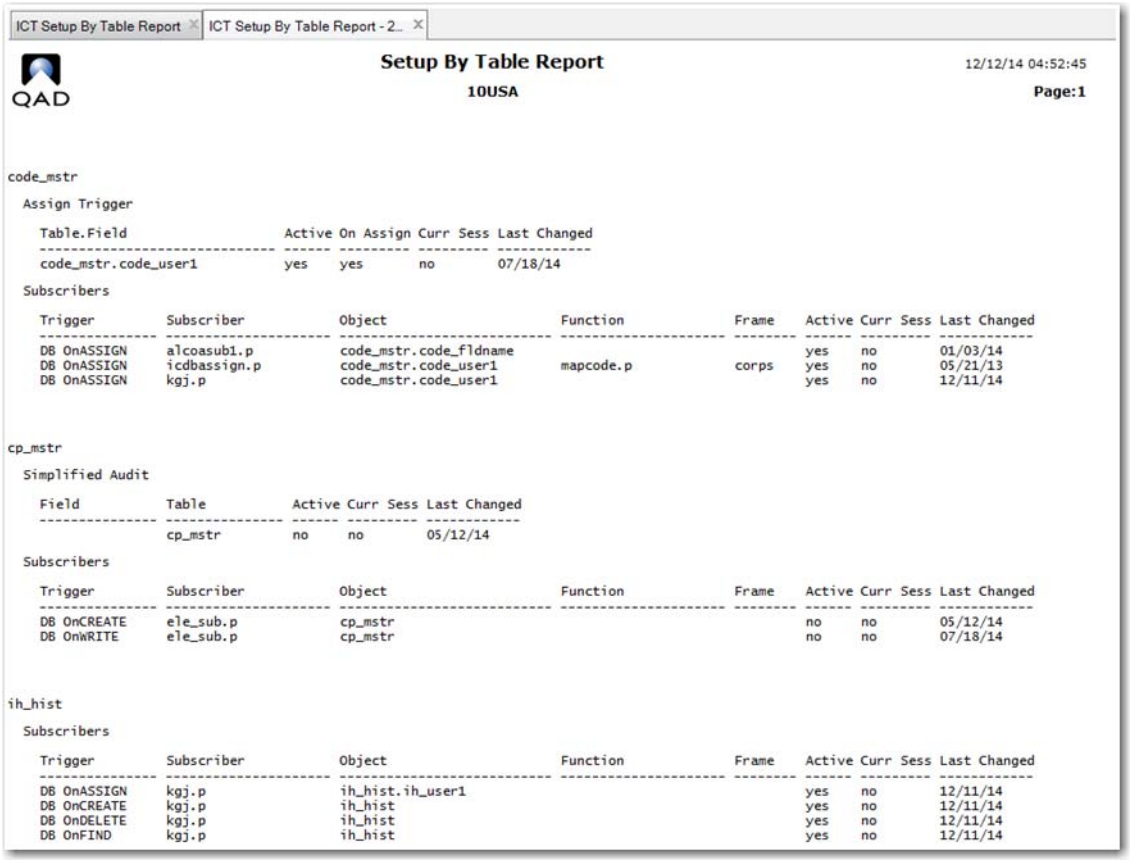
Subscribers. Determines whether related subscribers should be included in the report. Optional.

Act. In Curr. Session Only. Determines whether only setup active in the current session should be included in the report. Optional.

Output. Output destination for the report. Mandatory.

Batch ID. Batch identifier. Optional.

Sample Output from Report



ICT Setup By Table Report | ICT Setup By Table Report - 2...

QAD

Setup By Table Report

10USA

12/12/14 04:52:45

Page:1

code_mstr

Assign Trigger

Table.Field	Active	On Assign	Curr Sess	Last Changed
code_mstr.code_user1	yes	yes	no	07/18/14

Subscribers

Trigger	Subscriber	Object	Function	Frame	Active	Curr Sess	Last Changed
DB OnASSIGN	alcoasub1.p	code_mstr.code_fldname			yes	no	01/03/14
DB OnASSIGN	icdbassign.p	code_mstr.code_user1	mapcode.p	corps	yes	no	05/21/13
DB OnASSIGN	kgj.p	code_mstr.code_user1			yes	no	12/11/14

cp_mstr

Simplified Audit

Field	Table	Active	Curr Sess	Last Changed
	cp_mstr	no	no	05/12/14

Subscribers

Trigger	Subscriber	Object	Function	Frame	Active	Curr Sess	Last Changed
DB OnCREATE	ele_sub.p	cp_mstr			no	no	05/12/14
DB OnWRITE	ele_sub.p	cp_mstr			no	no	07/18/14

ih_hist

Subscribers

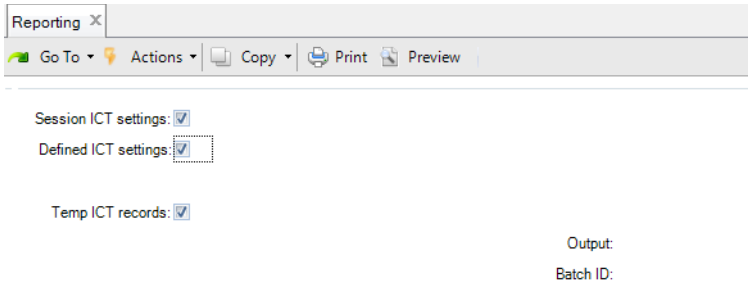
Trigger	Subscriber	Object	Function	Frame	Active	Curr Sess	Last Changed
DB OnASSIGN	kgj.p	ih_hist.ih_user1			yes	no	12/11/14
DB OnCREATE	kgj.p	ih_hist			yes	no	12/11/14
DB OnDELETE	kgj.p	ih_hist			yes	no	12/11/14
DB OnFIND	kgj.p	ih_hist			yes	no	12/11/14

Using the Program

1. Run the report using the standard QAD Enterprise Applications system menu.
2. Specify the range of tables, select the types of ICT setup that you want to list, and select the output.
3. Analyze the report.
4. Close the report and exit ICT Setup By Table Report.

Reporting (90.24.10)

Reporting is a tool that allows you to generate a report on defined and loaded ICT settings.



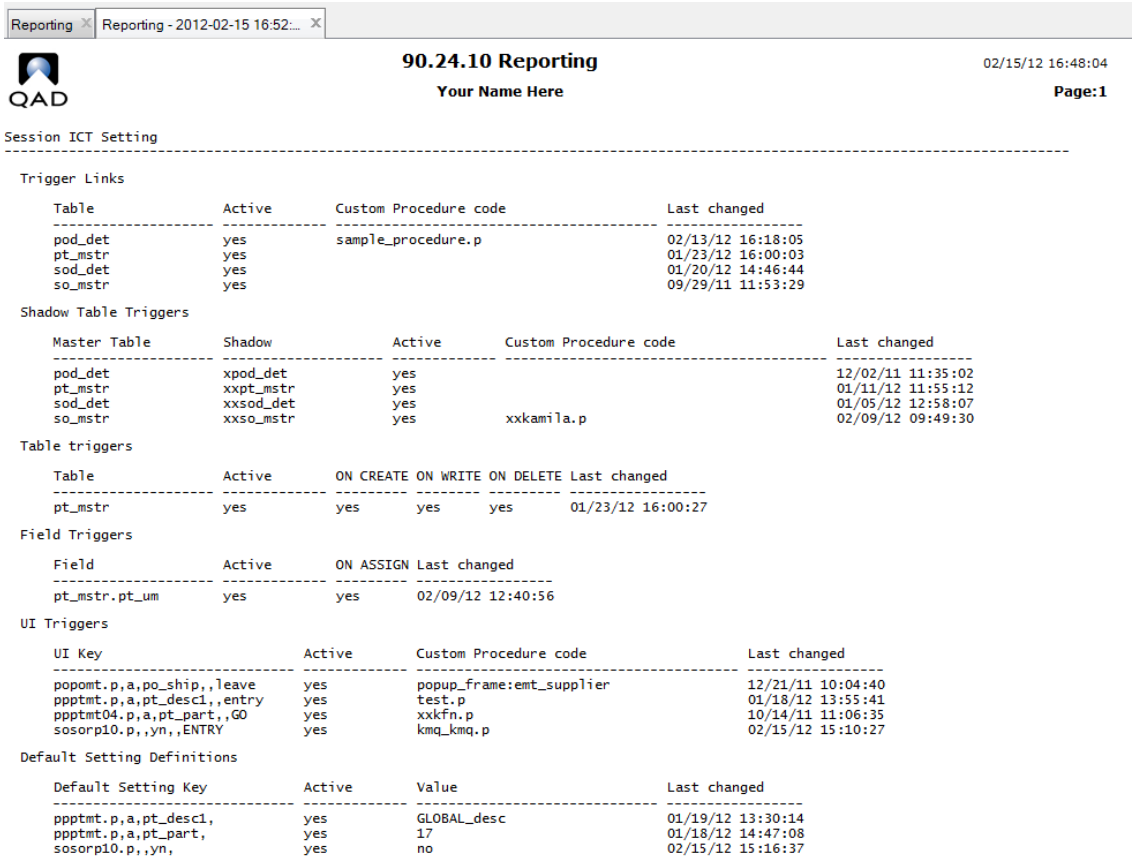
Session ICT settings. Determines whether settings loaded for the current session should be reported.

Defined ICT settings. Determines whether settings defined in ICT (not necessarily active in the current session) should be reported.

Temp ICT records. Determines whether temporary ICT records should be reported.

Output and Batch ID. Output destination for the report.

Sample Output from Reporting



Reporting | Reporting - 2012-02-15 16:52:...

QAD

90.24.10 Reporting 02/15/12 16:48:04

Your Name Here **Page:1**

Session ICT Setting

Trigger Links

Table	Active	Custom Procedure code	Last changed
pod_det	yes	sample_procedure.p	02/13/12 16:18:05
pt_mstr	yes		01/23/12 16:00:03
sod_det	yes		01/20/12 14:46:44
so_mstr	yes		09/29/11 11:53:29

Shadow Table Triggers

Master Table	Shadow	Active	Custom Procedure code	Last changed
pod_det	xpod_det	yes		12/02/11 11:35:02
pt_mstr	xxpt_mstr	yes		01/11/12 11:55:12
sod_det	xxsod_det	yes		01/05/12 12:58:07
so_mstr	xxso_mstr	yes	xxkamila.p	02/09/12 09:49:30

Table triggers

Table	Active	ON CREATE	ON WRITE	ON DELETE	Last changed
pt_mstr	yes	yes	yes	yes	01/23/12 16:00:27

Field Triggers

Field	Active	ON ASSIGN	Last changed
pt_mstr.pt_um	yes	yes	02/09/12 12:40:56

UI Triggers

UI Key	Active	Custom Procedure code	Last changed
popomt.p,a,po_ship,,leave	yes	popup_frame:emt_supplier	12/21/11 10:04:40
ppptmt.p,a,pt_desc1,,entry	yes	test.p	01/18/12 13:55:41
ppptmt04.p,a,pt_part,,GO	yes	xxkfn.p	10/14/11 11:06:35
sosorp10.p,,yn,,ENTRY	yes	kmq_kmq.p	02/15/12 15:10:27

Default Setting Definitions

Default Setting Key	Active	Value	Last changed
ppptmt.p,a,pt_desc1,	yes	GLOBAL_desc	01/19/12 13:30:14
ppptmt.p,a,pt_part,	yes	17	01/18/12 14:47:08
sosorp10.p,,yn,	yes	no	02/15/12 15:16:37

Using the Program

1. Run Reporting using the standard QAD Enterprise Applications system menu.
2. Specify the query parameters and select the output.
3. Analyze the report.
4. Close the report and exit the Reporting function.

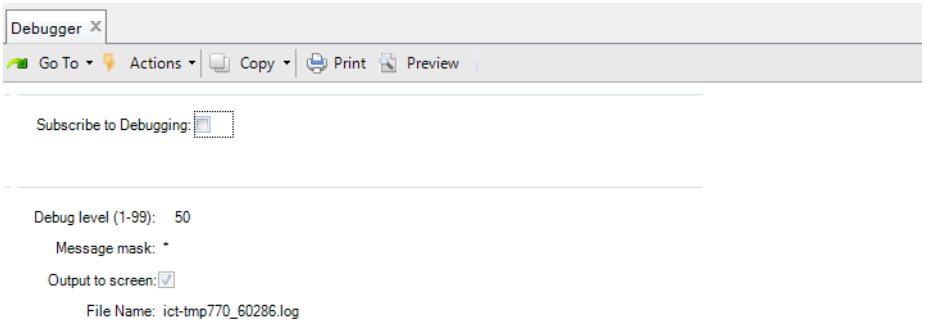
Debugger (90.24.11)

Debugger is a tool that allows the QAD ICT programmers to:

- **Debug the QAD ICT processing.** All QAD ICT processing functions have debugging breakpoints implemented. When you make the debugger active, the system displays messages about the process currently being performed, including key parameters.
- **Debug the customization.** Using the `ictdbg.i` include file in customized files, the ICT programmers customizing the system can define their own debugging breakpoints and parameters, which will be displayed when the customized program is running.

Note Debugging is active for a single session.

Note Debugging is not available in the .NET UI and Desktop interface.



Subscribe to Debugging. Determines whether debugging should be available in the current session. Mandatory.

Debug level (1-99). A level for the debugging. QAD ICT Debugger will filter debug messages from 0 to the specified level. Mandatory.

The internal ICT processing functions are assigned to the following levels:

Level	Processing
80	Executing internal or external source code attached to the Trigger & Consistency links
90	Occurrence of the GO, ENTRY, LEAVE interface triggers
99	Internal ICT processing, such as: • Default values processing • Validations • Frame and field properties processing • UI triggers processing • Registering ICT temporary records

Message mask. The type of processing you want. Examples of masks are shown in the following table. Optional.

Mask	Displayed Breakpoint
REGISTER	Creating temporary records
*RUN*IN*	Customized program of function (from super library) execution
SUPER	
GPRUN.I	Customized external program execution
VALIDATION	Validation process
DEFAULT VALUE	Setting default values processing
Frame & FIELD	Performing the Frame and Fields properties processing

Note Remember to open and close the mask definition using an asterisk (*).

Output to screen. Determines whether debug messages should be displayed on the screen. Optional.

File Name. Name of the file that will store the debug messages. The system generates this value automatically based on a unique session number. Read only.

Using the Program

1 Put the `icdbg.i` include file into your customized procedure. You can use the following parameters:

Parameter	Description
&T1 - &T3	Optional parameters for text for the debug message
&V1 - &V8	Optional parameters for variables for the debug message
&LEVEL	Optional debug level of the message, default is 50
&PROGRAM	Optional parameter for program stack information, default is program-name(1) and program-name(2)

Sample Code (for sample code for QAD Enterprise Applications Enterprise Edition 2011 and higher, see Appendix A on page 107):

```
{icdef.i}

define variable lvrRowid as rowid no-undo.
define variable l as character no-undo.

l = "abc".

{icdbg.i &T1='"Task-001"'
         &T2='"Start of xxtest.p"'}

lvrRowid = getRecord("pt_mstr").
find pt_mstr no-lock where rowid(pt_mstr) = lvrRowid no-error.

{icdbg.i &T1='"Task-001"'
        &T2='"Start of xxtest.p"'
        &V1=l
        &V2=pt_part
        &V3=pt_abc
        &LEVEL=80
        }
```

2 Compile your source code to the proper QAD Enterprise Applications program structure.

3 Run Debugger using the standard QAD Enterprise Applications system menu.

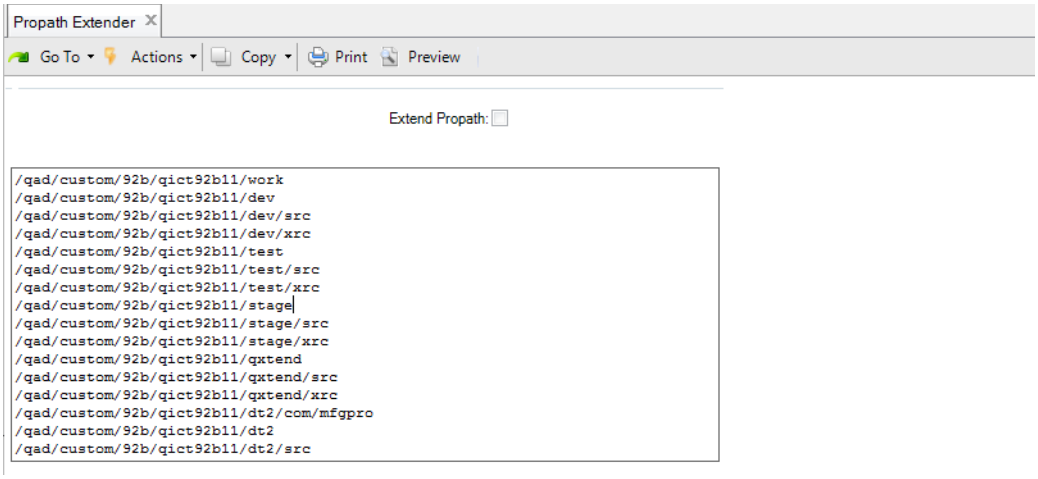
4 Activate the ICT Debugger. Define the debugging level and mask.

5 Run the customized program to watch the debugging messages.

Propath Extender (90.24.12)

You can use Propath Extender to extend the session Propath by adding paths to the `src` and `xrc` directories. The usual session Propath is limited to the installed product paths and does not include paths to the source code, which are necessary for development activities. Beginning with ICT version 4.0, users can also modify each directory manually, adding and deleting particular directories.

Note Propath Extender changes only the session Propath.



Extend Propath. Determines whether the Propath is extended with `src` and `xrc` directories. Optional.

Using the Program

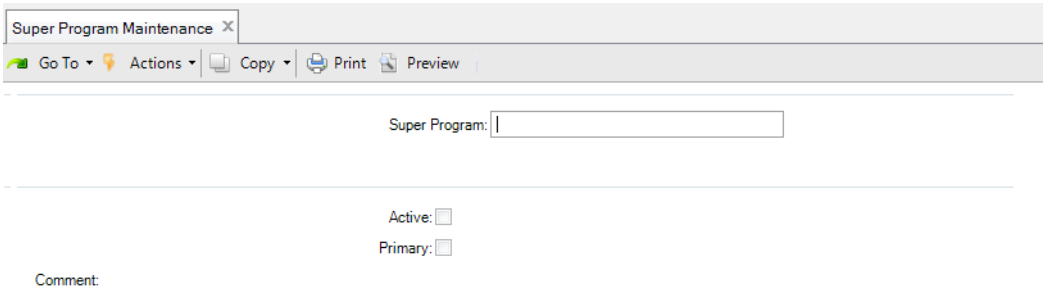
1. Run Propath Extender using the standard QAD Enterprise Applications system menu.
2. To extend the session Propath by adding paths to the `src` and `xrc` directories, set Extend Propath to Yes and confirm.
3. Go to the appropriate line and edit it to modify an existing path.
4. To delete an existing path, go to its line and delete it as you would in a text editor.
5. To add a new path, choose the line where new content should be added and put the cursor at the beginning of this line. Add a new line by pressing Enter and typing a new path to the wanted directory.
6. Save additions/deletions/changes using Next (F1) or discard them with Back (F4).

Note Added and modified entries are not validated.

Super Program Maintenance (90.24.13)

Super Program Maintenance allows the QAD ICT programmers to:

- **Switch on/off QAD ICT processing.** QAD ICT processing is defined in one main `icprcs.p` procedure, which is required to make QAD ICT processing run.
- **Attach prepared programming libraries to QAD ICT processing.** You can attach prepared programming libraries to QAD ICT processing by running them as a super procedure. Then, all internal procedures defined in the attached libraries will be available for QAD ICT customizations.



Super Program. Name of the procedure. Mandatory.

Active. Determines whether the super procedure should be active/inactive. Mandatory.

Primary. The main ICT procedure has to be marked as PRIMARY. In case of internal procedure naming conflicts, procedures defined in the primary procedure will be chosen to be run. Read only.

Comment. Additional information, such as the reason for activation. Optional.

Using the Program

1. Run Super Program Maintenance using the standard QAD Enterprise Applications system menu.

   Note If your super program manipulates database records, remember to release each locked record using the Progress “release” statement:

   ```
   RELEASE record [NO-ERROR]
   ```

2. Enter the name of a procedure or choose an existing one to edit the settings.

   Note The program name field is validated. Only programs with available r-code placed in the proper subdirectories structure `LANG_DIR/TWO-LETTER` directory are permitted.

3. Choose whether the super procedure should be active/inactive.

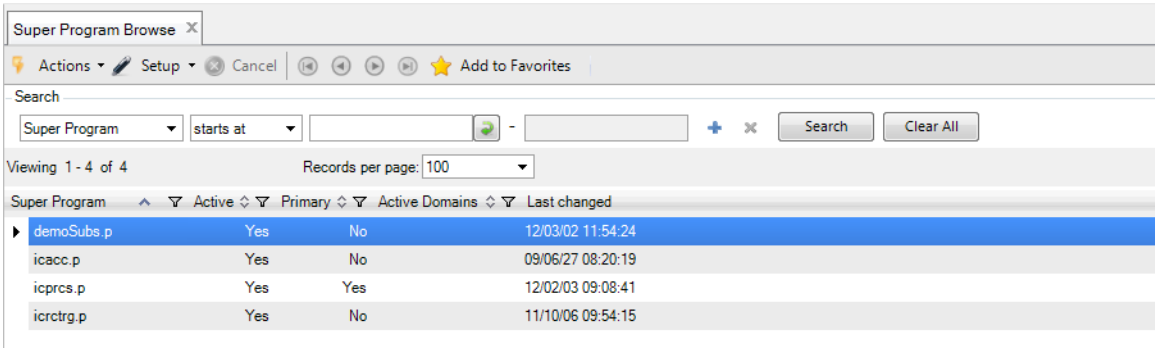
4. Add, edit, or delete the comment.

5. Exit Super Program Maintenance.

6. Restart ICT to load your customization.

Super Program Browse (90.24.14)

Super Program Browse is a standard browse for super programs defined in Super Program Maintenance.

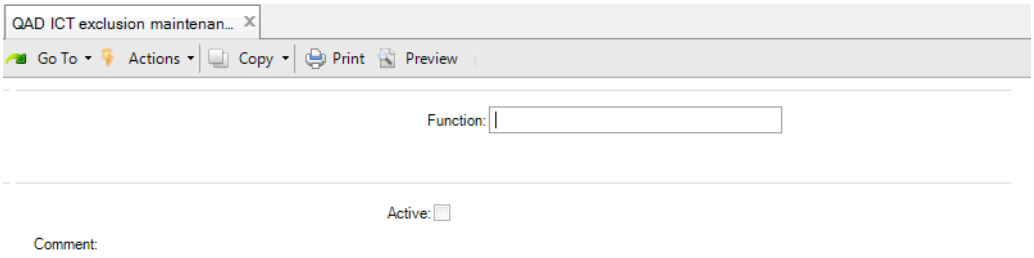


QAD ICT Exclusion Maintenance (90.24.16)

QAD ICT Exclusion Maintenance allows the QAD ICT programmers to switch on/off the ICT find triggers for a selected report or browse.

The purpose of turning off QAD ICT find triggers for selected reports or browses is improved performance. There is no need to run find triggers during execution of a report or browse because it affects performance and there is no functionality behind it.

Note Beginning with ICT version 4.1.20, you can use a wildcard in the function name. For example, if you enter `ppptiq*`, all inquiries from the 1.4 menu (`ppptiq05.p`, `ppptiq04.p`, `ppptiq01.p`, and so on) will be affected.



Function. Report or browse name that you want to add to the list of excluded reports to improve the program's performance. You can enter part of a function name with an asterisk (*) as a wildcard. Mandatory.

Active. Make the setting active/inactive. Mandatory.

Comment. Space for additional information, such as the reason for adding the report or browse to QAD ICT Exclusion Maintenance. Optional.

Using the Program

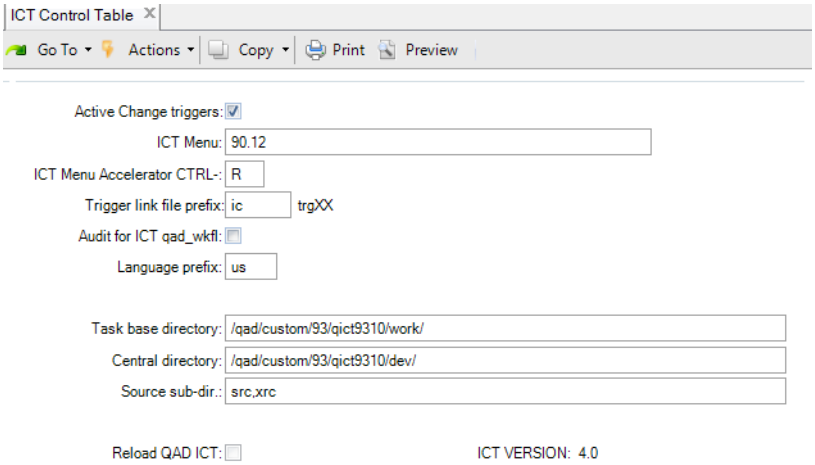
1. Run QAD ICT Exclusion Maintenance using the standard QAD Enterprise Applications system menu.
2. Specify the report or browse name that you want to add to the ICT excluded reports to improve performance.
3. Make the setting active/inactive.
4. Add, edit, or delete the comment.

5 Exit QAD ICT Exclusion Maintenance.

6 Restart ICT processing.

ICT Control Table (90.24.24)

ICT Control Table allows you to change several parameters for the QAD ICT product.



Active Change triggers. Specifies if the system will register all changes done by users. Changes can be exported to the file by using Dump Changed Developers Data (90.24.1). Optional.

ICT Menu. Specify which menu should be used as an ICT Menu. Functions that are available under that menu will be displayed when the user presses the menu accelerator shortcut (available only under character mode). By default, it is 90.12 (QAD ICT Developers Menu). Optional.

ICT Menu Accelerator. The system displays the shortcut that will be used to display the ICT Menu. See “Changing the ICT Development Menu Accelerator” on page 17 for details.

Trigger link file prefix. Specification of the prefix for the trigger files automatically generated by QAD ICT. Valid values are ic, icc, icq, or blank. Optional.

Audit for ICT qad_wkfl. Indicates if QADICT settings (changes in qad_wkfl) should be audited. To activate audit for qad_wkfl table, you must use Simplified Audit. If Audit for ICT qad_wkfl is set to No, ICT settings are not audited despite the fact that qad_wkfl audit is enabled. Optional.

Language prefix. Prefix defined for default language settings. By default it is US. Mandatory.

Task base directory. Determines where task subdirectories are created. The task directory name obtained by concatenating the base directory and the task ID. The new or modified custom procedures will be kept in the task directory and compiled there as long as the task is open. Once the task is closed, all such files will be moved automatically to the central directory (r-code to global_user_lang_dir + first two letters subdirectory, source code files to the first source subdirectory) and thus become visible to all users. Mandatory.

Central directory. Used to indicate the directory where the customer customizations are stored (subscriber, UI triggered, validation, other non ICT-related procedures). This directory will usually be the first one in the Propath. Mandatory.

Source sub-directories. Definition of a comma-separated list of source subdirectories used to extend the Propath. The first subdirectory will be the target subdirectory for customized source files in the central directory. Mandatory.

Reload QAD ICT. Enables restarting the ICT processing. Optional. Not available in .NET UI. To restart ICT processing in .NET UI, you must restart the Connection Manager.

ICT Version. Displays the installed QAD ICT version. Read only.

Chapter 4

QAD ICT Restrictions and Example Customizations

This chapter describes some restrictions associated with QAD ITC processing and provides real-world examples of how to create customizations.

QAD ICT Restrictions

This section describes restrictions—actions that cannot be set up in QAD ICT. Please note that these restrictions result from Progress limitations—not QAD ICT itself.

UI Triggers and Functions Restrictions

The following table describes which UI triggers and functions are supported (Yes) or unsupported (No) in .NET UI and CIM processing.

Trigger	Supported in .NET UI	Supported in CIM
On Entry on Field	No	No
On Leave on Field	No	No
On Go on Field	No	No
On Entry on Frame	Yes	No
On Go	Yes	No
GetFieldValue()	Yes	No
SetFieldValue()	Yes	No
GlobalValue()	Yes	No
Validation	Yes	No
Default Value	Yes	No
New fields on standard frame	Yes	No

Menu Accelerator Restrictions

Please note that the menu accelerator cannot be used when the cursor is in a field that is updated in the message area at the bottom of the screen.

CIM Processing Restrictions

Custom Frames

Because of limitations of CIM processing, custom frames added with UI triggers are not accessible during CIM load. To enable a custom frame for CIM processing, add it using Program Hook.

Custom Fields in Standard Frames

Custom fields added to standard frames are not accessible during CIM load, either. To enable additional fields during CIM load, follow these steps:

1. Create a custom frame in Pop-up Frame Maintenance and set Batch Only to Yes. This means that the defined frame will be visible only during CIM load.
2. Use Frame & Field Properties Maintenance to add custom fields to the frame and define the right order for processing them.
3. Use Program Hook Maintenance to add your pop-up frame in the proper position.

Note Remember to create the batch file with custom fields placed in the right position for the added pop-up frame.

Validation

To add validation to custom fields, you must add a validation for both fields added to the standard frame and the CIM pop-up frame. If validation is added only for the added fields in the standard frame, validation does not apply during CIM load. The converse is also true: validation added to a field in a CIM frame will not guarantee validation in manual mode, where the user sees the added fields in a standard frame.

SOX Compliance

To ensure SOX compliance, ICT allows you to define which data should be dumped/loaded in the following menu options:

- Dump Changed Developers Data (90.24.1)
- Load Developers Data (90.24.2)
- Report/Export Tasks (90.23.18)

To restrict data dump/load using ICT to particular tables, the system administrator should define Generalized Codes.

An example configuration in Generalized Codes Maintenance:

```
mgcodemt.p 3+          36.2.13 Generalized Codes Maintenance          09/20/13
+------------------------------ Generalized Codes -------------------------------+
|              Field Name: icdatdmp_table                                        |
|                   Value: qad_wkfl                                              |
|                                                                                |
|                Comments:                                                       |
|                                                                                |
|                   Group: APP                                                   |
+--------------------------------------------------------------------------------+
mgcodemt.p 3+          36.2.13 Generalized Codes Maintenance          09/20/13
+------------------------------ Generalized Codes -------------------------------+
|              Field Name: icdatdmp_table                                        |
|                   Value: lbl_mstr                                              |
|                                                                                |
|                Comments:                                                       |
|                                                                                |
|                   Group: APP                                                   |
+--------------------------------------------------------------------------------+
```

The above definitions allow you to dump/load data only from two tables: qad_wkfl and lbl_mstr. If no definition for the icdatdmp_table field is available, all tables that are under ICT control will be dumped/loaded.

If there is at least one definition for icdatdmp_table in Generalized Codes Maintenance, the warning “Tables to dump/load are restricted in GCM (fieldname icdatdmp_table)” will be displayed in Dump Changed Developers Data, Load Developers Data, and Report/Export Tasks.

Example Customization Scenarios

This section includes examples of how to make the following customizations using QAD ICT:

- Adding a Custom Frame with Fields from a Shadow Table
- Adding Custom Fields to a Standard Frame
- Adding Simplified Audit
- Adding a Custom Procedure on Assign on the Specified Field

All customizations must be done using an ICT task. See “QAD ICT Tasks” on page 52 for details.

On the diagrams in this section, you can find the Restart the Connection Manager block twice. The first instance, before closing the ICT task, loads the customization that was created in the current task. It is visible only for the user who has selected the task. The second instance, after closing the task, loads the customization for all users.

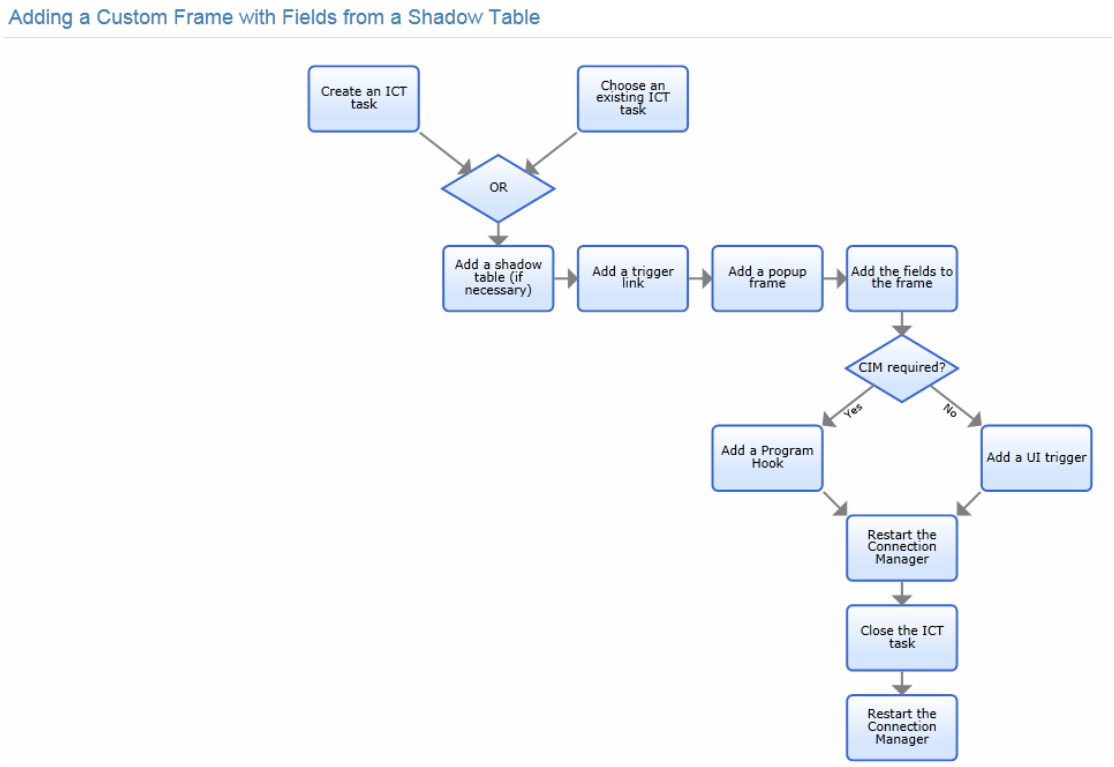
Adding a Custom Frame with Fields from a Shadow Table

This example shows, step by step, how to add a new shadow table for po_mstr and add a custom frame in Purchase Order Maintenance (5.7). Use the same task flow to add a shadow table for another table and a custom frame to other maintenance programs. This example is divided into three main tasks:

1 Add a shadow table.

2 Add a custom frame.

3 Place the frame in the proper position.



Add a Shadow Table

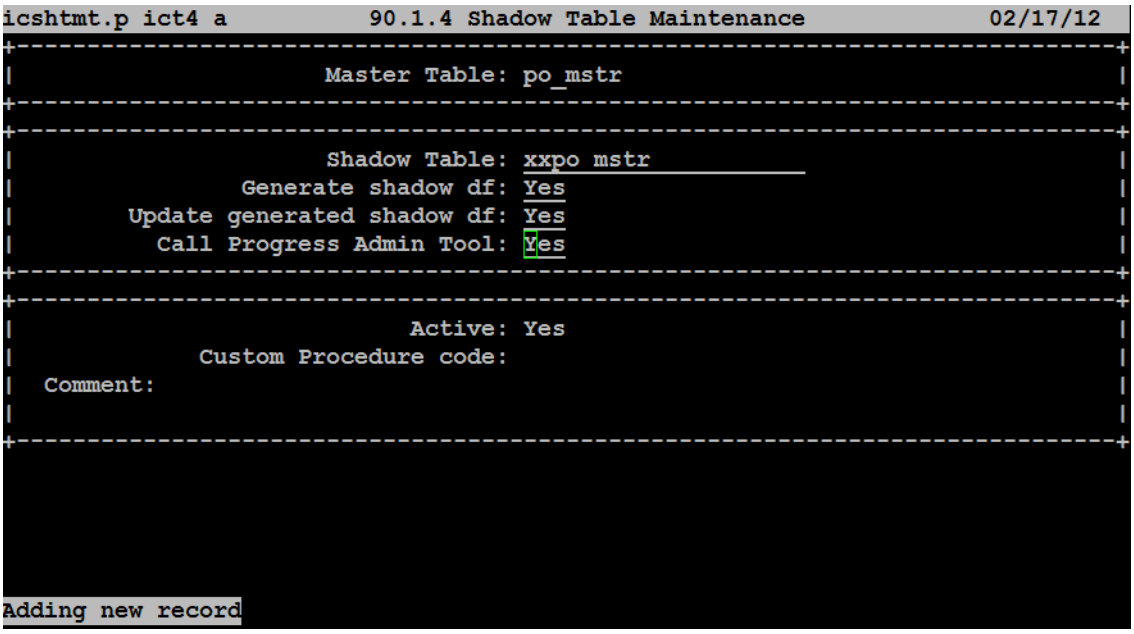
This example is performed in character mode, because some of Shadow Table Maintenance functions are not available in .NET UI.

1 Open Shadow Table Maintenance (90.1.4).

2 Enter the name of a master table. For this example, it will be po_mstr.

3 Enter the name of a shadow table. For this example, it will be xxpo_mstr.

4 Leave Generate shadow df, Update generated shadow df, and Call Progress Admin Tool flags set to Yes.



5 Confirm the inserted data with the F1 key. In response, QAD ICT should display: “Dump of definitions completed. Shadow table .df file has been generated:xxpo_mstr.df.”

6 Press the space bar to continue. Progress Editor will be started with the generated `.df` file.

```
File Edit Search Buffer Compile Tools Help QVersion
------------------------------------------------------------------------------
ADD TABLE "xxpo_mstr"
   AREA "Schema Area"
  DESCRIPTION "Purchase Order Master"
  DUMP-NAME "xxpo_mstr"

ADD FIELD "xxpo_nbr" OF "xxpo_mstr" AS character
  DESCRIPTION "The unique code for a purchase order."
  FORMAT "x(8)"
  INITIAL ""
  LABEL "Purchase Order"
  MAX-WIDTH 80
  COLUMN-LABEL "Order"

ADD FIELD "xxpo_domain" OF "xxpo_mstr" AS character
  DESCRIPTION "The business domain for this data."
  FORMAT "x(8)"
  INITIAL ""
  LABEL "Domain"
- File: /qad/custom/92b/qict92b11/work/xxpo_mstr.df ----------------------------
```

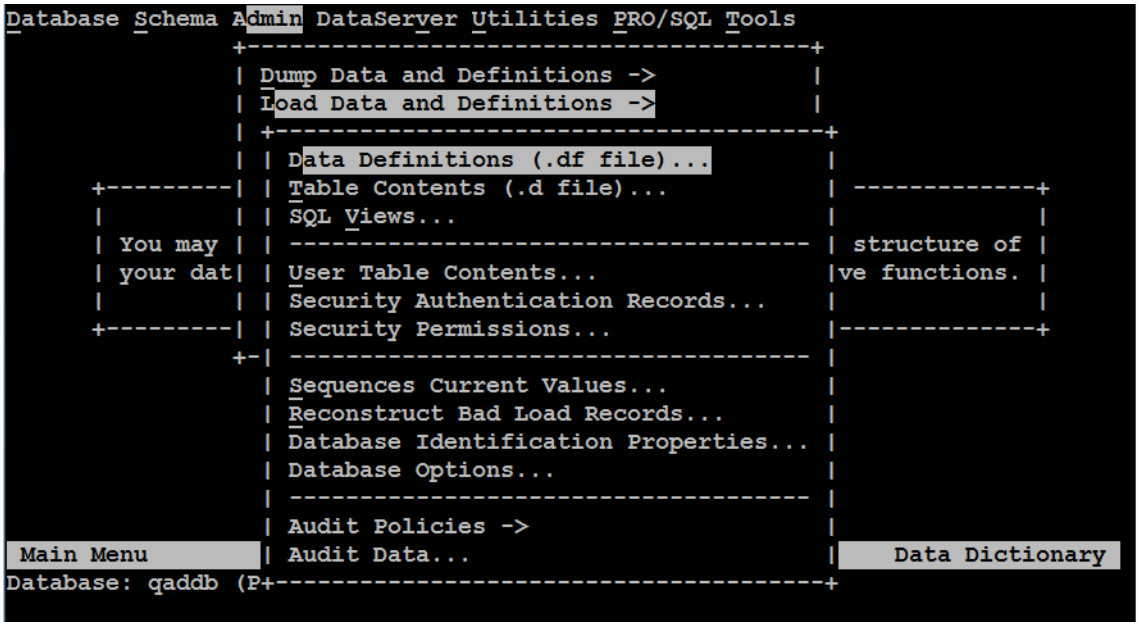
7 Modify the `.df` file, save your changes, and close Progress Editor. Example code for a new field is shown below.

```
ADD FIELD "xxpo_myField" OF "xxpo_mstr" AS character
  DESCRIPTION "Custom field added for demo purposes."
  FORMAT "x(8)"
  INITIAL ""
  LABEL "Demo"
  MAX-WIDTH 8
```

You can add a new field later, in Data Dictionary.

Note Before you press Go and call the Progress Admin Tool, please make sure that no users are connected to the QAD Enterprise Applications system. Changing the database schema with users connected to it is impossible.

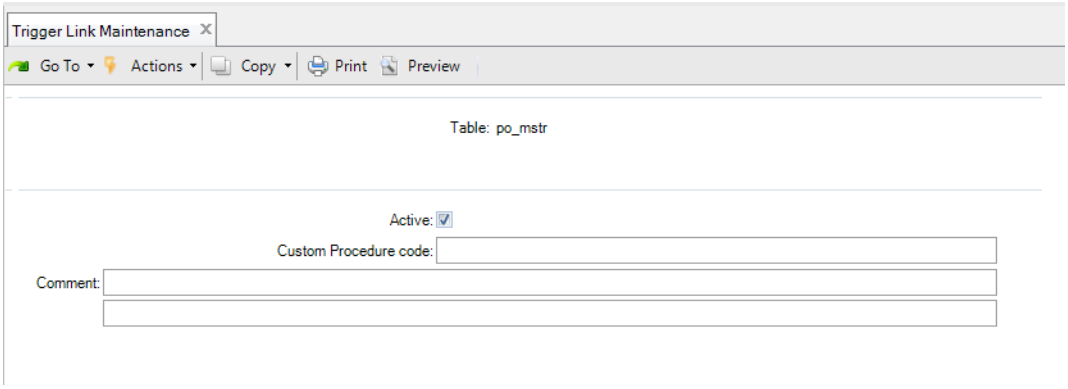
8 Use option Admin -> Load Data and Definitions -> Data Definitions (.df file) to load the generated `.df` file.



Choose an input file and define a code page if necessary. “Load completed” information should be displayed. Exit the Data Dictionary.

9 Leave the Active field set to Yes and exit Shadow Table Maintenance. Answer Yes to the “Do you want to rebuild Link&Consistency triggers?” prompt.

10 Add the trigger link on the po_mstr table. See “Trigger Link Maintenance (90.1.1)” on page 17. Answer Yes to the “Do you want to rebuild Link&Consistency triggers?” prompt.

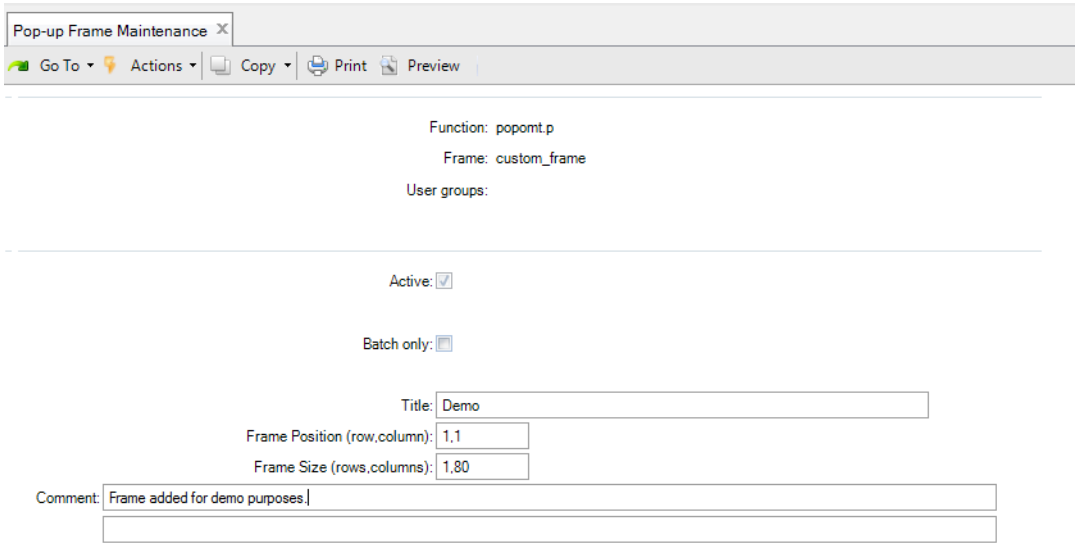


Note It is very important to ensure that the correct index is defined for the shadow table. Otherwise, the shadow table will not be linked with the master one.

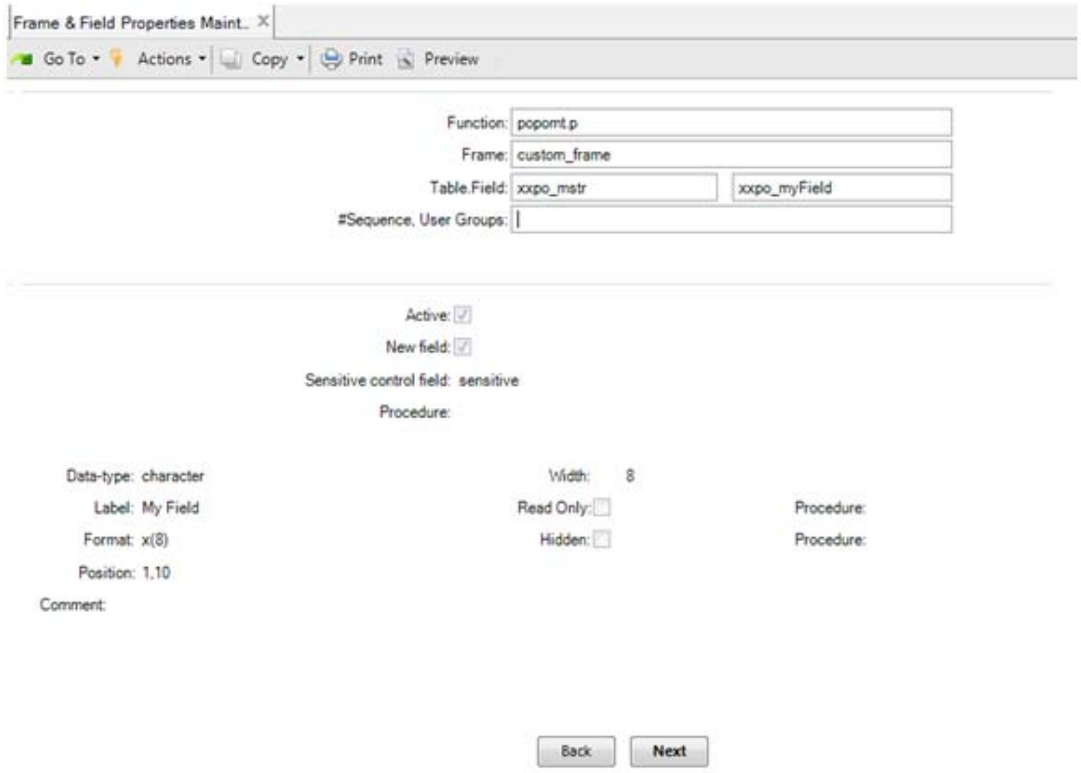
Add a Custom Frame

1 Open Pop-up Frame Maintenance (90.3.1) and add a custom frame. See “Pop-up Frame Maintenance (90.3.1)” on page 27 for information.

Example data is shown below.

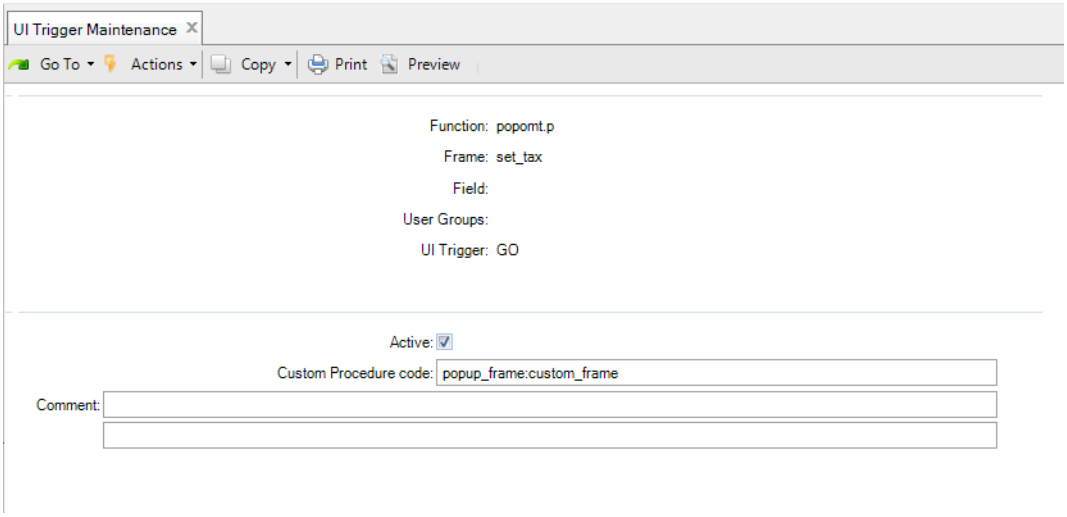


2 Use Frame & Field Properties Maintenance to add a custom field to the frame.



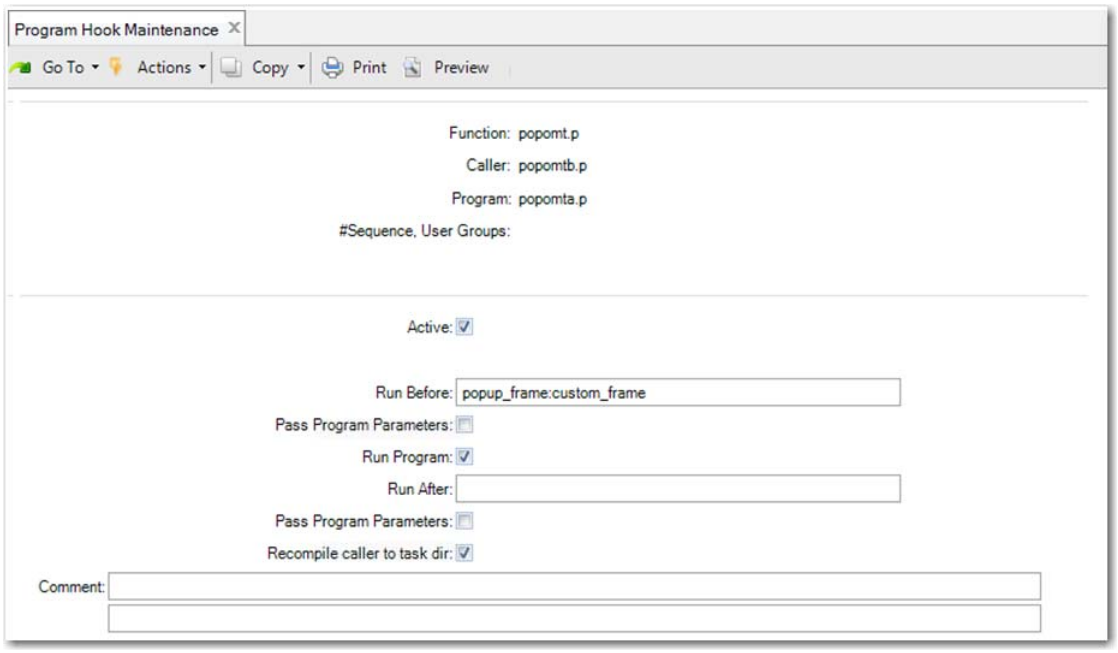
Place the Frame in the Proper Position

1. Decide if compliance with CIM is required. If it is, go to step 5. Otherwise, continue with the next step.
2. If compliance with CIM is not required, use a UI trigger to place your frame in the proper position. Open UI Trigger Maintenance and add a trigger (note the UI trigger restrictions explained on page 82). In the following example, the pop-up frame will be displayed after the tax details frame.



3. Exit UI Trigger Maintenance.
4. Restart the ICT processing.
5. If compliance with CIM is required, use a program hook to put your frame in the proper position. Open Program Hook Maintenance and add a program hook based on the example under “Program Hook Maintenance (90.5.13)” on page 46. Sample data are shown below.

   Note If the compilation was not performed by ICT, remember to recompile the Caller procedure (in this example, `popomtb.p`). When automatic is compilation made by ICT, the r-code is compiled to the current task directory. It is moved to the central directory when the task is closed.

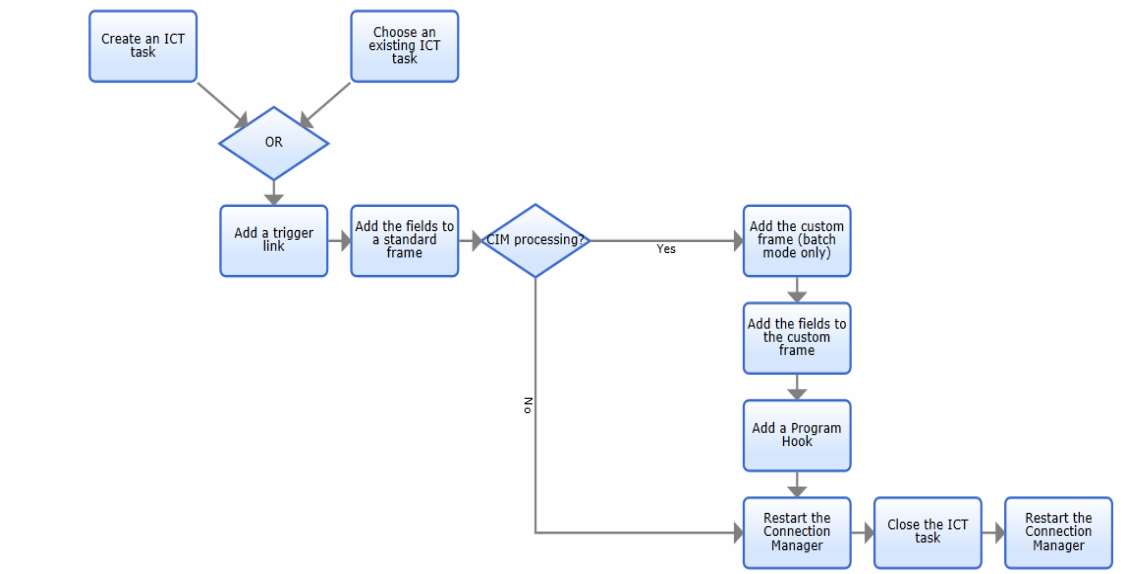


6 Restart the ICT processing.

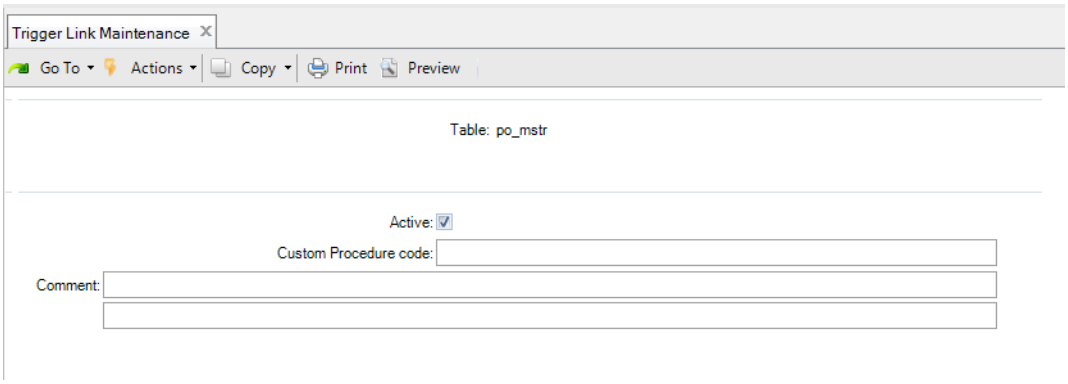
Adding Custom Fields to a Standard Frame

This example shows how to, step by step, add custom fields to a standard QAD Enterprise Applications frame. An additional (optional) step describes how to ensure compliance with CIM and can be omitted if CIM compliance is not necessary. In this example, the new field (po__chr01) will be added to Purchase Order Maintenance (5.7).

Adding Custom Fields to a Standard Frame

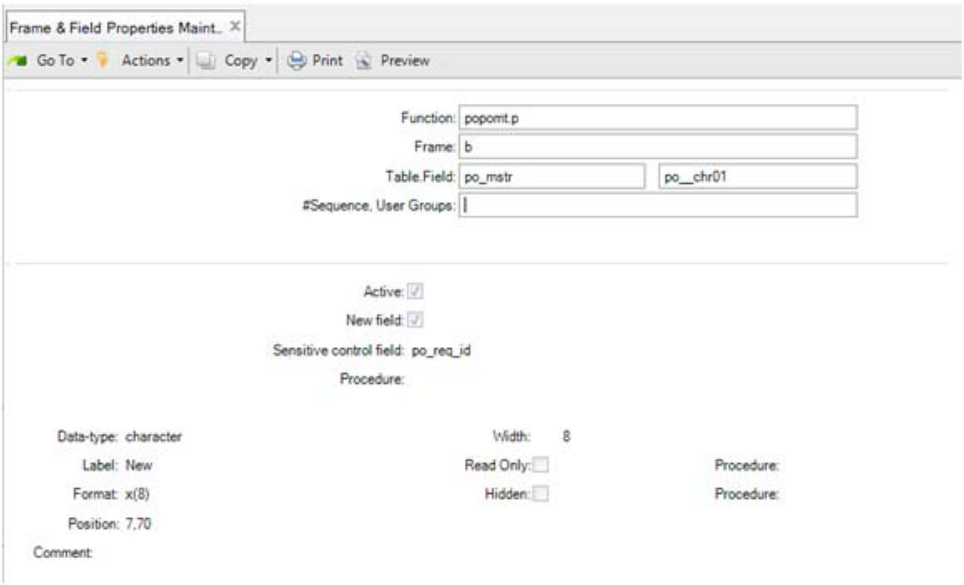


1 Add a trigger link for table po_mstr to enable linking the current displayed record with the added field to ensure automatic updates to the database.



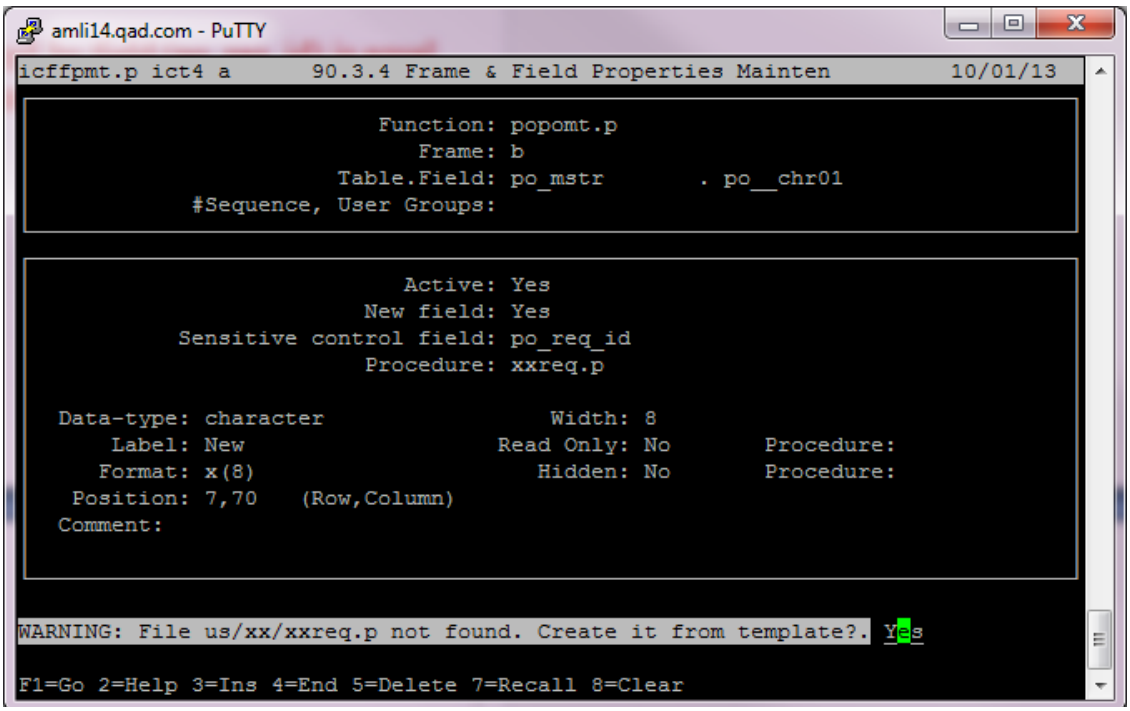
Note Sometimes it is also necessary to add the ON CREATE trigger on the table. In some standard maintenance programs, when new records are created, the ON FIND trigger is not fired. This is why sometimes the added field is not linked with the database field when a new record is created.

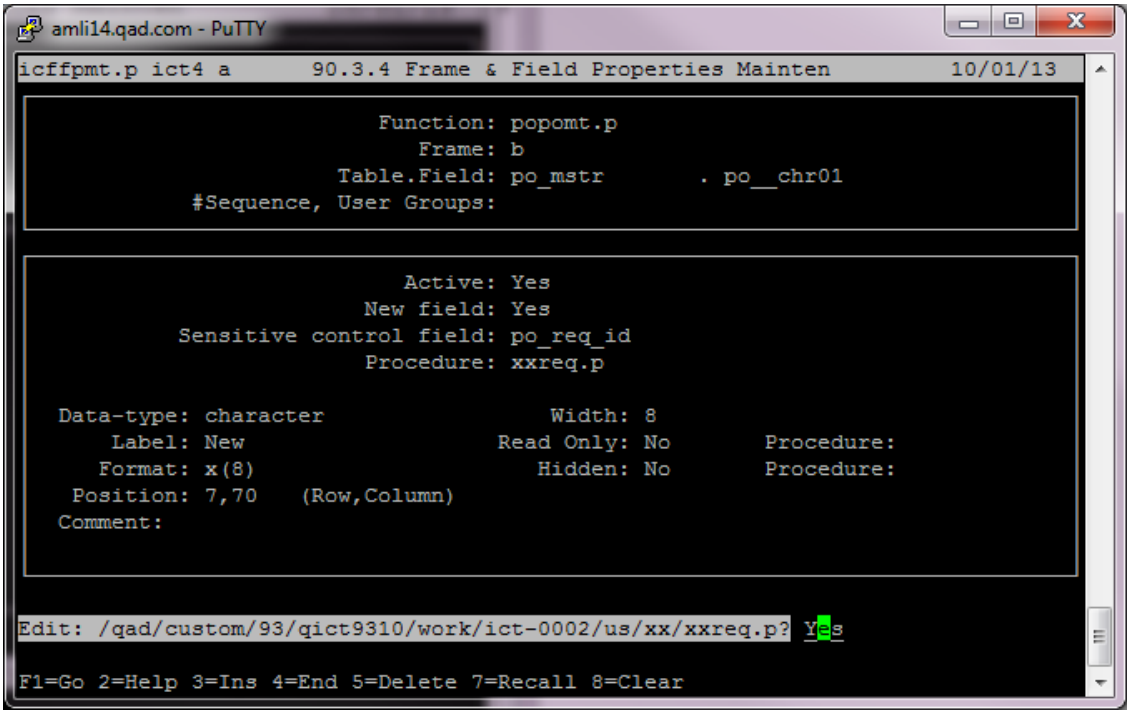
2 Open Frame & Field Properties Maintenance and fill in the fields as shown in the following example.

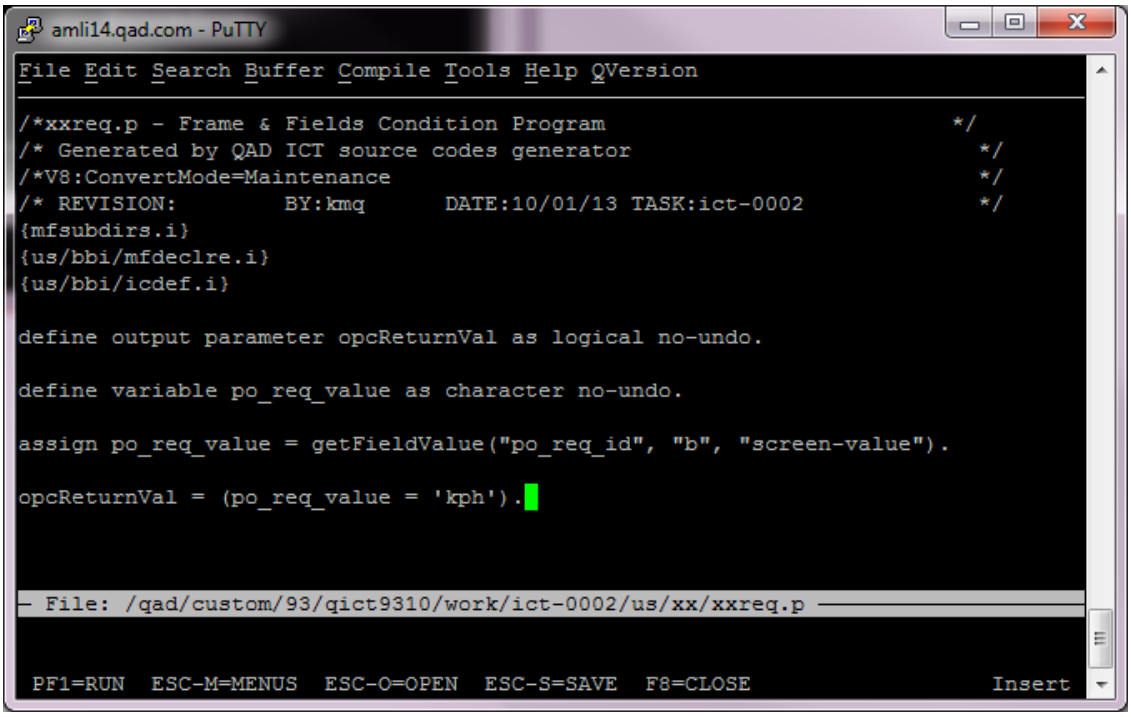


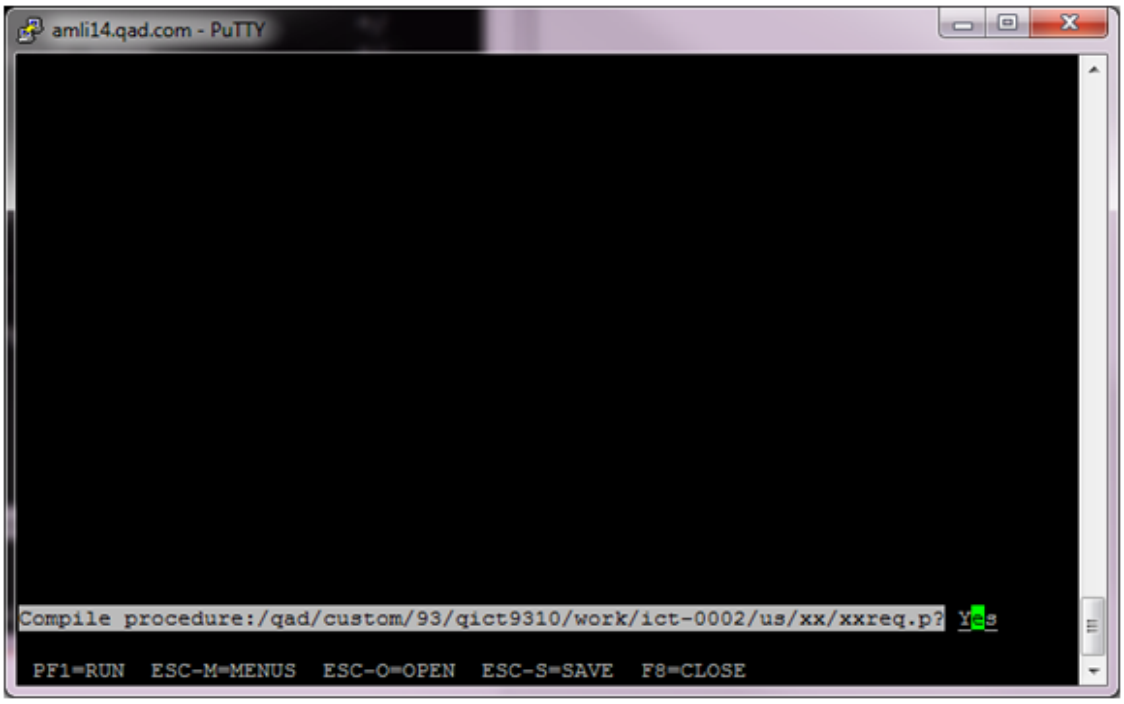
The new field has just been added to the standard frame.

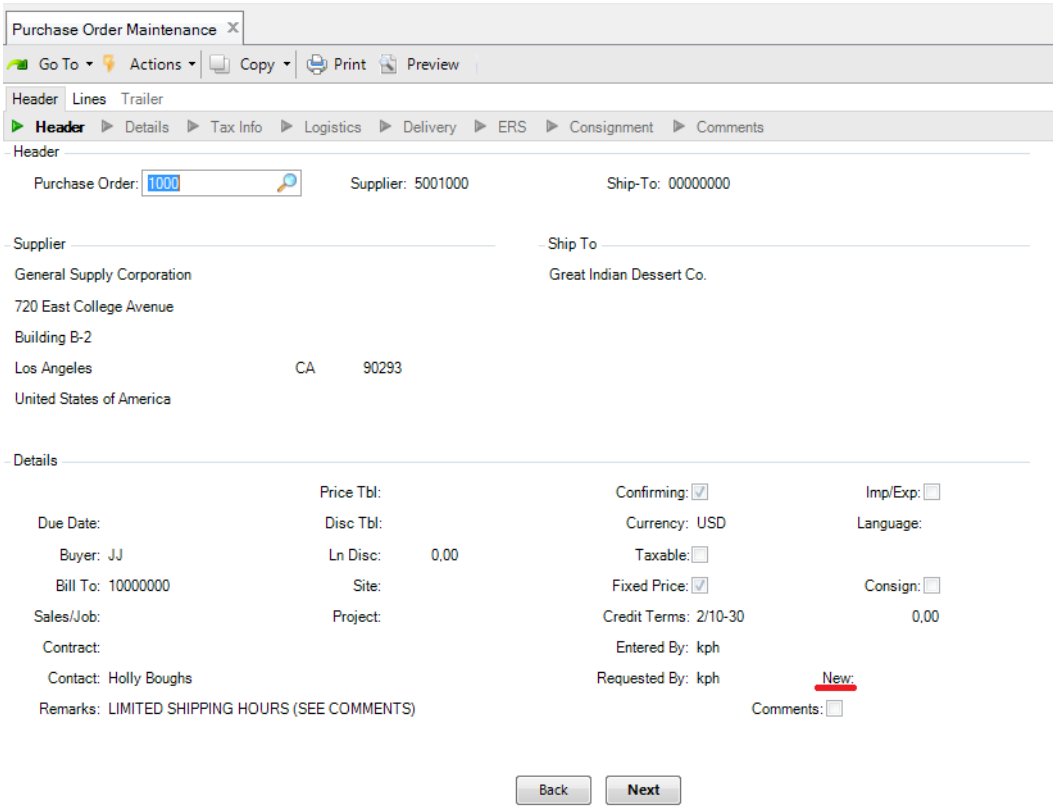
Assume that the field should be visible only if the Requested By field (po_req_id) is equal to “kph”. You must define a procedure with that condition (using the character UI).




amli14.qad.com - PuTTY
icffpmt.p ict4 a 90.3.4 Frame & Field Properties Mainten 10/01/13
Function: popomt.p
Frame: b
Table.Field: po_mstr . po__chr01
#Sequence, User Groups:
Active: Yes
New field: Yes
Sensitive control field: po_req_id
Procedure: xxreq.p
Data-type: character Width: 8
Label: New Read Only: No Procedure:
Format: x(8) Hidden: No Procedure:
Position: 7,70 (Row,Column)
Comment:
Edit: /qad/custom/93/qict9310/work/ict-0002/us/xx/xxreq.p? Yes
F1=Go 2=Help 3=Ins 4=End 5=Delete 7=Recall 8=Clear

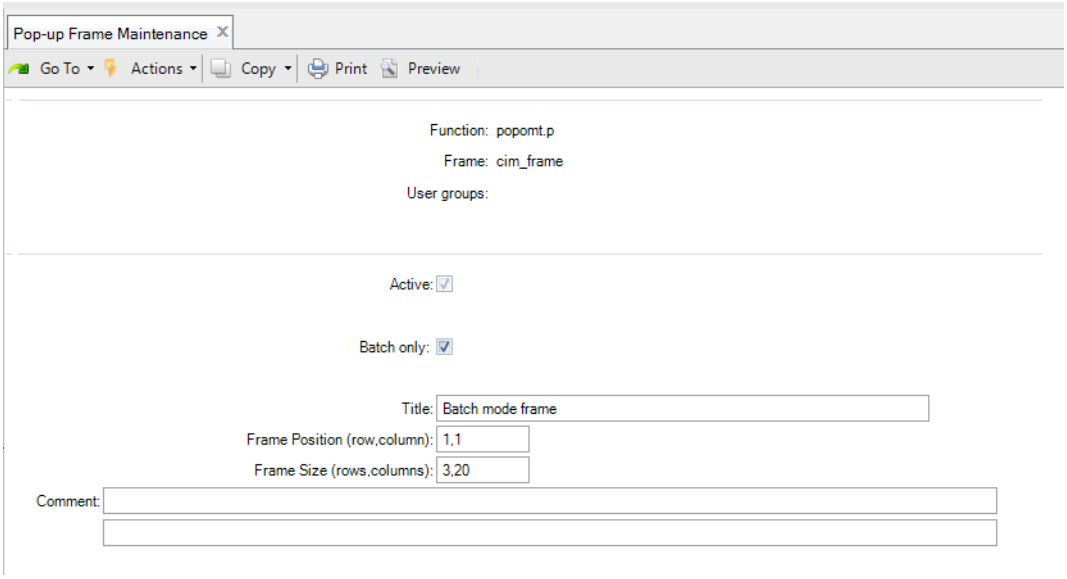

amli14.qad.com - PuTTY
File Edit Search Buffer Compile Tools Help QVersion
/*xxreq.p - Frame & Fields Condition Program */
/* Generated by QAD ICT source codes generator */
/*V8:ConvertMode=Maintenance */
/* REVISION: BY:kmq DATE:10/01/13 TASK:ict-0002 */
{mfsubdirs.i}
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}
define output parameter opcReturnVal as logical no-undo.
define variable po_req_value as character no-undo.
assign po_req_value = getFieldValue("po_req_id", "b", "screen-value").
opcReturnVal = (po_req_value = 'kph').
File: /qad/custom/93/qict9310/work/ict-0002/us/xx/xxreq.p
PF1=RUN ESC-M=MENUS ESC-O=OPEN ESC-S=SAVE F8=CLOSE Insert


amli14.qad.com - PuTTY
Compile procedure:/qad/custom/93/qict9310/work/ict-0002/us/xx/xxreq.p? Yes
PF1=RUN ESC-M=MENUS ESC-O=OPEN ESC-S=SAVE F8=CLOSE

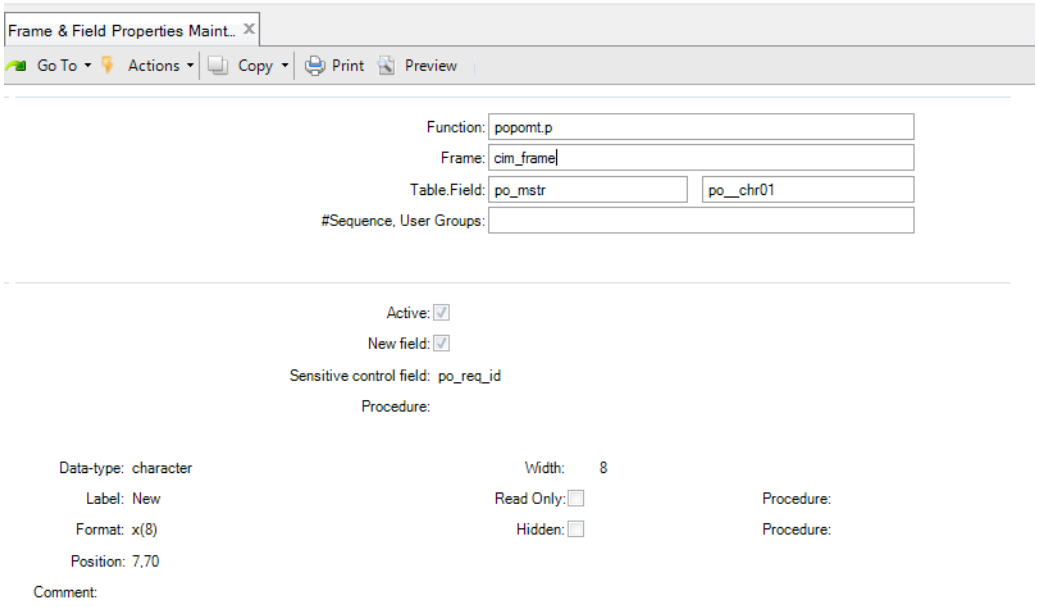

Purchase Order Maintenance
Go To
Actions
Copy
Print
Preview
Header
Lines
Trailer
Header
Details
Tax Info
Logistics
Delivery
ERS
Consignment
Comments
Header
Purchase Order: 1000
Supplier: 5001000
Ship-To: 00000000
Supplier
General Supply Corporation
720 East College Avenue
Building B-2
Los Angeles CA 90293
United States of America
Ship To
Great Indian Dessert Co.
Details
Price Tbl:
Disc Tbl:
Ln Disc: 0,00
Site:
Project:
Due Date:
Buyer: JJ
Bill To: 10000000
Sales/Job:
Contract:
Contact: Holly Boughs
Remarks: LIMITED SHIPPING HOURS (SEE COMMENTS)
Confirming:
Currency: USD
Taxable:
Fixed Price:
Credit Terms: 2/10-30
Entered By: kph
Requested By: kph
Imp/Exp:
Language:
Consign:
0,00
New:
Comments:
Back
Next

To ensure compliance with CIM requirements, perform the following steps. (If CIM compliance is not necessary the customization is finished.)

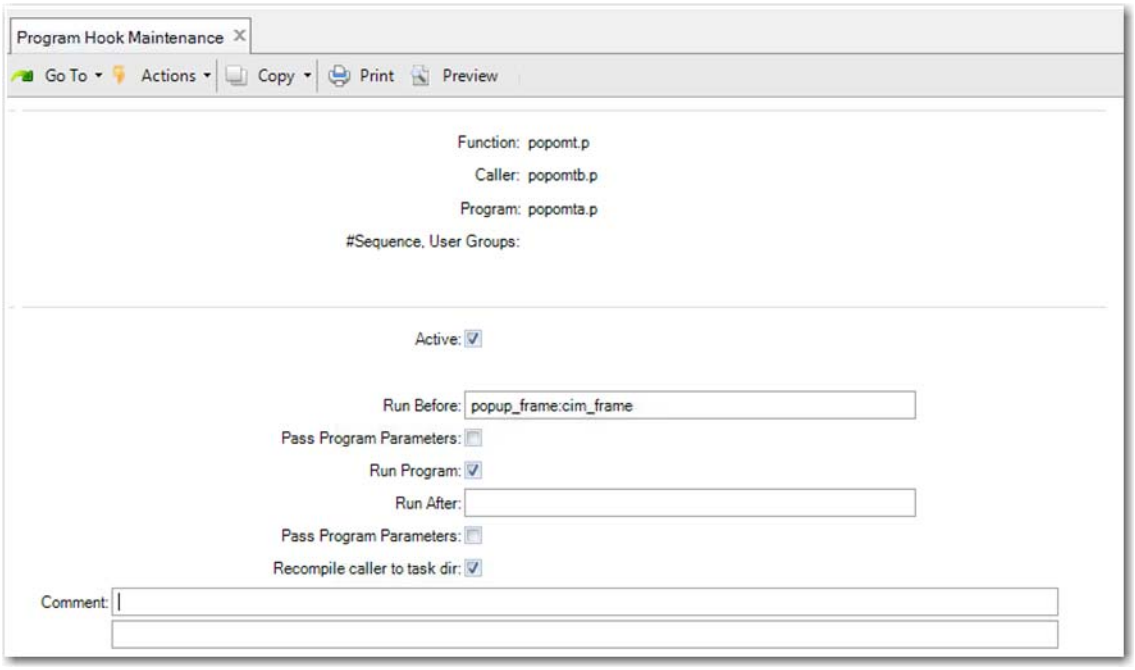
3 If CIM compliance is required, add a custom frame using Pop-up Frame Maintenance (90.3.1). Batch Only should be set to Yes.



4 Add the field to the custom frame.



5 Use a program hook to put your frame in the proper position. Open Program Hook Maintenance and add a program hook based on the example under “Program Hook Maintenance (90.5.13)” on page 46. Sample data are shown below.

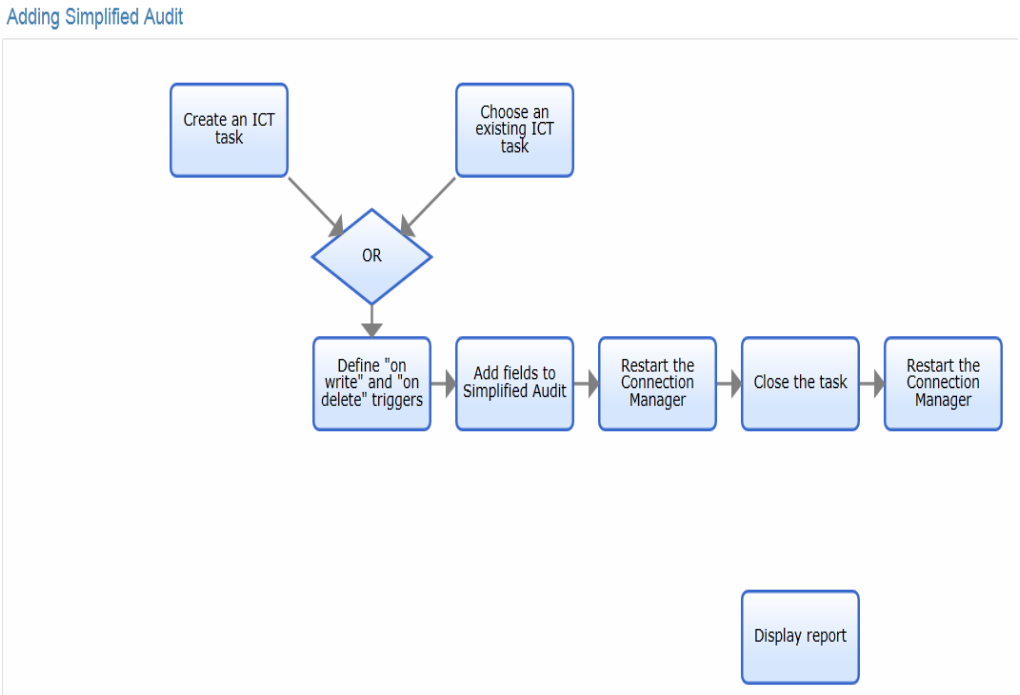


Note Remember to recompile the Caller procedure (in this example, `popomtb.p`).

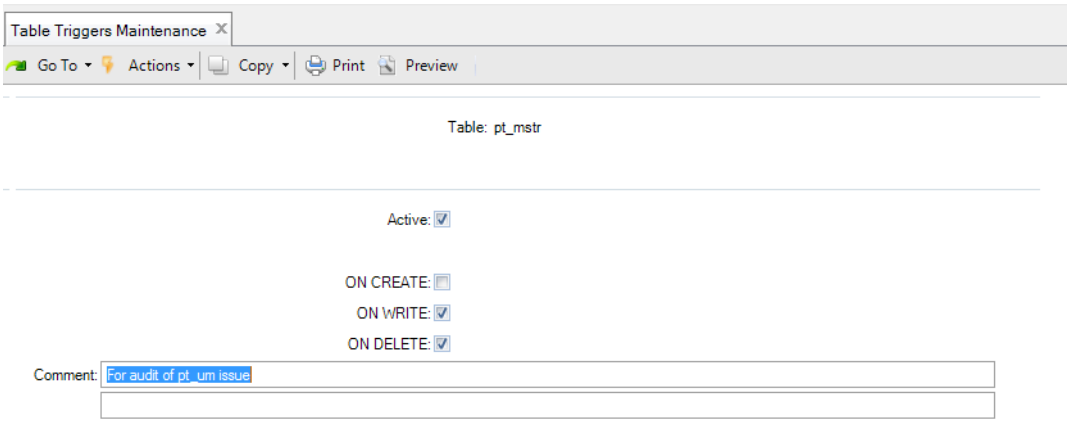
Adding Simplified Audit

This example shows, step by step, how to add simplified audit for the pt_um field from the pt_mstr table.

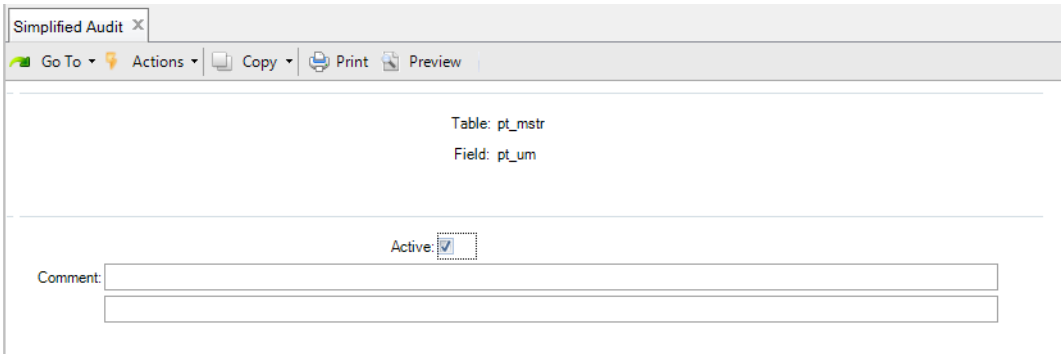
Note This field and table is only an example. The steps described work for all fields and tables.



1 Open Table Triggers Maintenance (90.1.13). Activate ON WRITE and ON DELETE triggers for table pt_mstr. Answer Yes to the “Do you want to rebuild Link&Consistency triggers?” prompt. See “Table Triggers Maintenance (90.1.13)” on page 21 for details.



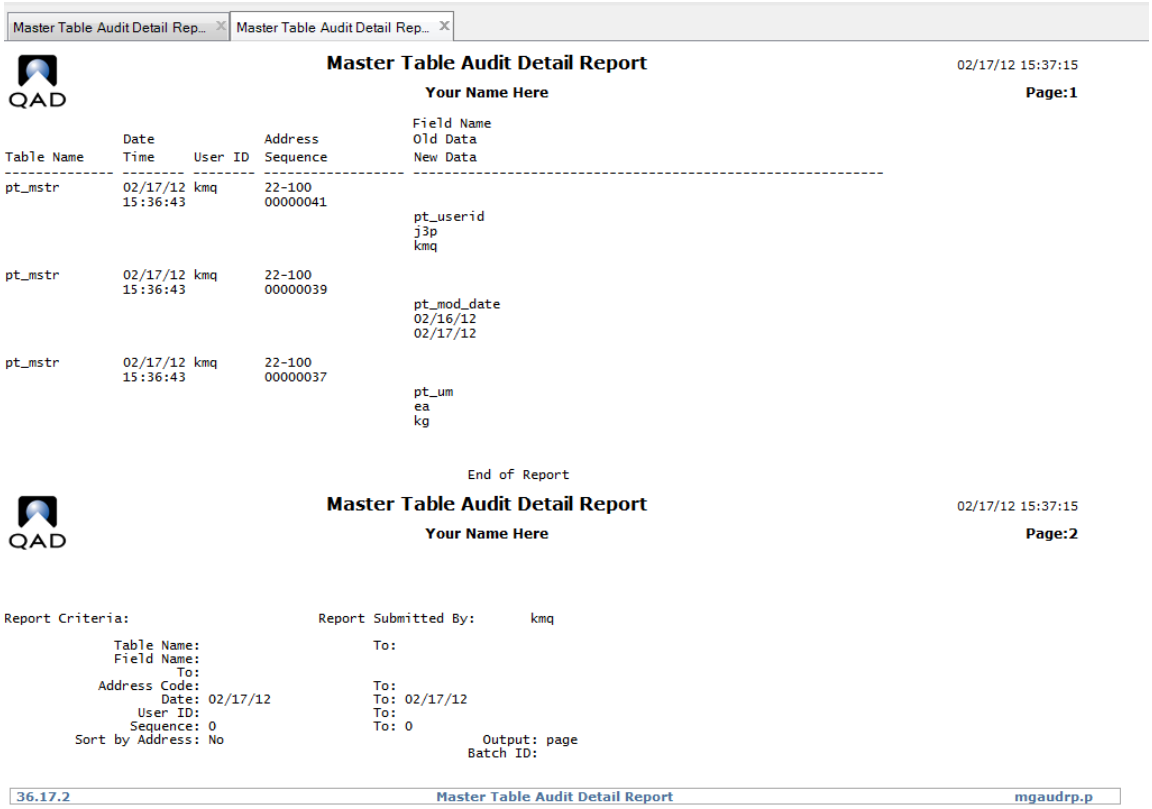
2 Open Simplified Audit (90.1.19). Enter the name of the table and field. For details, see “Simplified Audit (90.1.19)” on page 24.



3 Restart the ICT processing.

4 Make some changes that affect pt_um. Open Master Table Audit Detail Report (36.17.2) and list the changes.

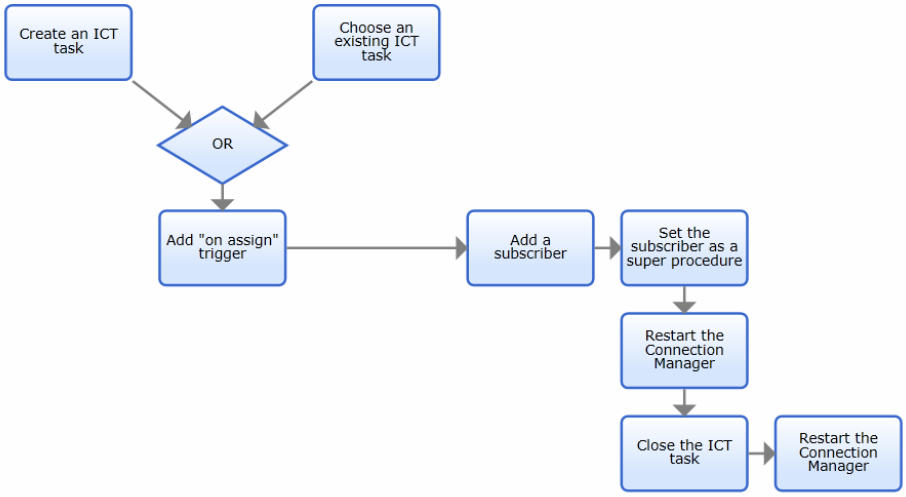
Note In some QAD Enterprise Applications versions, Master Table Audit Detail Report is not available. In this case, you must prepare a browse for the aud_det table.



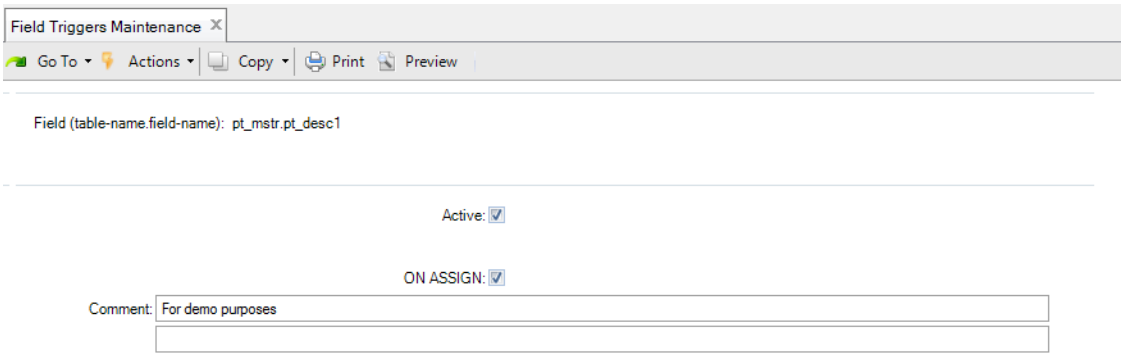
Adding a Custom Procedure on Assign on the Specified Field

The example shows how to, step by step, add a custom procedure on an ON ASSIGN trigger on the specified field. In this example, we will track changes on pt_mstr.pt_desc1. Of course, the same process can be used to track changes on any table/field.

Adding a Custom Procedure on Assign on the Specified Field



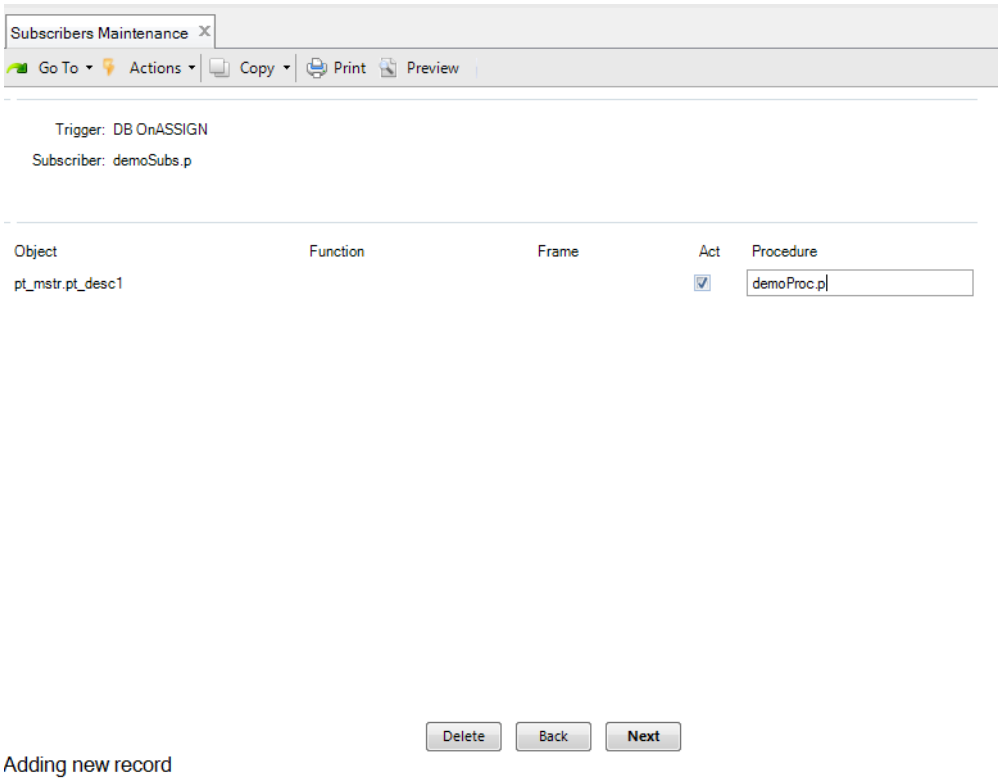
1 Add the proper ON ASSIGN trigger in Field Triggers Maintenance. See the configuration settings below:



2 Rebuild Link & Consistency triggers.

3 Add a subscriber to the ON ASSIGN event using Subscribers Maintenance (90.1.22).

4 Enter the name of the object you want to track changes in. Define the procedure and the frame if you want. In this example, the procedure and frame are blank values, so the added procedure will be fired on any ON ASSIGN event in pt_mstr.pt_desc1.



Adding new record

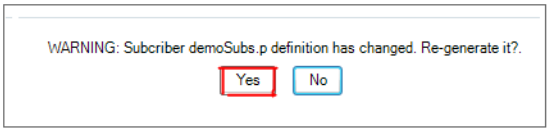
5 Allow ICT to create the procedure from the template. If you use character UI, the procedure will be edited and can be changed and compiled. Otherwise, you must edit it manually.

The procedure generated by ICT will have the following structure:

```
/*************************************************************************/
/* onAssign procedure                                                    */
/*    pt_mstr - pt_mstr record                 */
/* oldpt_desc1 - old pt_desc1 value       */
/* entry(num-entries(ipcParam),ipcParam) - handler for the buffer */
/*************************************************************************/
{mfdeclre.i}
{icdef.i}
define input parameter ipcParam as character no-undo.
define variable oldpt_desc1 as character no-undo.
define variable i as int.

{icdbg.i &T1="'onAssign custom program'"
         &V1=this-procedure:name
         &V2=ipcParam
         &LEVEL=80}

do i = 1 to num-entries(ipcParam):
     if i > 1 and i < num-entries(ipcParam)
     then do:
          if oldpt_desc1= " " then
               oldpt_desc1 = entry(i,ipcParam).
          else
               oldpt_desc1 = oldpt_desc1 + "," + entry(i,ipcParam).
     end.
end.

find first pt_mstr no-lock where
     rowid(pt_mstr) = to-rowid(entry(1,ipcParam)) no-error.

{pxmsg.i &MSGTEXT=" 'demoProc '" &ERRORLEVEL=1} /*custom message*/
```

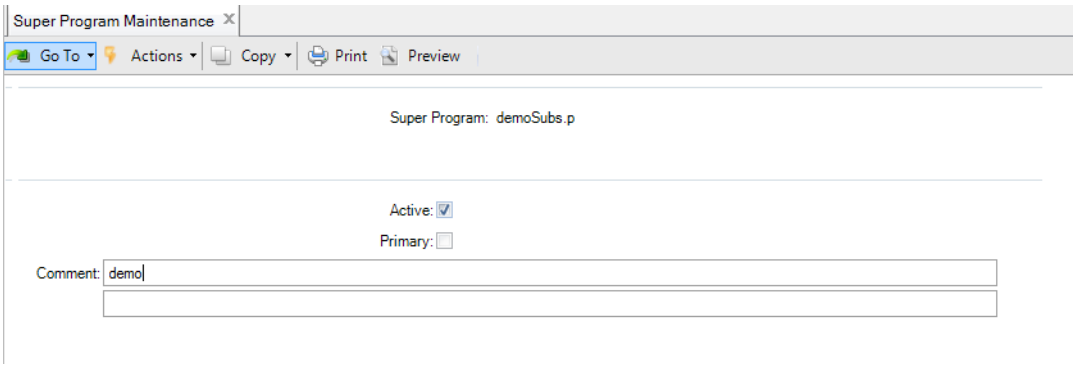
Note Bold text is custom code added for demo purposes.

6 Allow ICT to regenerate the subscriber file.



Note .NET UI does not let you edit and compile created files. Remember to compile the custom procedure and subscriber files before you perform the next step.

7 Add the subscriber as a super procedure in Super Program Maintenance (90.24.13).



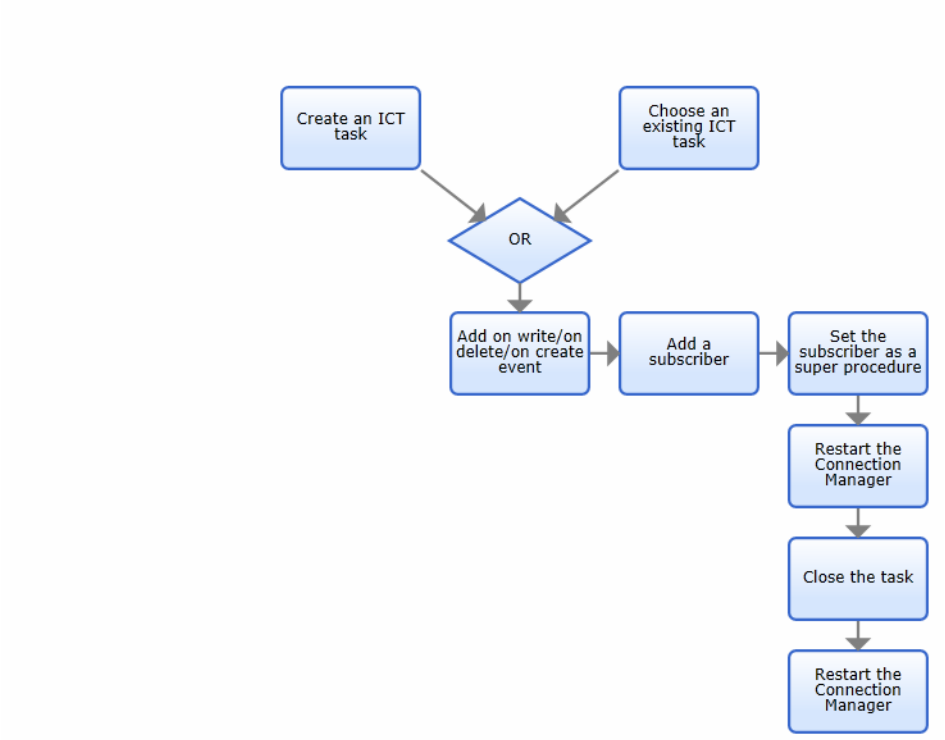
8 Restart the Connection Manager and confirm that your customization works correctly. Change the pt_mstr.pt_desc1 field in any maintenance program and check whether the message appears.

demoProc

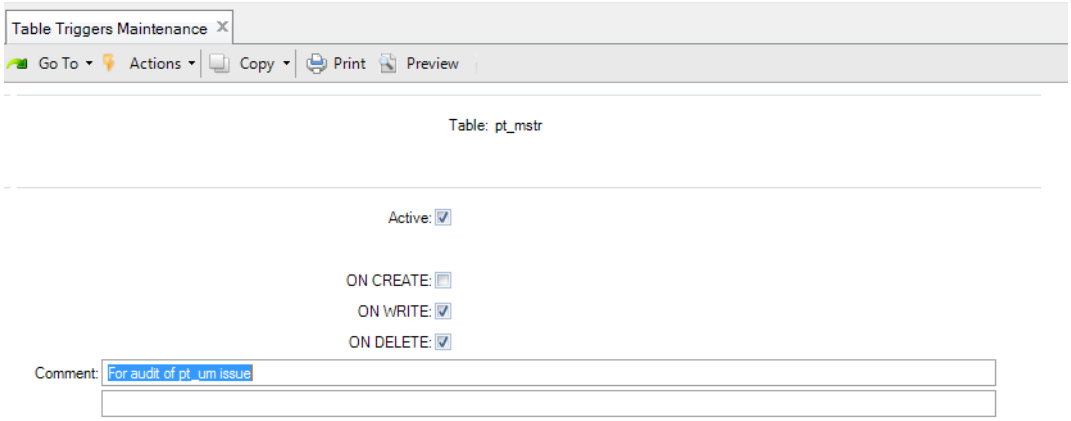
Adding a Custom Procedure on Delete/on Write/on Create of Record

The example shows how to, step by step, add a custom procedure on an ON DELETE/ON WRITE/ON CREATE trigger of the specified table. This example tracks the “write” action on pt_mstr. Of course, the same process can be used to track any other event on any table.

Adding a Custom Procedure on Delete/on Write/on Create of record

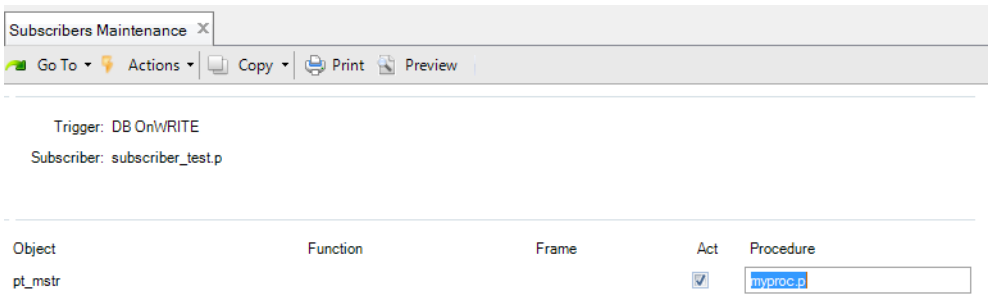


1 Open Table Triggers Maintenance (90.1.13). Activate an ON WRITE trigger for table pt_mstr. Answer Yes to the “Do you want to rebuild Link&Consistency triggers?” prompt. See “Table Triggers Maintenance (90.1.13)” on page 21 for details.



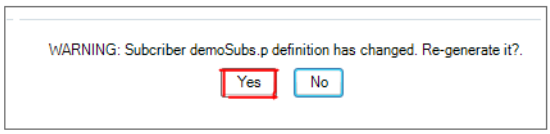
2 Add a subscriber to the ON ASSIGN event using Subscribers Maintenance (90.1.22). Enter the subscriber name and select an ON WRITE event.

3 Type the name of the table as object, then select the program and/or frame where the procedure should be fired. If the program and frame are blank, the procedure will be fired everywhere that ON WRITE on pt_mstr occurs.



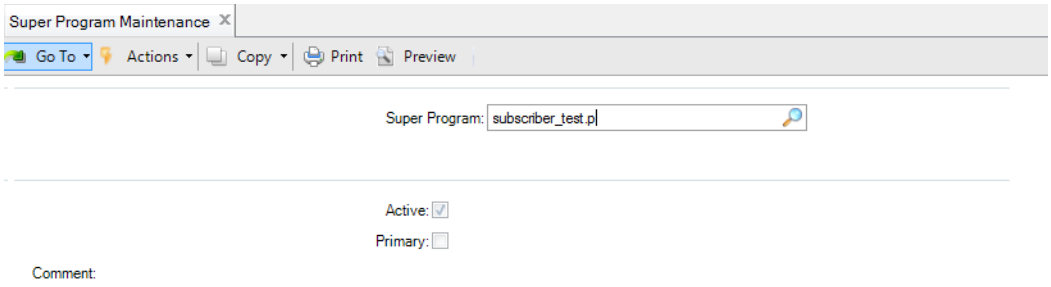
4 Allow ICT to create the procedure from the template. If you use Character UI, the procedure will be edited and can be changed and compiled. Otherwise, you must edit it manually.

5 Allow ICT to regenerate the subscriber file.



Note .NET UI does not let you edit and compile created files. Remember to compile the custom procedure and subscriber files before you perform the next step.

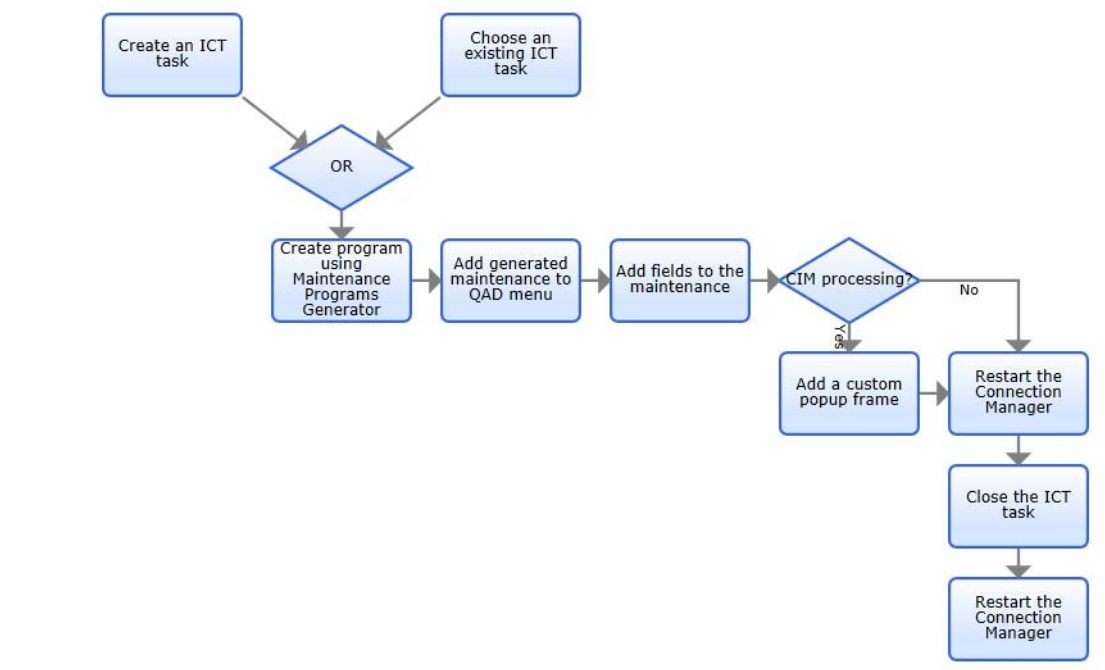
6 Add the subscriber as a super procedure in Super Program Maintenance (90.24.13).



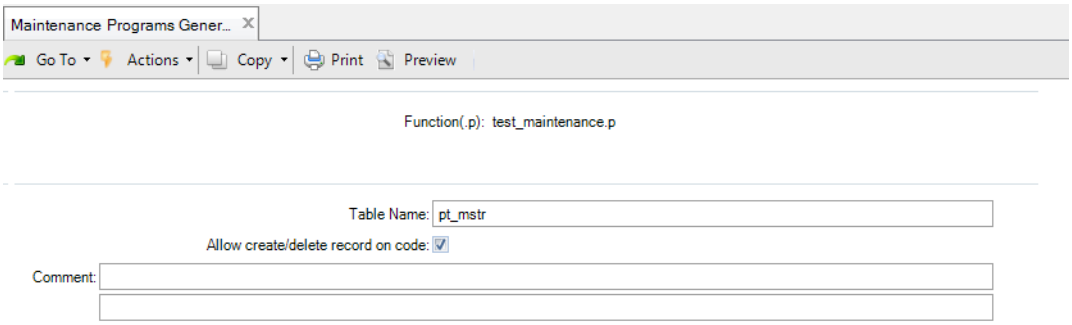
7 Restart the Connection Manager and confirm that your customization works correctly.

Creating a Custom Maintenance Program Using Maintenance Programs Generator

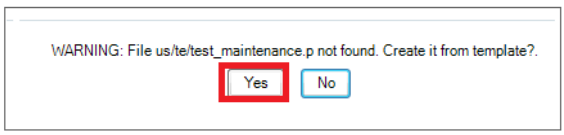
The example shows how to, step by step, create a custom maintenance program for the pt_mstr table using Maintenance Programs Generator. Of course, the same process can be used to create a custom maintenance program for any other table.



1 Open Maintenance Programs Generator (90.3.10) and type the function name and table name. Decide whether additions/deletions should be allowed in the generated program.



2 Allow ICT to create the maintenance program from the template.

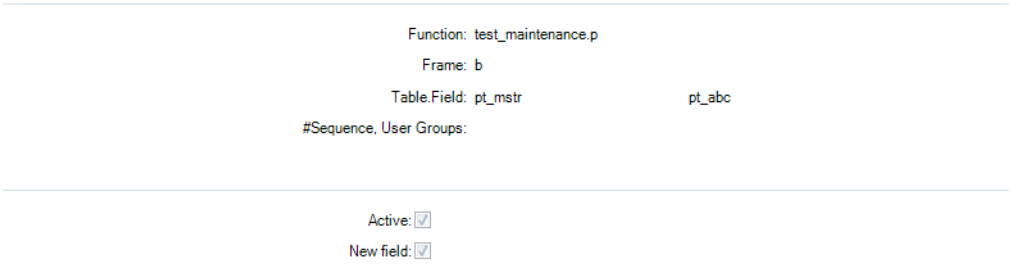


3 The code has been created and saved in the task directory. Use the Character UI to edit and compile the code.

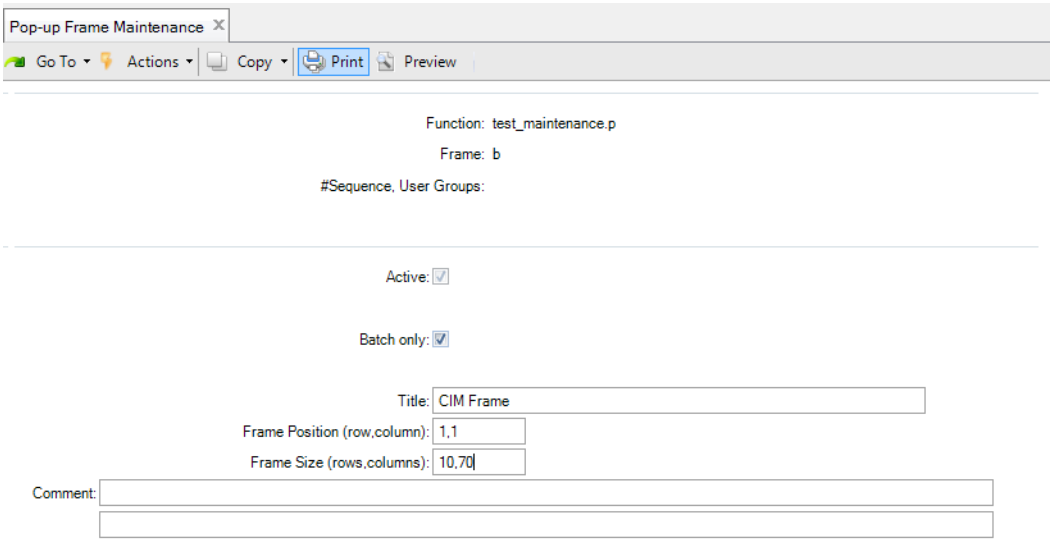
Note .NET UI does not let you edit and compile created files. Remember to compile the custom procedure before you perform the next step.

4 Add the generated maintenance program to the QAD menu. In QAD Enterprise Applications Enterprise Edition, define required permissions.

5 Add required fields to the maintenance program. To do that, open Frame & Field Properties Maintenance (90.3.4) and add new fields to frame b in the generated maintenance program.



6 If CIM processing is required for the generated maintenance program, add a pop-up frame in Popup Frame Maintenance (90.3.1) called “b” with the Batch only flag set to Yes.



7 Restart the Connection Manager and confirm that your customization works correctly.

Appendix A

Function Examples for QAD EE 2011 and Higher

This chapter provides examples of function usage for QAD Enterprise Applications Enterprise Edition version 2011 and higher.

Register Functions

Use register functions to register a definition or value in QAD ICT.

registerTrigger(definition)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

registerTrigger("FIND pt_mstr").
on find of pt_mstr do:
    disable triggers for dump of cp_mstr.
    define variable temp_value as character.
    …
end.
```

registerRecord(identifier,rowid)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

registerTrigger("FIND pt_mstr").
on find of pt_mstr do:
    registerRecord("pt_mstr",rowid(pt_mstr)).
end.
```

registerBuffer(identifier,handle)

Example

```
{us/bbi/mfdeclre.i}
{us/bbiicdef.i}

find first pt_mstr no-lock no-error.
if available(pt_mstr) then
    registerBuffer("pt_mstr", buffer pt_mstr:handle).
```

registerValue(variable_name,character_value)

Example

```
{us/bbi/mfdeclre.i}
{us/bbiicdef.i}

define variable lvc_variable as character no-undo.
assign lvc_variable = "value that should be stored".

registerValue("lvc_variable", lvc_variable).
```

registerPersistent(program,handle)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable lvhHandle as handle no-undo.
{gprun.i ""icuipers.p"" "persistent set lvhHandle"}

registerPersistent("icuipers.p", lvhHandle).
```

registerSuper(program,handle)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

…
if not (qad_key5 = "" or qad_key5 = gsvICTTaskID) then next.

if session:add-super-procedure(getProcHandle(qad_key2, false)) then do:
    if existsInSuper("HandleEvent") then
        registerSuper(qad_key2, getProcHandle(qad_key2, false)).
end.
```

registerWidget(program,widget,handle)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

create fill-in lvwWdg
assign
    name                = entry(3,ttic_key2)
    frame               = lvhFrameHdl
    data-type           = ttic_charfld[2]
    format              = ttic_charfld[4]
    col                 = FCOL(ttic_charfld[5],ttic_charfld[6])
    row                 = FROW(lvhFrameHdl,ttic_charfld[5],ttic_charfld[6])
    sensitive           = no
    .

registerWidget (entry(1,ttic_key2),
                lvwWdg:name,
                lvwWdg:handle).
```

registerShadow(identifier,rowid)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

registerTrigger("FIND xxpt_mstr").
on find of xxpt_mstr do:
    registerShadow("xxpt_mstr",rowid(xxpt_mstr)).
end.
```

Retrieve Functions

getRecord(identifier)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable i_rowid as rowid.

assign i_rowid = getRecord("pt_mstr").
for first pt_mstr where rowid(pt_mstr) = i_rowid exclusive-lock:
end.
```

getBuffer(identifier)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable iphBuffer as handle no-undo.
define temp-table ttpt_mstr no-undo like pt_mstr.

iphBuffer = getBuffer("pt_mstr").
create ttpt_mstr.
Buffer ttpt_mstr:buffer-copy(iphBuffer).
```

getValue(variable_name)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable lvc_variable as character no-undo.

registerValue("my_value", "test").
…
lvc_variable = getValue("my_value").
```

getBufferValue(buffer,field,index)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

find first pt_mstr no-lock no-error.

registerBuffer("pt_mstr", buffer pt_mstr:handle).

display getBufferValue("pt_mstr", "pt_um", 0).
```

getShadow(identifier)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable i_rowid as rowid.

assign i_rowid = getRecord("xxpt_mstr").
for first xxpt_mstr where rowid(xxpt_mstr) = i_rowid exclusive-lock:
end.
```

Screen (Widget Walker) Manipulation Functions and Procedures

getFieldValue(variable,frame,property)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable i_handle as widget-handle no-undo.
```

```
define variable lvc_value as character no-undo.

assign i_handle = widget-handle(getFieldValue("pt_part","a","handle")).
assign lvc_value - i_handle:format.
```

setFieldValue(variable,frame,screen-value)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

run setFieldValue("pt_um", "a", "kg").
```

Additional Functions and Procedures

saveFirstTime(identifier)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

if firstTime("pt_mstr") then do:
    display pt_um.
    saveFirstTime("pt_mstr").
end.
```

popup_frame(name)

Example

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

run popup_frame("my_frame").
```

ICT Debugger

```
{us/bbi/mfdeclre.i}
{us/bbi/icdef.i}

define variable lvrRowid as rowid no-undo.
define variable l as character no-undo.

l = "abc".

{us/ic/icdbg.i &T1='"Task-001"'
            &T2='"Start of xxtest.p"'}

lvrRowid = getRecord("pt_mstr").
find pt_mstr no-lock where rowid(pt_mstr) = lvrRowid no-error.

{us/ic/icdbg.i &T1='"Task-001"'
       &T2='"Start of xxtest.p"'
       &V1=l
       &V2=pt_part
       &V3=pt_abc
       &LEVEL=80
       }
```

Product Information Resources

QAD offers a number of online resources to help you get more information about using QAD products.

QAD Forums (community.qad.com)
Ask questions and share information with other members of the user community, including QAD experts.

QAD Knowledgebase (knowledgebase.qad.com)*
Search for answers, tips, or solutions related to any QAD product or topic.

QAD Document Library (documentlibrary.qad.com)
Get browser-based access to user guides, release notes, training guides, and so on; use powerful search features to find the document you want, then read online, or download and print PDF.

QAD Learning Center (learning.qad.com)*
Visit QAD's one-stop destination for all courses and training materials.

*Log-in required