Support Debugging Tool for Microsoft Dynamics™ GP

User's Guide

Build 19

Application & documentation designed and developed by

David Musgrave
Escalation Engineer – Microsoft Dynamics GP
Microsoft Dynamics Support - Asia Pacific

Contents

Chapter 1: Introduction

The Support Debugging Tool is a Dexterity-based suite of utilities and tools created to assist with the task of supporting Microsoft Dynamics GP (formerly Microsoft Business Solutions – Great Plains).

The Support Debugging Tool has two modes. Standard Mode is a read-only mode which helps with application logging and providing resource and security information. Advanced Mode adds triggering and scripting capabilities, data export and import and dictionary control. Advanced Mode is disabled by default.

Support Debugging Tool contains the following component features:

Feature	Description
Manual Logging Mode	Manually turn on SQL Logging and Dexterity Logging and Profiling
Individual Logging Control	Individually control SQL Logging and Dexterity Logging and Profiling
Dex.ini Settings	Change System and Debugger Dex.ini Settings
Resource Information	Obtain Details of any Table, Form, Window, Field or Report resource
Security Profiler	Monitor all Security check activity
Security Information	Display Security settings for specific resources for a user and company
Automatic Debugger Mode*	Automatically starts logging and watches for specified events
Dictionary Control*	Enable and Disable third party products
XML Table Export*	Export any table(s) to an XML file
XML Table Import*	Import previously exported tables
Runtime Execute*	Run Dexterity sanScript scripts
SQL Execute*	Run Transact SQL scripts
Configuration Export/Import	Export and Import settings
Configuration Maintenance*	Clear Debugging Tool settings
Administrator Settings*	Change system wide Administrator Settings controlling the behavior of the tool
Dex.ini Configuration*	Automatically update Dex.ini settings across multiple workstations.
ScreenShot	Capture and either email or save Screenshots and System Status information
Send Email	Send Email messages from within the application.
About Support Debugging Tool	Display version and build information or un-install Support Debugging Tool

* Signifies an Advanced Mode feature.

Examples of use

The Support Debugging Tool can be used for many purposes. Some examples of these are listed below:

- It can help you identify the specific series of events which lead up to an issue or bug in the code occurring.
- It can be used to quickly and simply turn on all logging and profiling capabilities without restarting the application. This can be useful when looking at performance problems.
- It can be used to find details about dictionary resources (Forms, Windows, Tables, Reports, Fields, Scripts) and other security objects.
- It can be used to identify resources (forms, reports and tables) causing security access issues.
- It can be used to enable and disable third party products or change the order of the products in the launch file.
- It can be used to export data from and import to any table.
- It can be used to run SQL or Dexterity scripts without needing Dexterity or SQL Administration Tools installed.
- It can be used to capture and email or save screenshots of all open windows and a system status report.
- It can be used to send email to the system administrator, even on systems without a locally installed Outlook client.
- It can assist in resolving issues with reports using Report Writer (RW) user defined functions.
- It can roll out Dex.ini setting changes to all workstation in a system.

Support

The Support Debugging Tool is a custom built tool to provide additional capabilities to troubleshoot issues and is not part of the standard Microsoft Dynamics GP released application. Technical support for this tool is not handled via the standard support systems and instead is provided via the public Microsoft Dynamics GP Community Forum. You can use the link below to access the forum:

http://community.dynamics.com/forums/32.aspx

To assist the partners and Microsoft employees who monitor the forum for these questions, please prefix any subject lines with the initials "SDT: ".

Support Debugging Tool Portal

You can also find release histories, FAQ documents and lots of articles as well as download links at the Support Debugging Tool Portal:

http://aka.ms/SDT

Chapter 2: Installation and Configuration

This chapter includes the following sections:

- *Installation, Navigation and Security*
- *Recommended Configuration*
- *SQL Profile Tracing Configuration*
- *Macro Recording Configuration*
- *About Support Debugging Tool*
- *Support Debugging Tool Feedback Survey*
- *Support Debugging Tool and the Web Client*

Installation, Navigation and Security

The Support Debugging Tool is installed by copying the Debugger.cnk (self-installing dictionary), Debugger.txt (readme file) and Debugger.pdf (this user guide manual) to Microsoft Dynamics GP application folder. When Microsoft Dynamics GP is next launched, select "Yes" to include new code.

If installing on a Windows Vista, Windows 7, Windows 8 (or 8.1), Windows Server 2008, Windows Server 2008 R2 or Windows Server 2012 system and User Account Control (UAC) is active, please launch Microsoft Dynamics GP with the Run as Administrator option to complete the installation.

The Support Debugging Tool does not use any SQL tables to store its data, instead it uses an XML file called Debugger.xml. By default this file is stored in the data subfolder beneath the application folder and will be created if required when logging into Microsoft Dynamics GP. The Debugger.xml file may be supplied with the initial installation files when the Support Debugging Tool has been pre-configured.

Security access must be granted to the forms of the Support Debugging Tool before it can be used by users other that those belonging to the POWERUSER security role.

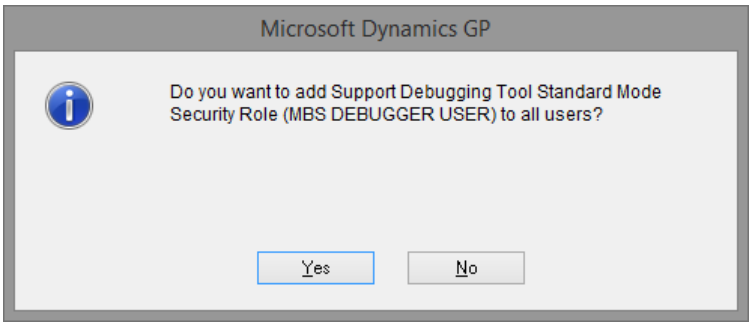
The Support Debugging Tool will automatically create the Security Tasks and Security Roles required to use the tool. The following two Security Roles are created.

MBS DEBUGGER USER (Debugging Tool User)

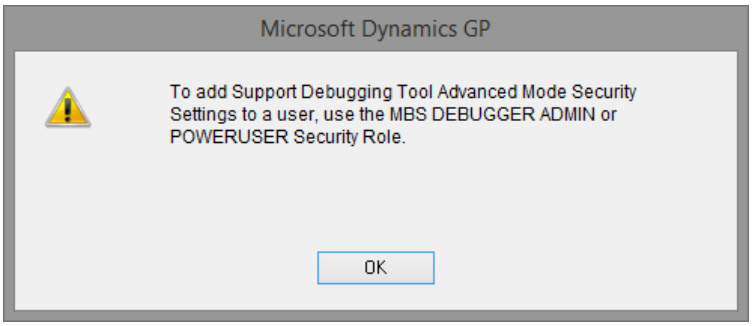
MBS DEBUGGER ADMIN (Debugging Tool Administrator)

The administrator security role grants access to all areas of the tool, while the user security role only grants access to the Standard Mode features. Advanced Mode features are only available to Microsoft Dynamics GP User IDs that also have the SQL Server sysadmin fixed server role or membership of the db_owner role on the system database (DYNAMICS) and the current company database, even if security is granted.

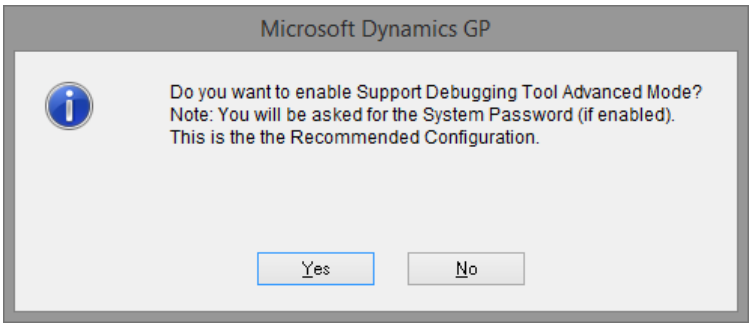
After installing the Support Debugging Tool: If logging into Microsoft Dynamics GP as a user belonging to the POWERUSER security role, and no users have been granted access to the MBS DEBUGGER USER security role, the system will offer to add this security role to all users for you. After which it will guide you through the Recommended Configuration.



If you respond Yes, the system remind to you to add the MBS DEBUGGER ADMIN security role to another other users who need access to the Advanced Mode features and do not already have access to the POWERUSER Security Role.

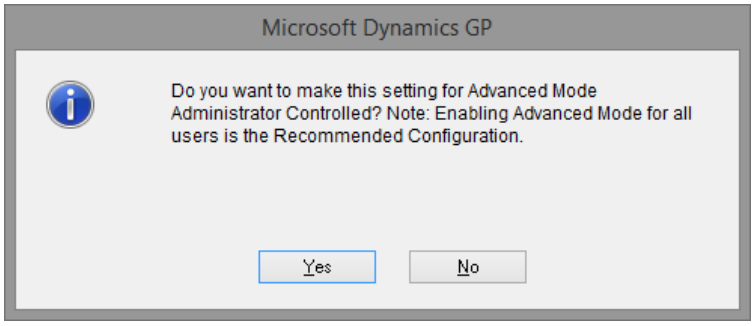


If Advanced Mode is not enabled, it will offer to change that setting for you.

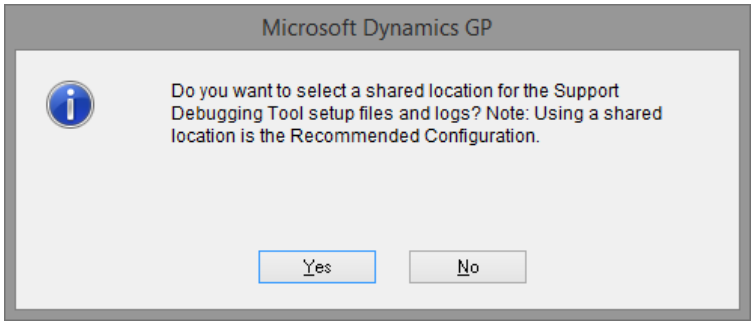


If the System Password is enabled, it will need to be entered before Advanced Mode can be enabled. Enabling Advanced Mode just displays the additional features of the Support Debugging Tool, a user without sufficient privileges will not be able to access the Advanced Mode features.

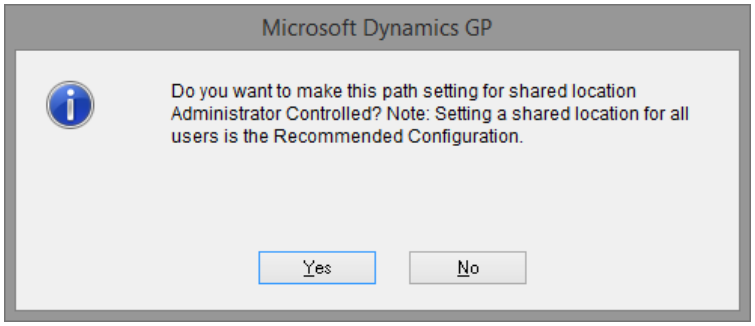
If enabling Advanced Mode you will be asked if you would like to make this Administrator Controlled. Making the setting Administrator Controlled, automatically rolls the setting out to all workstations in the system on their next login and is the Recommended Configuration.



You will then have the option to select a shared location for the setup files and logs to be stored in. If you select No, the default location is the Data folder in application folder for Microsoft Dynamics GP. If you select Yes, you will be presented with a dialog to select the path you wish to use. This path should point to a folder which has full control permissions for all users and can be specified using either a UNC pathname or a shared drive letter available to all users.



If you selected a shared location, then you will be asked if you would like to make this Administrator Controlled. Making the setting Administrator Controlled, automatically rolls the setting out to all workstations in the system on their next login and is the Recommended Configuration.

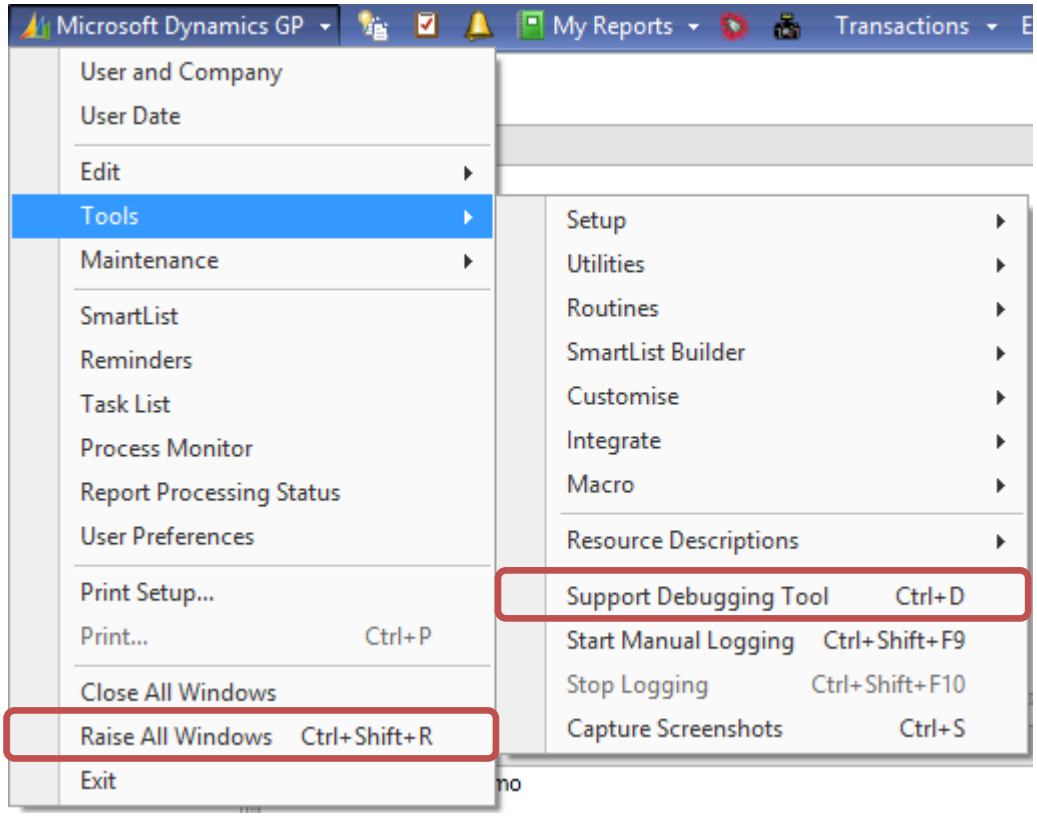


To manually grant security to the forms of the Support Debugging Tool use the User Security Setup window (Microsoft Dynamics GP >> Tools >> Setup >> System >> User Security). After selecting the user and company, select one of the two security roles below:

MBS DEBUGGER USER (Debugging Tool User)

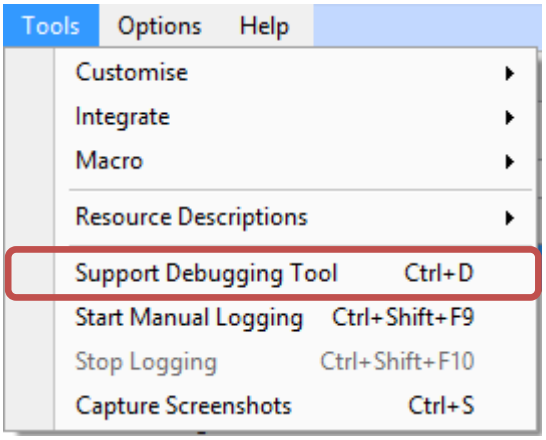
MBS DEBUGGER ADMIN (Debugging Tool Administrator)

Once logged into Microsoft Dynamics GP, a user with security access granted can find the Support Debugging Tool under the Tools menu underneath the Microsoft Dynamics GP menu (highlighted below). It also has the keyboard shortcut Ctrl+D assigned to it.



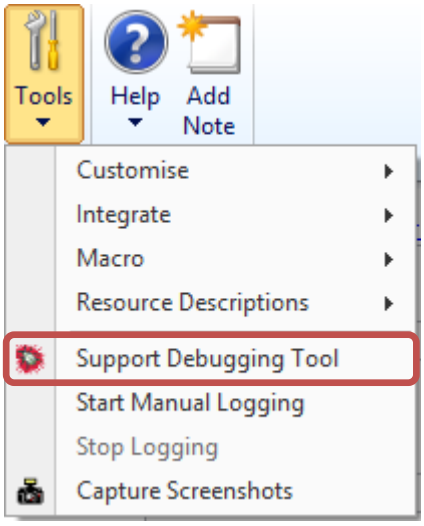
The Support Debugging Tool also adds the Raise All Windows option to the main application menu, to allow for an easy method to send the main application window to the background. It also has the keyboard shortcut Ctrl+Shift+R assigned to it.

In addition, the Support Debugging Tool is also found under the Tools menu on each individual window of Microsoft Dynamics GP (highlighted below).



You may need to press and release the Alt key on the keyboard to allow the window menu bar to activate before the shortcut keys work.

If using Microsoft Dynamics GP 2013 R2 or later in desktop mode with ribbons enabled instead of the menus, you can access the Support Debugging Tool under the Tools button on the ribbon.



If a user is not going to be using any of the windows of the Support Debugging Tool, they do not need to be assigned to a security role. Automatic Debugger Mode will work regardless of security settings.

The Support Debugging Tool can also be opened from the Standard Toolbar and from Quick Links on the Home Page.

When running on the Web Client, use the Quick Links on the Home Page to open the Support Debugging Tool as the other navigation options are not available.

Recommended Configuration

The Support Debugging Tool does not store its settings and data in SQL tables. Instead it uses an XML file called Debugger.xml. The recommended configuration is for the Support Debugging Tool to be installed on all workstations in the system and to point each workstation to use a single Debugger.xml file stored in a shared location.

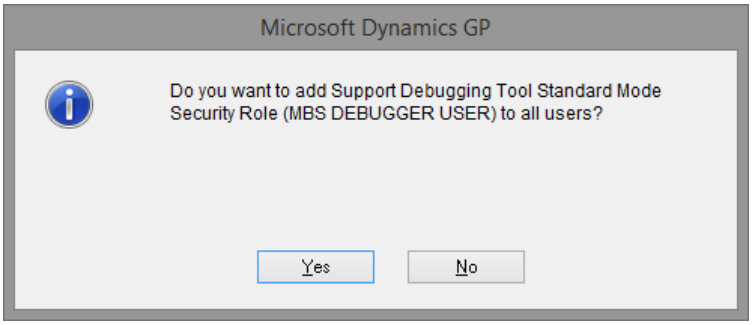
Below are step by step instructions to install and setup this recommended configuration. After the initial installation, you will be asked a series of questions which will set up the Recommended Configuration:

1. Initially install on a single instance of Microsoft Dynamics GP. Just copy the files from the archive to the application folder, usually under C:\Program Files\Microsoft Dynamics\GP.

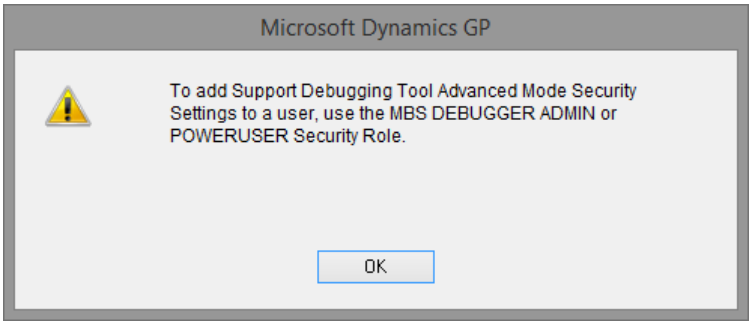
2. Extract the chunk file by launching Microsoft Dynamics GP using Run as Administrator and click Yes if asked "Do you wish to include new code now?"

3. Log into Microsoft Dynamics as 'sa' or a user with similar permissions.

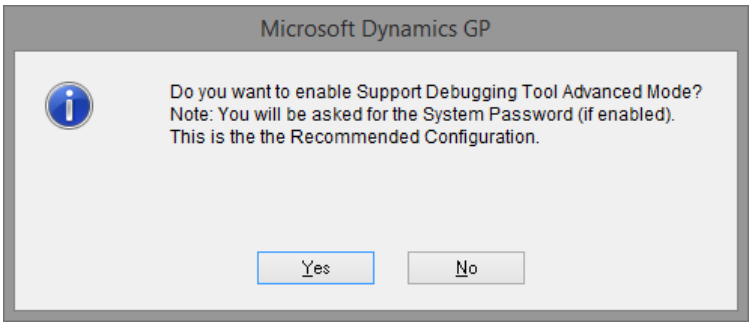
4. If asked to add the Standard Mode Security Settings to all users, click Yes.



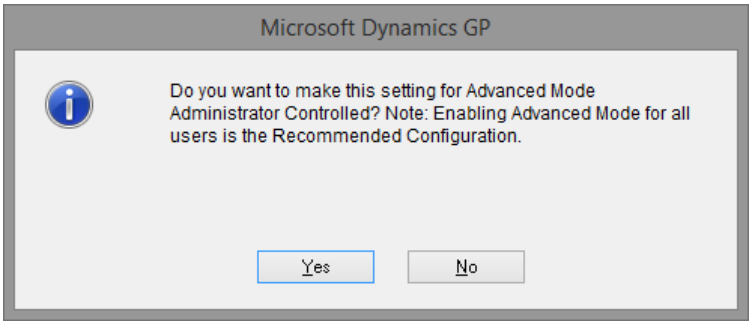
5. You will then be reminded that Advanced Mode security settings will need to be set up manually.



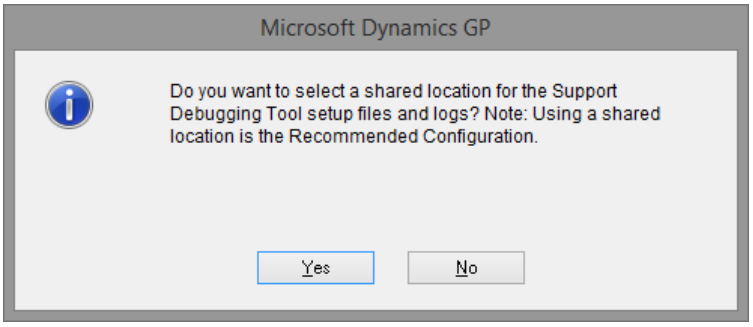
6. If asked about enabling Advanced Mode, click Yes.



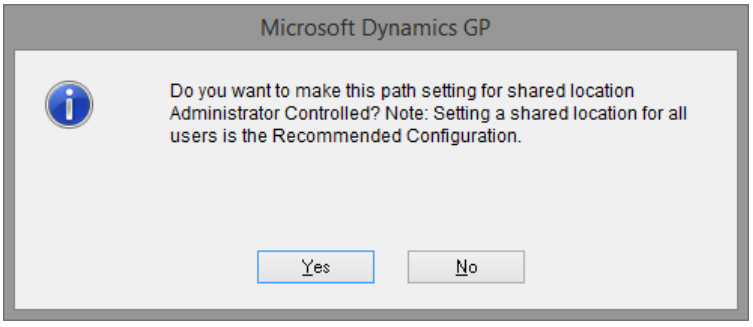
7. If asked about making Advanced Mode Administrator Controlled, click Yes.



8. If asked to select a shared location for the setup files and logs, click Yes and select the path you wish to use. This path should point to a folder which has full control permissions for all users and can be specified using either a UNC pathname or a shared drive letter available to all users.



9. If asked about making the path setting for the shared location Administrator Controlled, click Yes.



10. **Optional:** To manually change security settings, go to the User Security Setup window (Microsoft Dynamics GP >> Tools >> Setup >> System >> User Security), select the appropriate user and company and grant access to one or both of the following roles:

    For Standard Mode features:
    MBS DEBUGGER USER (Debugging Tool User)

    For Advanced Mode features:
    MBS DEBUGGER ADMIN (Debugging Tool Administrator)

It is recommended to grant all users in the system access to MBS DEBUGGER USER. Only System Administrators need access to MBS DEBUGGER ADMIN, unless they already have access to the POWERUSER Security Role.

11. Install the Support Debugging Tool on all other workstations in the system.

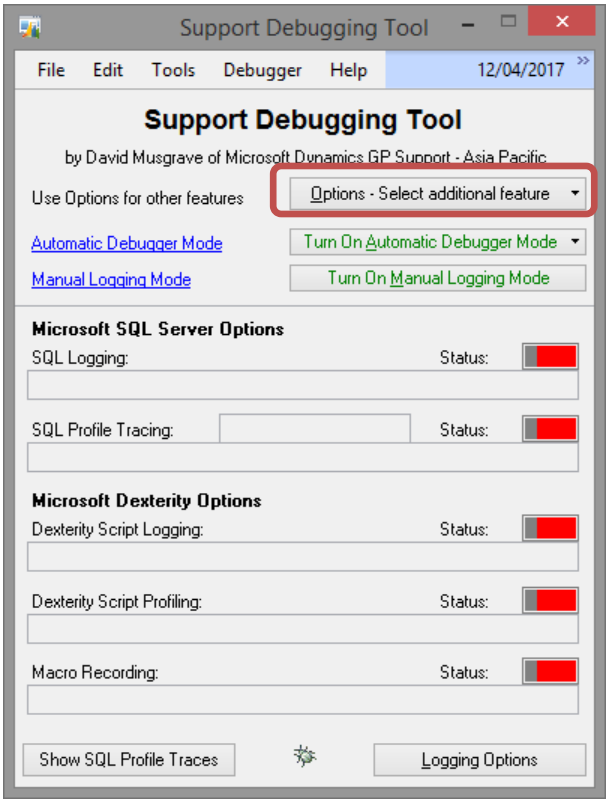
The Recommended Configuration is now be configured. To install on other workstations just requires the copying of the files and the including of new code.

Below are the manual steps showing where the responses to the dialogs for Advanced Mode, Shared Location path and Administrator Controlled settings can be manually changed:

1. Open the Support Debugging Tool main window (Microsoft Dynamics GP >> Tools >> Support Debugging Tool).

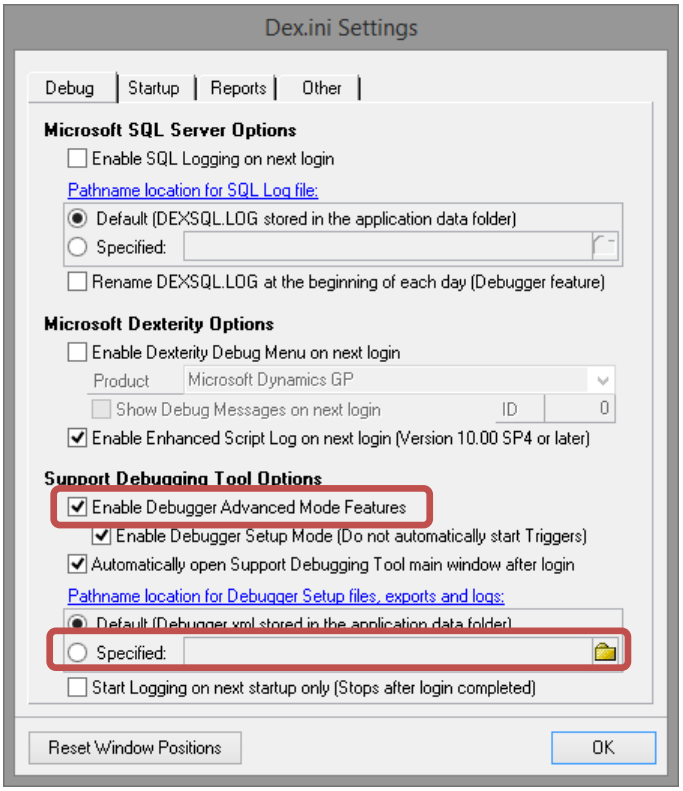
   You can also open the main window from the Tools menu on each individual window.

   Screenshots of menus shown in previous section.
2. From the Support Debugging Tool main window select Options >> Dex.ini Settings.



Unless Advanced Mode is already enabled, the Options menu will only contain the Standard Mode features. Enabling Advanced Mode features will make them available on the menus only if the user has access.

3. From the Dex.ini Settings window, on the Debug tab, make sure the Debugger Advanced Mode Features option is enabled and select a Specified Pathname location for Debugger Setup files. This pathname should be a network share available to all workstations, possibly a folder in the same location that the OLE note attachments are stored.

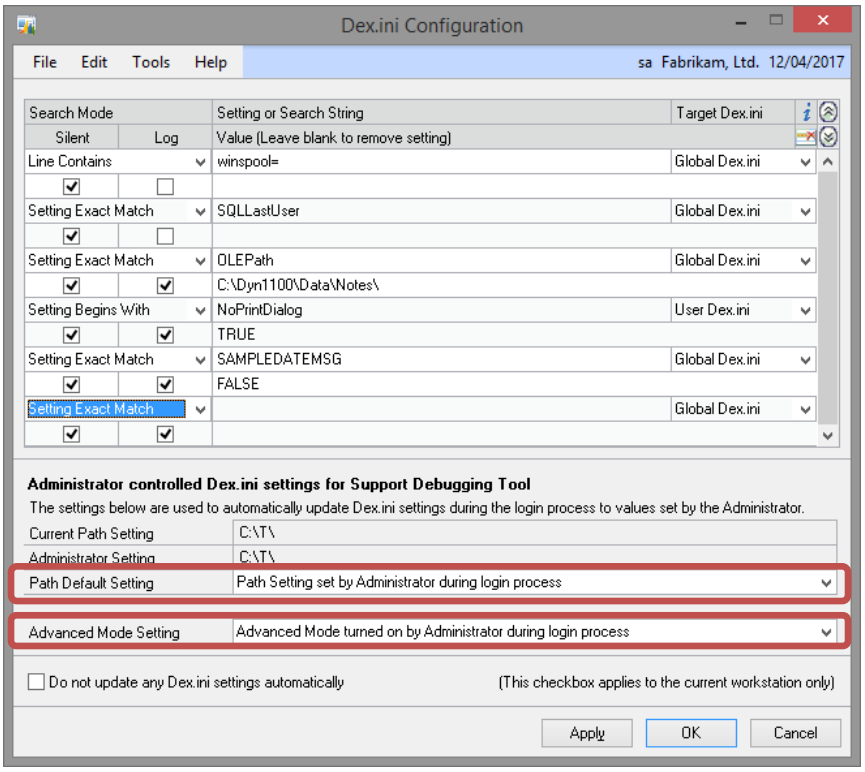


The pathname can be specified using a UNC path in the format \\Server\Share\Folder\.

4. Click OK to save the changes. This will update the current workstation's DEX.INI file only. The next step will allow these two settings to be automatically rolled out to all workstations that have the Support Debugging Tool installed.

Do not enable Debugger Setup Mode on a live system as this prevent Triggers from starting automatically. This mode is designed for use by consultants and support engineers.

5. From the main window select Options >> Dex.ini Configuration to open the Dex.ini Configuration window. Change the Path Default Setting and the Advanced Mode Setting to be controlled by the Administrator during login process.



6. Click OK to save the changes. Now all workstations with the Support Debugging Tool installed will have these two settings automatically applied.

That is all that is required for the Recommended Configuration.

SQL Profile Tracing Configuration

For more information on setting up and enabling SQL Profile Tracing please see the section under the Administrator Settings window.

Below are step by step instructions to configure the recommended settings for SQL Profile Tracing:

1. On the SQL Server machine create a folder on a local drive for where the SQL Profile Trace files will be stored while they are being created. Note this local path for later.

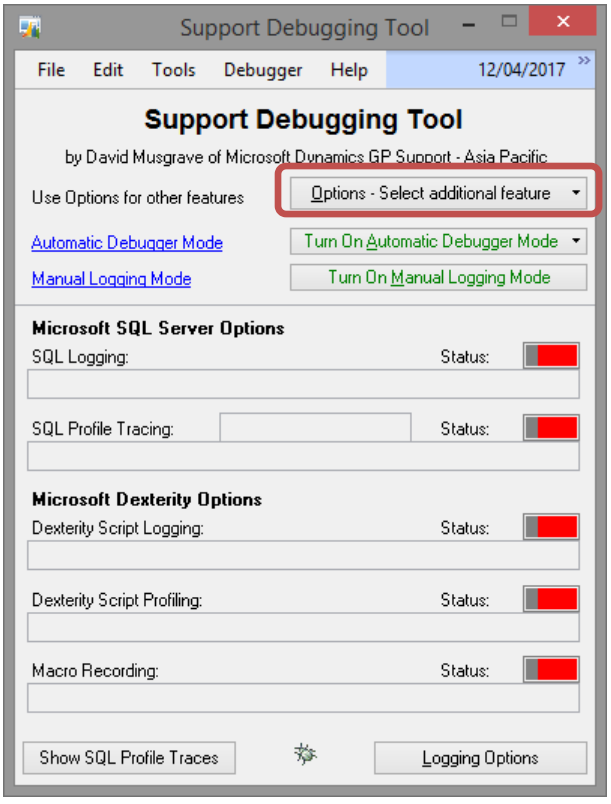
2. Share this local folder on the network, so that all Microsoft Dynamics GP users will have Full Control to the folder. Note this network UNC path for later.

3. Create a user (for example: SQLTraceUser) to be used by SQL Profile Tracing system. The user can be a local user on the SQL Server or a domain user, but needs local Administrator rights on the SQL Server machine. It is recommended to set the password to not expire. Note the User ID and password for later.

4. Log into Microsoft Dynamics as 'sa' or a user with similar permissions.

5. Open the Support Debugging Tool main window (Microsoft Dynamics GP >> Tools >> Support Debugging Tool).

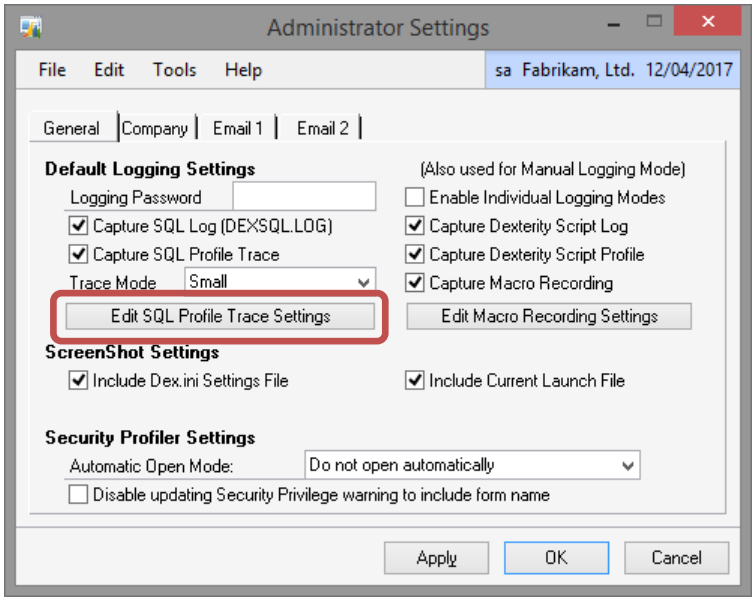
6. From the Support Debugging Tool main window select Options >> Dex.ini Settings.



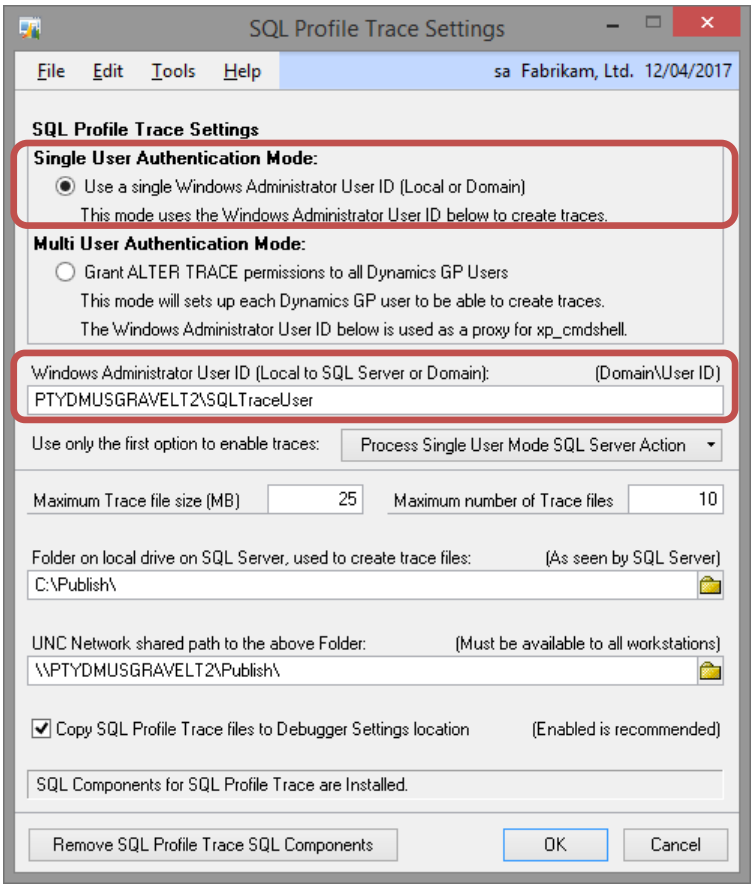
7. From the main window select Options >> Administrator Settings to open the Administrator Settings window.

If the Administrator Settings choice is not available on the menu, then Advanced Mode is not yet enabled. Please see the previous Recommended Configuration section for steps to enable Advanced Mode.

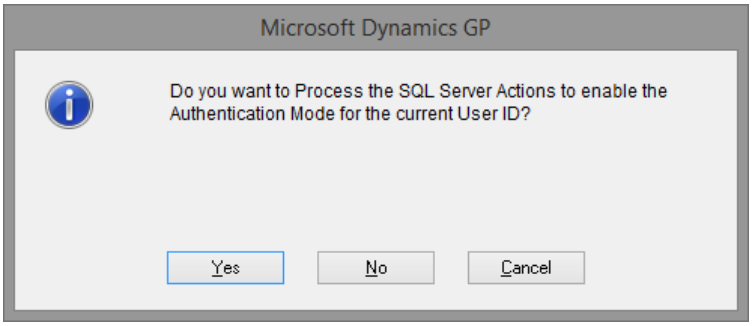
8. From the Administrator Settings window, on the General tab, click Edit SQL Profile Trace Settings to open the SQL Profile Trace Settings window.



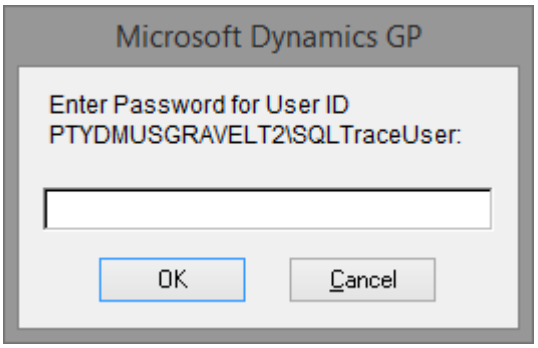
9. On the SQL Profile Trace Settings window, make sure Single User Authentication Mode is selected. In this mode only the single user created earlier will need permissions to create SQL Traces and the permissions for individual users do not need to be changed or elevated.



10. Enter the user created previously and press tab. The system will then ask if you want to process the SQL Server Actions to enable the Authentication Mode, click Yes.

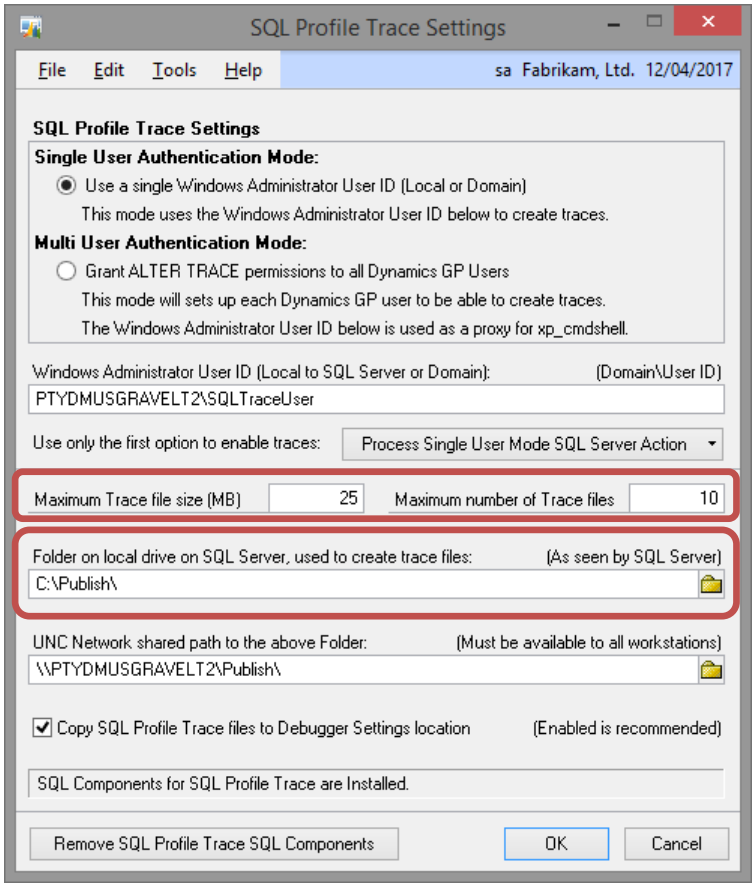


11. As each step of the SQL Server actions needed to enable the Authentication Mode are completed a desktop alert will be displayed. You will also be asked for the password for the user for the Enable xp_cmdshell proxy account step. The password is not validated at this time, so please ensure it is entered correctly.

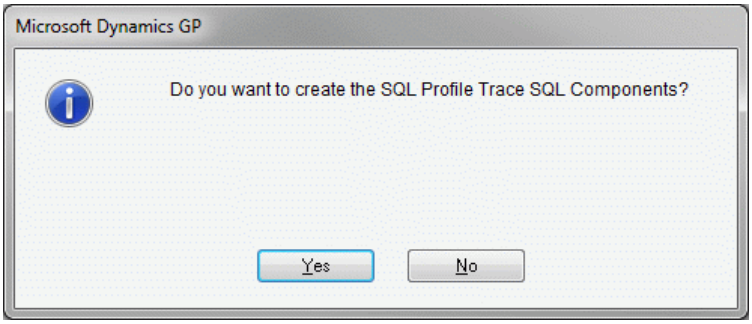


To see the list of individual steps for enabling or disabling the Authentication Mode, click the Process Single User Mode SQL Server Action or Process Multi User Mode SQL Server Action button. You can select to manually run all of the steps or select individual steps from the list.

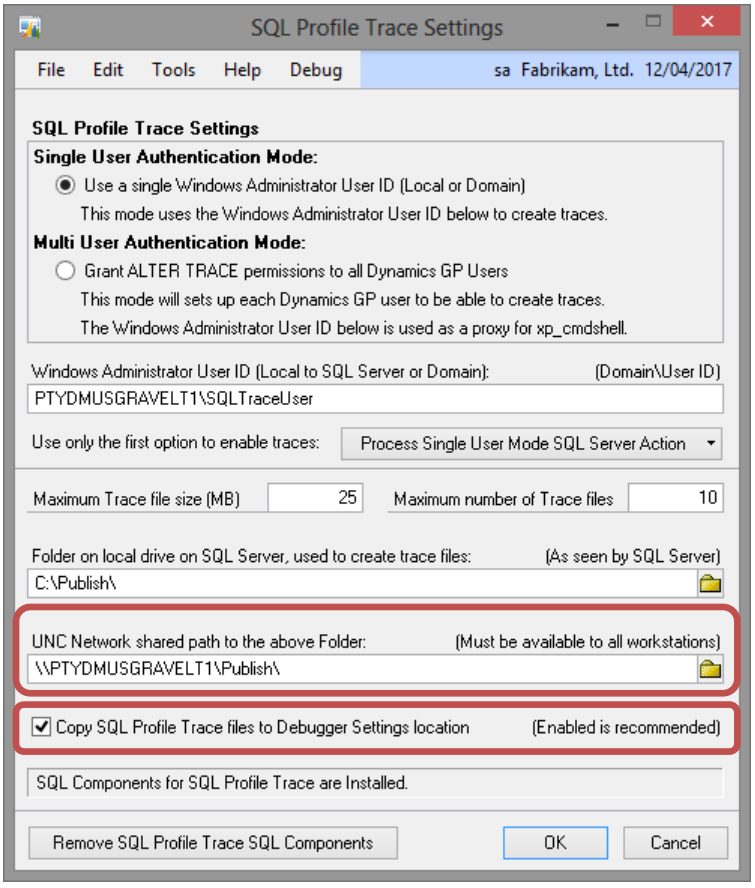
12. You can change the Maximum Trace file size and Maximum number of Trace files if desired, or just leave the default values.



13. Enter in the Local Path set up previously (as created in step 1) and press tab. The system will then ask if you want to create the SQL Profile Trace SQL Components, click Yes to create the stored Procedures in the DYNAMICS system database.



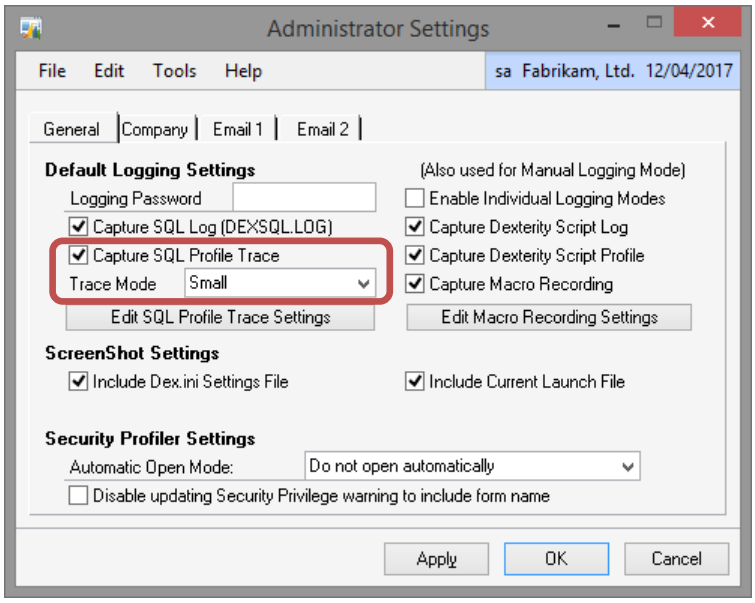
14. Enter the UNC Network Path set up previously (as created in step 2) and press tab.



15. Make sure the Copy SQL Profile Trace files to Debugger Settings location option is enabled. This will copy the completed trace files from the SQL Server to the folder used for the Debugger Settings and logs.

16. Click OK to save the settings and close the SQL Profile Trace window.

17. On the Administrator Settings window, on the General tab, enable the Capture SQL Profile Trace option and set the desired Trace Mode (use Small, if unsure). This will enable SQL Profile Tracing for Manual Logging Mode and as the default value for Automatic Debugger Mode.



18. Click OK to save the settings and close the Administrator Settings window.

Macro Recording Configuration

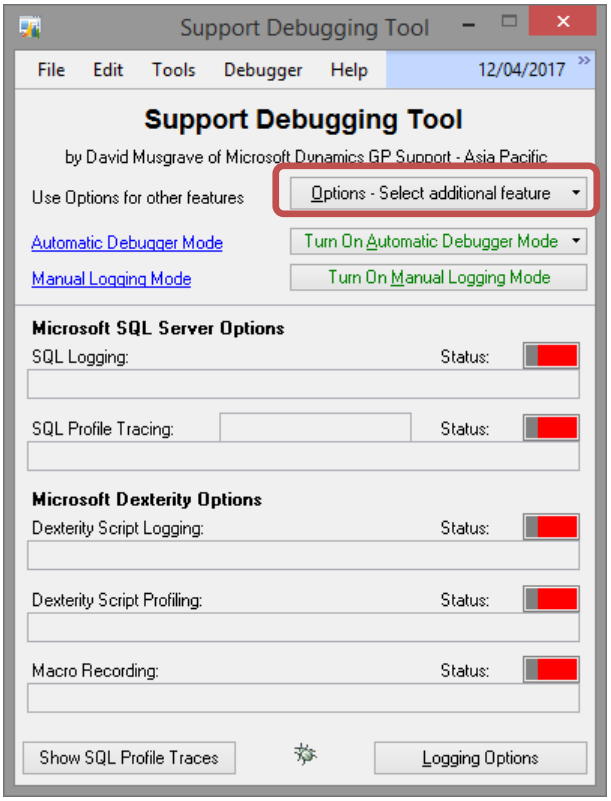
For more information on enabling Macro Recording please see the section under the Administrator Settings window.

Below are step by step instructions to enable Macro Recording:

1. Log into Microsoft Dynamics as 'sa' or a user with similar permissions.

2. Open the Support Debugging Tool main window (Microsoft Dynamics GP >> Tools >> Support Debugging Tool).

3. From the Support Debugging Tool main window select Options >> Dex.ini Settings.

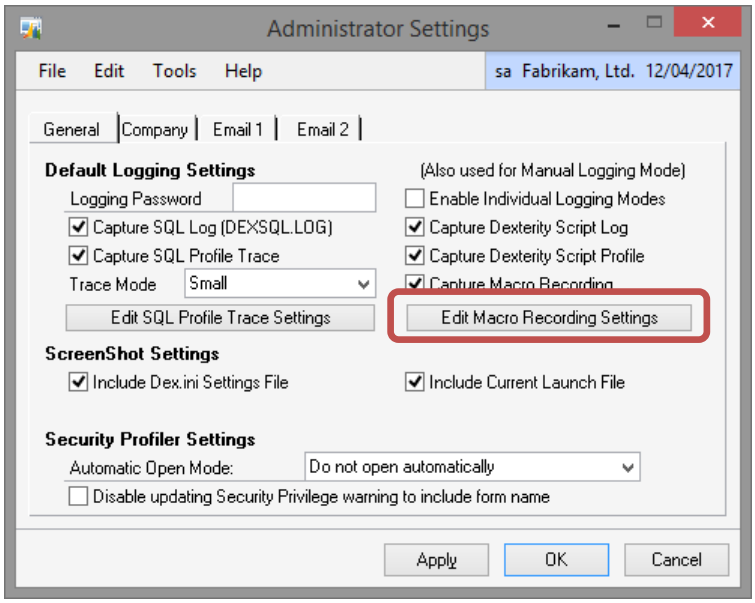


4. From the main window select Options >> Administrator Settings to open the Administrator Settings window.

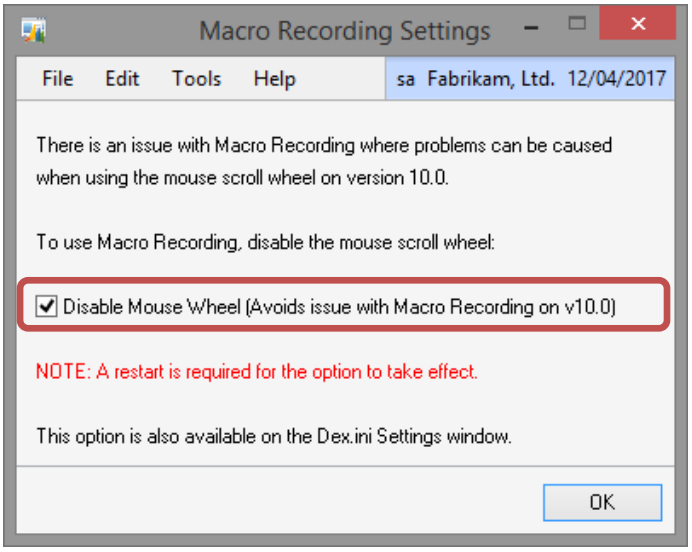
If the Administrator Settings choice is not available on the menu, then Advanced Mode is not yet enabled. Please see the previous Recommended Configuration section for steps to enable Advanced Mode.

5. If running Microsoft Dynamics GP 2010 or later, skip to step 9.

6. For v10.00 only: From the Administrator Settings window, on the General tab, click Edit Macro Recording Settings to open the Macro Recording Settings window.

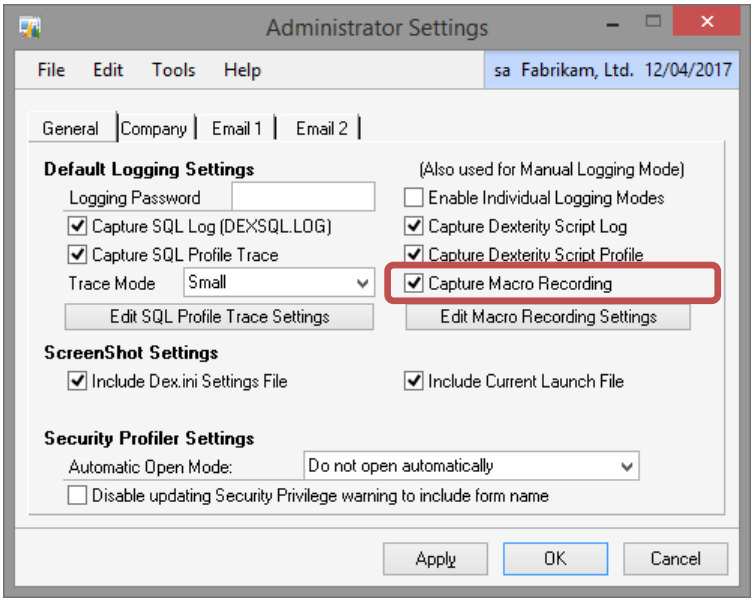


7. For v10.00 only: Check the Disable Mouse Wheel option. Click OK.



8. For v10.00 only: If you changed this option, click OK to close the Administrator Settings window and close and restart the Microsoft Dynamics GP application. Log back in and re-open the Administrator Settings window.

9. On the Administrator Settings window, on the General tab, enable the Capture Macro Recording option. This will enable Macro Recording for Manual Logging Mode and as the default value for Automatic Debugger Mode.

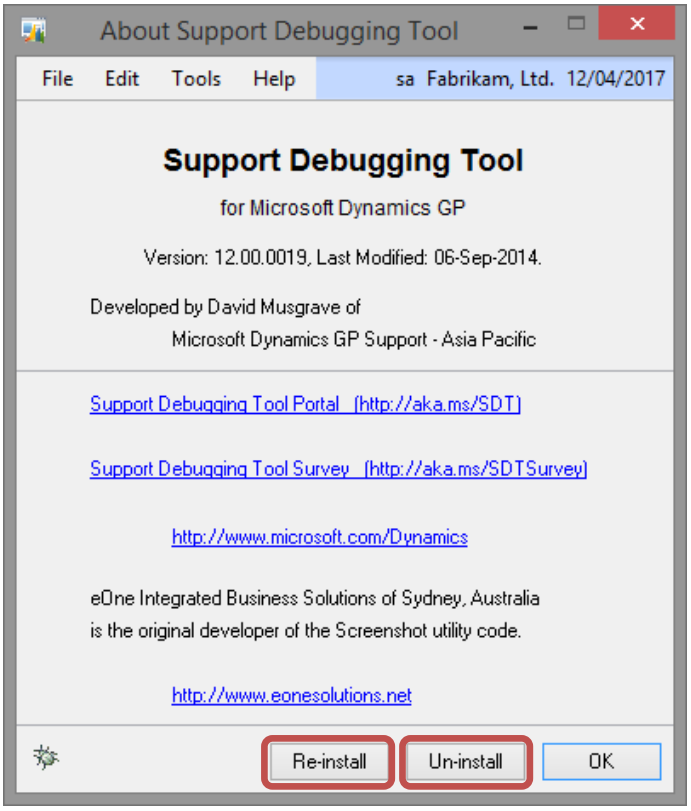


10. Click OK to save the settings and close the Administrator Settings window.

About Support Debugging Tool

You can open this window by selecting About Support Debugging Tool from the Options button drop list on the main window.

The About Support Debugging Tool window shows the current version, build and last modified date information.

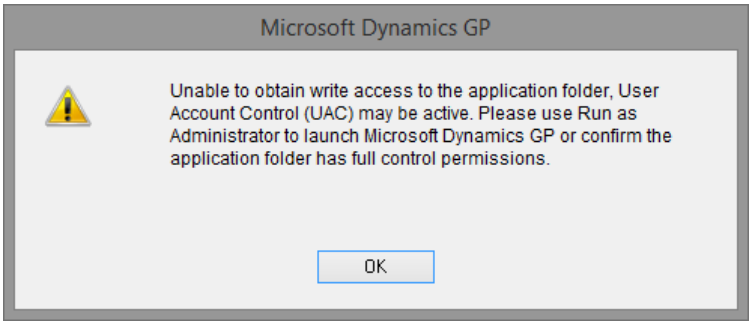


You can un-install the Support Debugging Tool from this window. Clicking Un-install will remove the Support Debugging Tool from the menus and security tables and remove any Dex.ini settings added.

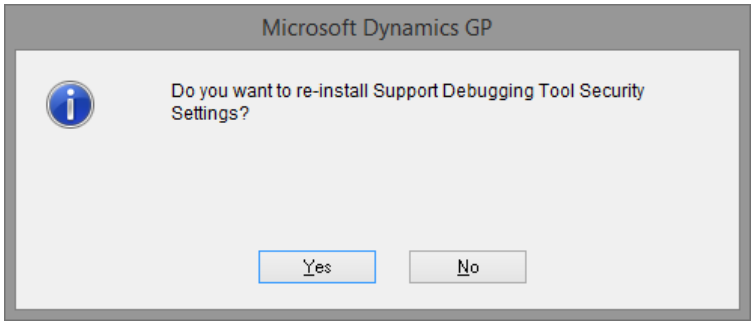
If SQL Profile Tracing is enabled, you will be asked if you want to remove the SQL Server permissions and components created by the Support Debugging Tool.

You will also be asked if you want the Dynamics.set launch file updated to remove the Support Debugging Tool, so that it does not re-install itself next time Microsoft Dynamics GP is launched.

If User Account Control (UAC) is preventing writer access to the application folder, you will see the following dialog displayed. You will need to use Run as Administrator to allow access and complete the un-install.



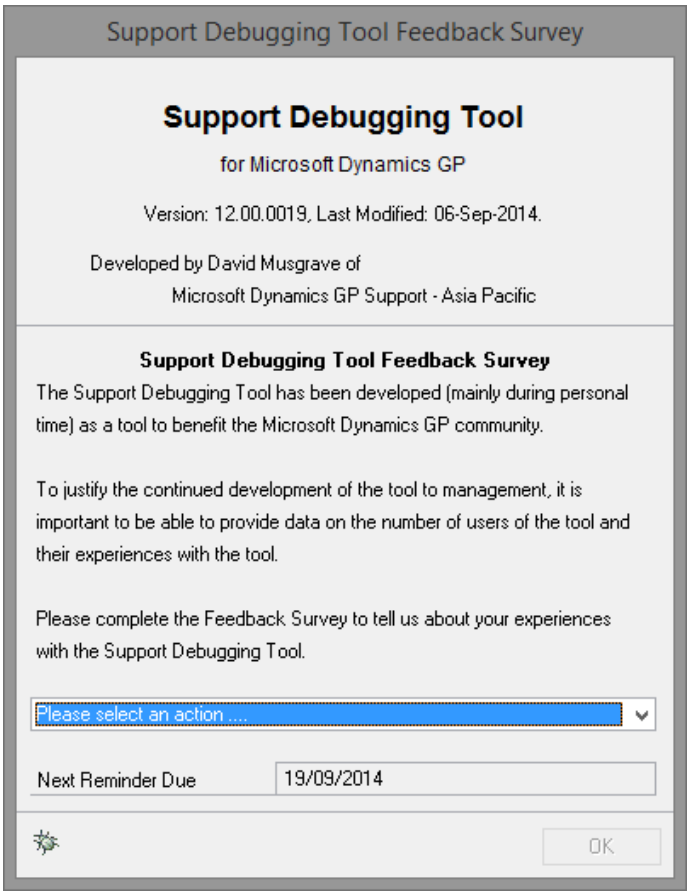
You can also re-install the Support Debugging Tool from this window. Clicking Re-install will, after a confirmation dialog, remove the Support Debugging Tool from the menus and security tables, then re-run the installation as discussed in the Installation section above.



Support Debugging Tool Feedback Survey

The Support Debugging Tool now includes a dialog to prompt users to provide feedback via an online survey (http://aka.ms/SDTSurvey).

The feedback is vital to help provide data back to management on the number of users installing the Support Debugging Tool, what parts of the tool they use and what their satisfaction with the tool is.



The dialog only automatically opens for users with POWERUSER application security or the SQL Server sysadmin fixed server role. It will open two days after a new installation of the Support Debugging Tool or immediately with an upgrade of the Support Debugging Tool.

Once the dialog is opened, a selection of an action from the drop down list is required to close the window and continue. You can decide to complete the Survey which will open the default web browser to the page, or you can decide to postpone the survey to a later time (next login, tomorrow, 30 days, or after installing the next build).

Once completed, the dialog will display the date and user details.

The dialog can also be opened manually by all users from the Options menu on the main Support Debugging Tool window.

Support Debugging Tool and the Web Client

The Support Debugging Tool works with the Web Client, however some features are disabled as the functionality is not supported in the Web Client environment.

Below is a summary of features which are disabled or modified when running on the Web Client:

- *Accessing the tool is only via the Quick Links pane on the Home page.*
- *Macro Logging Mode is disabled.*
- *ScreenShot cannot capture bitmap images, but can email System Status and other files.*
- *Changing Windows Titles to show User and/or Company is disabled.*
- *Changing background colors with Company Color Schemes is disabled.*
- *Microsoft Outlook Client email mode is not supported.*
- *Changing the launch file from Dictionary Control is disabled.*
- *Disabling VBA from Dictionary Control is disabled.*
- *Disabling Visual Studio Tools from Dictionary Control is disabled.*
- *Remembering position and size of windows is disabled.*
- *Using splitters on windows with two panes is disabled.*
- *Desktop Alerts show using a System Dialog.*
- *User Account Control (UAC) checks are disabled.*

Chapter 3: Standard Mode Features

This chapter includes the following sections:

- *Manual Logging Mode*
- *Individual Logging Control*
- *SQL Profile Traces*
- *Dex.ini Settings*
- *Resource Information*
- *Security Profiler*
- *Security Information*
- *Configuration Export/Import*
- *ScreenShot*
- *Send Email*

Manual Logging Mode

The Microsoft Dynamics GP core application runs on the Dexterity runtime engine from which the following logging facilities are available:

SQL Logging

SQL Logging tracks all communication between the Microsoft Dynamics GP client and the SQL Server. The default file in which the SQL communication is stored is named DEXSQL.LOG.

The SQL Logging is tracked on a per workstation basis and will include information from more than one application session, if more than one session of Microsoft Dynamics GP is launched from the same application folder. This is normally the case for Terminal Server and Citrix Metaframe installations.

Communication with the SQL Server using alternate methods of data access is not logged. For example; data access via Visual Basic for Applications (VBA) using ActiveX Data Objects (ADO) will not be captured by SQL Logging.

SQL Profile Tracing

SQL Profile Tracing can be used to log all activity at the SQL Server, including commands from inside Stored Procedures. The default file in which the SQL Profile Trace is stored is named Trace.trc.

SQL Profile Tracing is not enabled until it has been setup using the SQL Profile Trace Settings window under the Administrator Settings.

SQL Profile Tracing will capture all activity at the SQL Server for the current user in the DYNAMICS database and the current company database, so communication with the SQL Server using alternate methods of data access (as described above) will be logged.

Dexterity Script Logging

Dexterity Script Logging tracks all Dexterity event script, procedure and function calls, including the script hierarchy. The default file in which the script log is stored is named Script.log.

Dexterity Script Profiling

Dexterity Script Profiling tracks the number of calls to each event script, procedure and function and how much time the calls have taken. It also tracks all table activity initiated by Dexterity and the time taken. The default file in which the script profile is stored is named Profile.txt.

Macro Recording

Macro Recording captures all activity performed by the user at the User Interface. The recorded Macro can be replayed to repeat the actions, or opened in Notepad.exe for analysis. The default file in which the macro is stored is named Macro.mac.

For v10.00: Macro Recording is not enabled until it has been setup using the Macro Recording Settings window under the Administrator Settings.

Macro Recording can only work when a single instance of Microsoft Dynamics GP is running on a workstation, or if multiple instances are running, Macro Recording will only work on the first instance launched.

Macro Recording is disabled when running on the Web Client.

Using any logging facility will create additional processing overhead for the application. Logging should only be used when actually looking to resolve an issue with the system.

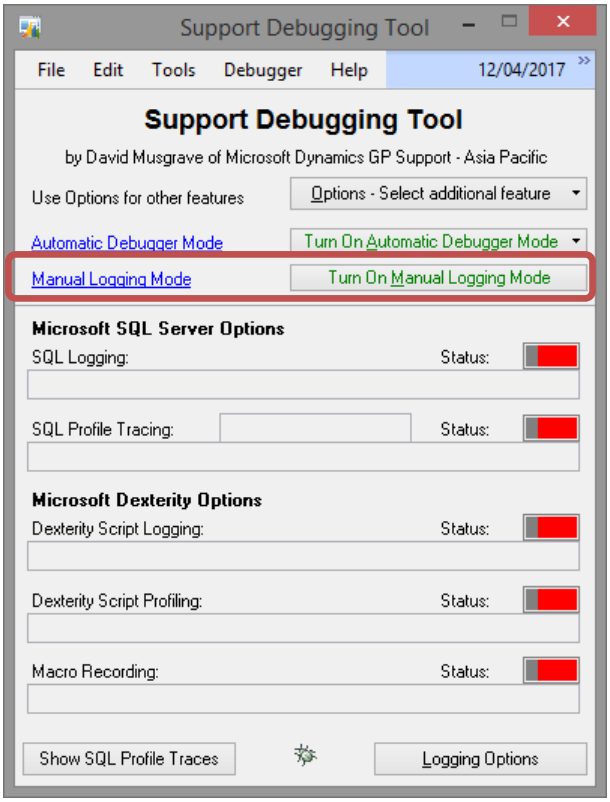
Manual Logging Mode

By default, Manual Logging Mode will activate all of these logging options with a single mouse click without requiring the application to be restarted. The Dexterity runtime will continue to log activity in the application until stopped.

You can use the Administrator Settings window to select which logging modes are enabled when using Manual Logging Mode.

To ensure that the log files are not overwritten, the User, Company and date and time information are appended to the default file name.

To start Manual Logging Mode click on the Turn On button (highlighted below).



To stop Manual Logging Mode click on the same button, now labeled as Turn Off.

Manual Logging can also be turned on using the Ctrl+Shift+F9 keyboard shortcut and turned off again using the Ctrl+Shift+F10 keyboard shortcut.

You may need to press and release the Alt key on the keyboard to allow the window menu bar to activate before the shortcut keys work.

When using Manual Logging Mode to look at a specific issue (possible programming bug or performance problem), it is important to minimize the information captured in the logs to just the events directly related to the issue. To achieve this, request the user to perform all the actions in Microsoft Dynamics GP up to just prior to where the issue occurs. At this point, activate the manual logging and perform the action that exhibits the issue. Once the issue has occurred, stop the logging as soon as possible.

The results of the logging can be found in the folder where the Support Debugging Tool is storing its data files. The default location is the data subfolder beneath the Microsoft Dynamics GP application folder. The location can be changed from the default path using the Pathname location for Debugger Setup files, exports and logs option on the Dex.ini Settings windows (see section in this chapter).

The individual logs will be stored in the following files:

- Debugger_<User>_<Company>.log

  This file will contain all the details of the actions performed by the Support Debugging Tool including the names of the files created during the logging process. Any error or warning messages from the Support Debugging Tool will also be logged to this file.

- DEXSQL_<User>_<Company>.LOG

  These files will contain the SQL Logging results.

- Trace_<User>_<Company>_<Date>_<Time>_<Mode>.trc

  These files will contain the SQL Profile Tracing results.

- Script_<User>_<Company>_<Date>_<Time>.log

  These files will contain the Dexterity Script Logging results.

- Profile_<User>_<Company>_<Date>_<Time>.txt

  These files will contain the Dexterity Script Profiling results.

- Macro_<User>_<Company>_<Date>_<Time>.mac

  These files will contain the Macro Recording results.

<User> will be substituted with the current User ID and <Company> will be substituted with the current Company ID code (InterCompany ID). <Date>_<Time> will contain the date and time at which the logging was started in the format YYYYMMDD_HHMMSS. <Mode> will be replaced with a letters A to E depending on the SQL Profile Trace mode used.

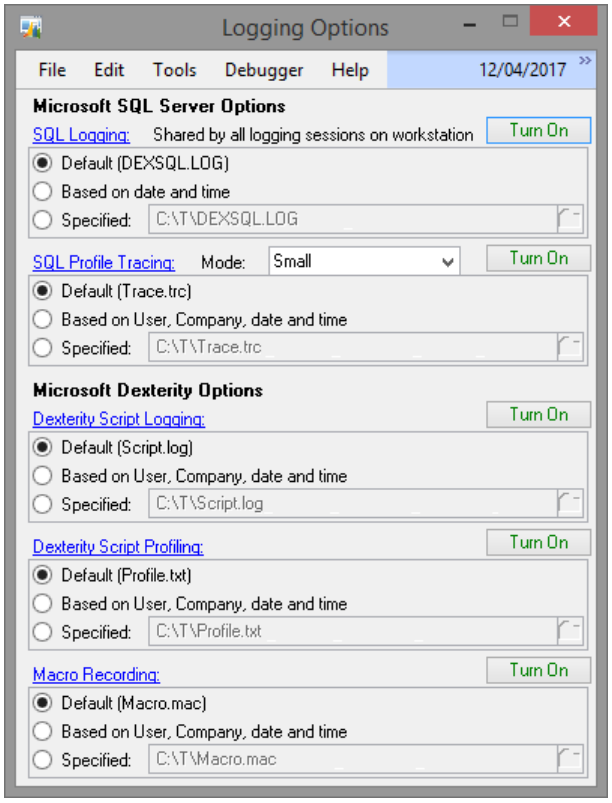
When using the Dex.ini Setting to Start Logging on next startup, the file names used will not have a User ID or Company ID code as these will not be known until after login has completed.

There is an optional password which can be used to control access to Manual Logging Mode. It is set up using the Logging Password field on the General Tab of the Administrator Settings window.

Individual Logging Control

Individual Logging Control allows each of the logging options to be controlled independently. To access the Individual Logging Control features it must be enabled via Administrator Settings. Then you can click on the Logging Options button on the Support Debugging Tool main window.

The pathnames of the resulting files can be left as default, created automatically based on User, Company and date and time information or they can be manually specified.



Turning all logging methods on using the Based on date and time and Based on User, Company, date and time is the same as using Manual Logging Mode.

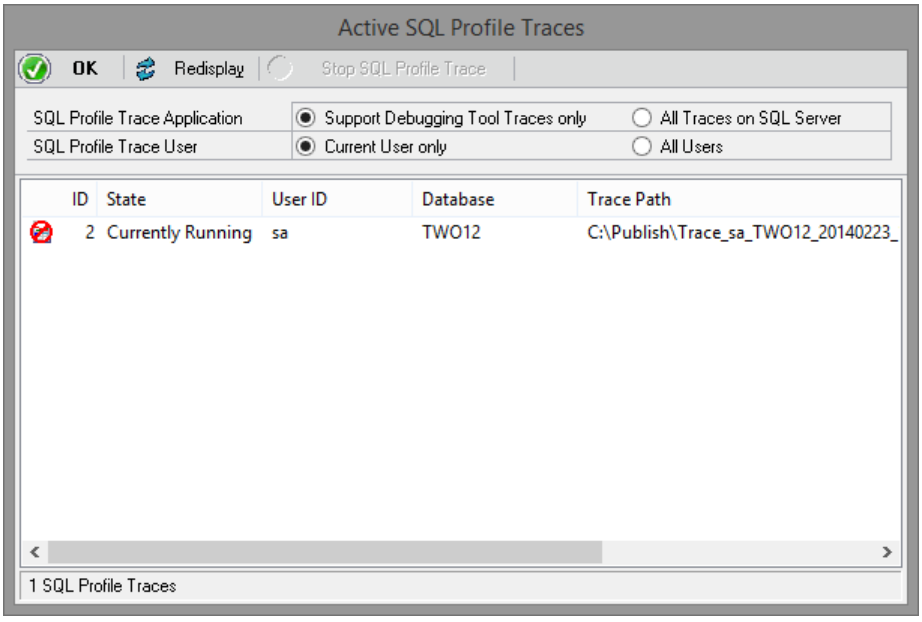
When the Support Debugging Tool is in Advanced Mode you can click on the highlighted prompts for the logging methods to open the folder where the log will be written to.

Access to Individual Logging Modes can be enabled using the Enable Individual Logging Modes option on the General Tab of the Administrator Settings window.

SQL Profile Traces

Active SQL Profile Traces can be viewed by pressing the Show SQL Profile Traces button on the Support Debugging Tool main window. This will open the Active SQL Profile Traces window.

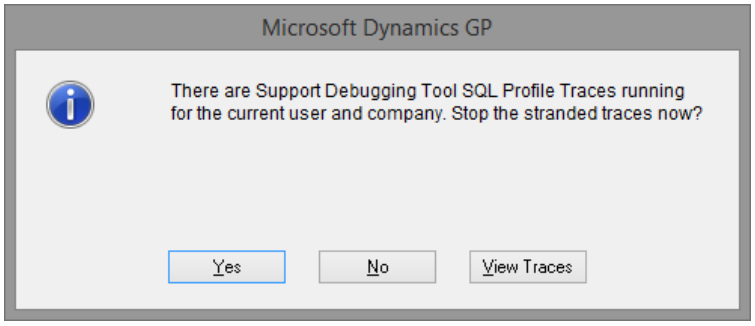
The window defaults to showing Support Debugging Tool Traces only for the SQL Profile Trace Application, and the Current User only for SQL Profile Trace User. A user with the sysadmin rights at the SQL Server level will be allowed to select All Traces on SQL Server or All Users modes.



Stranded SQL Profile Traces are traces created by the Support Debugging Tool where the Microsoft Dynamics GP has unexpectedly terminated and left the trace running at the SQL Server. They can be stopped from this window by selecting the traces (use control and shift keys to multi-select) and then click Stop SQL Profile Trace.

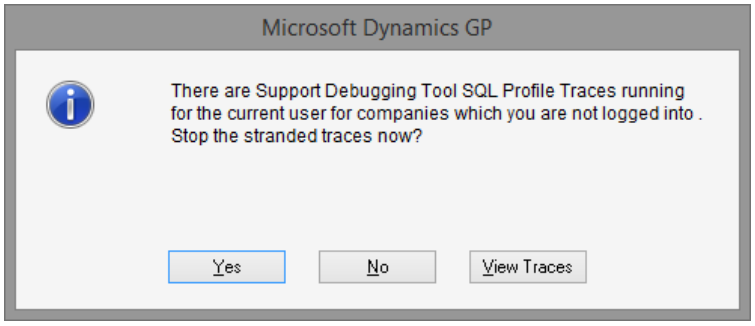
The Show SQL Profile Traces button is enabled once SQL Profile Tracing has been enabled. For more information on setting up and enabling SQL Profile Tracing please see the section under the Administrator Settings window or the section in the previous chapter.

When logging into Microsoft Dynamics GP, if there are stranded traces for the current user and company, the following dialog will be displayed.



The user can select whether to stop the stranded traces, leave them running or open the Active SQL Profile Traces window.

Also when logging into Microsoft Dynamics GP, if there are stranded traces for the current user in other companies which the user is currently not logged into, the following dialog will be displayed.



Again the user can select whether to stop the stranded traces, leave them running or open the Active SQL Profile Traces window.

Dex.ini Settings

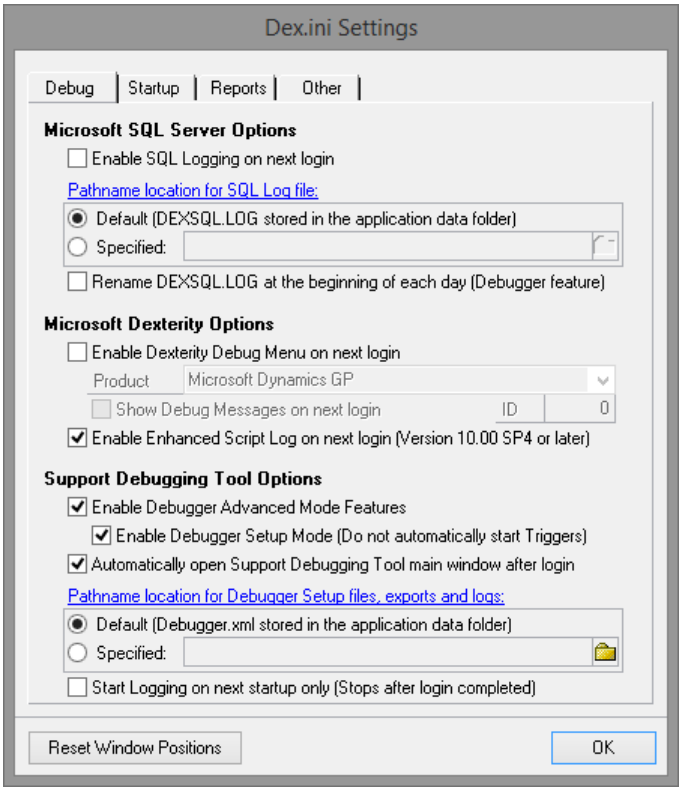
You can open the Dex.ini Settings window by selecting Dex.ini Settings from the Options button drop list on the main window.

The Dex.ini Settings window allows control of some system and Support Debugging Tool options which are stored in the Dex.ini file. It is divided into four tabbed sections.

For Microsoft Dynamics GP 2013 onwards, all settings in this window are stored in the Global level Dex.ini with the exception of the Enable Debugger Setup Mode and Automatically open Support Debugging Tool main window after login options which are stored in the User level Dex.ini.

Debug Tab

The Debug tab contains settings related to the use of the logging and debugging features of Microsoft Dynamics GP as well as settings for the Support Debugging Tool itself.



The following settings are available:

Enable SQL Logging on next login

This option will update the SQLLogSQLStmt and SQLLogODBCMessages Dex.ini settings to enable logging to the DEXSQL.LOG file on next login.

Pathname location for SQL Log file

This option will update the SQLLogPath Dex.ini setting to control the location of the DEXSQL.LOG file. This option controls where the log file is initially created. If using Automatic Debugger Mode or Manual Logging Mode, the log file will be renamed and possibly moved to a different folder.

Rename DEXSQL.LOG at the beginning of each day

This option is added by the Support Debugging Tool to stop the DEXSQL.LOG file growing too large. It renames the log each day. It stores the date when it last renamed the file in the SQLLogRename Dex.ini setting in the format YYYYMMDD. This option should not be used with Automatic Debugger Mode.

Enable Dexterity Debug Menu on next login

This option will update the ScriptDebugger Dex.ini setting to control whether the Debug Menu will be available on next login.

Dexterity Debug Menu Product

This option updates the ScriptDebuggerProduct Dex.ini setting to control the default dictionary Product ID context for the Debug Menu.

Show Debug Messages on next login

This option updates the ShowDebugMessages Dex.ini setting to control whether internal debug message dialogs are displayed when the Debug Menu is enabled.

If the Debug Menu is enabled, it is recommended that the Show Debug Messages option is not enabled for a production system. Having it enabled can cause additional dialogs to be displayed that could be confusing to users.

Enable Enhanced Script Log on next login

This option updates the ScriptLogEnhanced Dex.ini setting to control whether the enhanced Dexterity Script Log features added in v10.00 service pack 4 and later are enabled. Enabling this option adds time stamps and flagging of background processes to the script log. This option is enabled by default by the Support Debugging Tool.

Enable Debugger Advanced Mode Features

Enabling this Support Debugging Tool option will make the Advanced Mode features available on the menus. It uses the MBS_Debug_Mode Dex.ini setting. Enabling this option only makes the Advanced Mode features visible. To access the Advanced Mode features the user must have SQL administrator or database owner (dbo) privileges.

Once Automatic Debugger Mode is setup and enabled, access to the Advanced Mode features are no longer required and can be disabled (hidden) if desired.

Enable Debugger Setup Mode

Enabling this Support Debugging Tool option will prevent Triggers marked to Start Trigger Automatically on Login from starting. Setup Mode is designed to be used by consultants when setting up the Support Debugging Tool for use at a customer's site. It uses the MBS_Debug_SetupMode Dex.ini setting.

Debugger Setup Mode should not be enabled for a production system. It is designed to only be used on test systems or support engineer or partner consultant's workstations.

Automatically open Support Debugging Tool main window after login

This option will make the Support Debugging Tool main window open after a user logs in. It uses the MBS_Debug_AutoOpen Dex.ini setting.

Pathname location for Debugger Setup files, exports and logs

This option allows the location of the Debugger.xml setup file as well as any table exports and captured log files to be located in a specified location. It uses the MBS_Debug_Path Dex.ini setting. This option can be used to allow multiple workstations to use a single setup file.

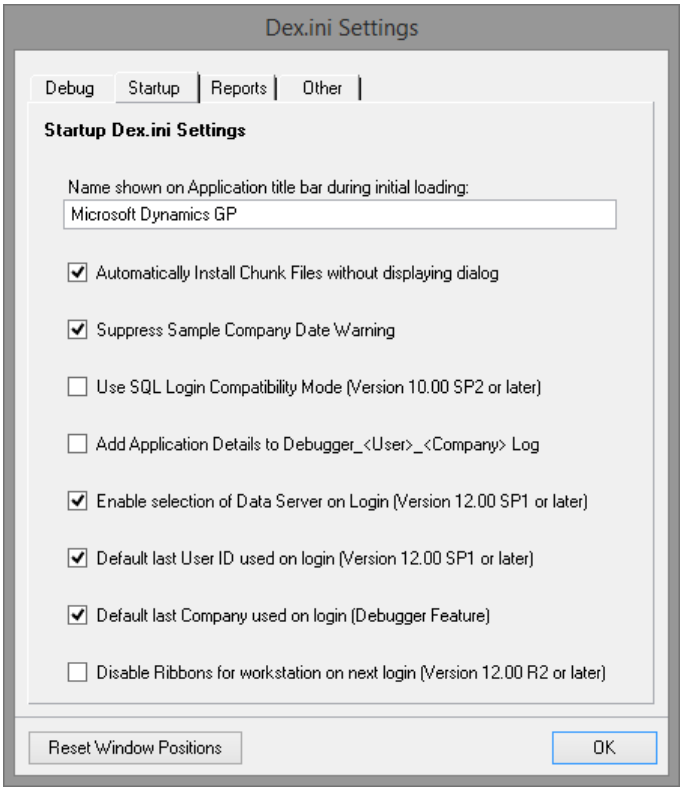
Start Logging on next startup only

Enabling this Support Debugging Tool option will automatically start Manual Logging Mode on application startup. This enables the capture of the logs during the login process. This option will turn itself off after it has been used once. It uses the MBS_Debug_LogOnStartup Dex.ini setting.

You can use the Reset Window Positions Button to clear the Dex.ini settings used for remembering the last window size, position and state for the Support Debugging Tool windows. Be sure all other Support Debugging Tool windows are closed when using this option.

Startup Tab

The Startup tab contains settings related to the startup of Microsoft Dynamics GP.



The following settings are available:

Name shown on Application title bar during initial loading

This option will update the ApplicationName Dex.ini setting to control the name shown by the Dexterity Runtime title bar during application startup. Entering a value into this field will override the default application name of "Dexterity Runtime" while the application is launching. Once the application has launched, the title is updated with the product name as shown in the Dynamics.set launch file.

Automatically Install Chunk Files without displaying dialog

This option will update the AutoInstallChunks Dex.ini settings to allow chunk files to install without the user being prompted.

Suppress Sample Company Date Warning

This option will update the SAMPLEDATEMSG Dex.ini setting to allow Microsoft Dynamics GP to login to the Fabrikam sample company without displaying the date warning dialog.

Use SQL Login Compatibility Mode

This option will update the SQLLoginCompatibilityMode Dex.ini setting to allow Microsoft Dynamics GP to continue attempting to login using backwards compatible password encryption methods. This option requires v10.00 service pack 2 and later.

If you continue to use SQL Login Compatibility Mode, a failed login attempt will register as four attempts at the SQL Server. This can prematurely lock out a user when an incorrect password is entered (when enforce password policy is enabled for the SQL Login and the SQL Native Client is used for the ODBC DSN configuration).

Add Application Details to Debugger_<User>_<Company> Log

This option will add an entry into the Debugger_<User>_<Company>.log file each time a user logs into a company. It uses the MBS_Debug_LogAppDetails Dex.ini setting.

Enable selection of Data Server on Login

This option controls whether the Server drop down list on the Login window is enabled. It uses the EnableServerDropDown Dex.ini setting.

Default last User ID used on login

This option controls whether the last User ID used is defaulted in on the Login window. It uses the DefaultLastUser Dex.ini setting.

Default last Company used on login

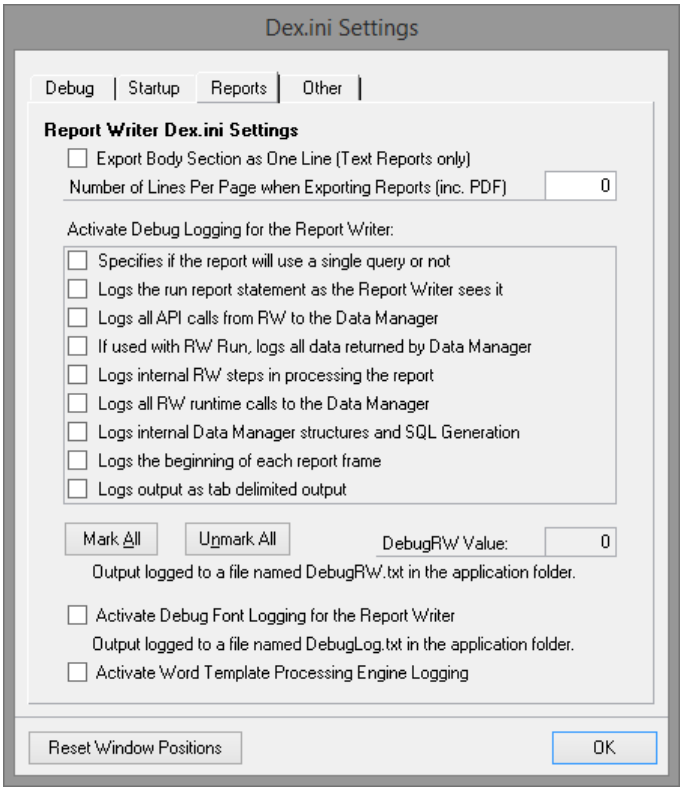
This option controls whether the last Company used is defaulted in on the Company Selection window. It uses the DefaultLastCompany Dex.ini setting.

Disable Ribbons for workstation on next login

This option can disable Ribbons on the desktop client for the current workstation. It uses the EnableWCRibbons Dex.ini setting.

Reports Tab

The Reports tab contains settings related to the behavior and debugging of the Microsoft Dynamics GP Report Writer.



The following settings are available:

Export Body Section as One Line

This option will update the ExportOneLineBody Dex.ini setting to control how the body section on a text report is printed. This option can be used when creating reports to be exported as tab-delimited or comma-delimited text files.

Number of Lines Per Page when Exporting Reports (inc. PDF)

This option will update the ExportLinesPerPage and ExportPDFLinesPerPage Dex.ini settings to control the number of lines on a report page when the report is exported rather than printed to a file (including PDF files).

Suggested values for this setting are 72 for A4 paper in portrait, 51 for A4 paper in landscape, 68 for US Letter paper in portrait and 52 for US Letter paper in landscape. Some trial and error testing might be required to find the best value.

Activate Debug Logging for the Report Writer

These options will update the DebugRW Dex.ini setting to ask the Report Writer to output a debugging log to the file DebugRW.txt. The actual value written to the Dex.ini is shown in the DebugRW Value field.

Mark All

Use this button to activate all the Report Writer debug logging.

Unmark All

Use this button to turn off Report Writer debug logging.

Activate Debug Font Logging for the Report Writer

This option will update the DebugFonts Dex.ini setting enable logging of font selections made by the Report Writer. The results will be written to a DebugLog.txt file. For more information see Knowledge Base (KB) article 870341:

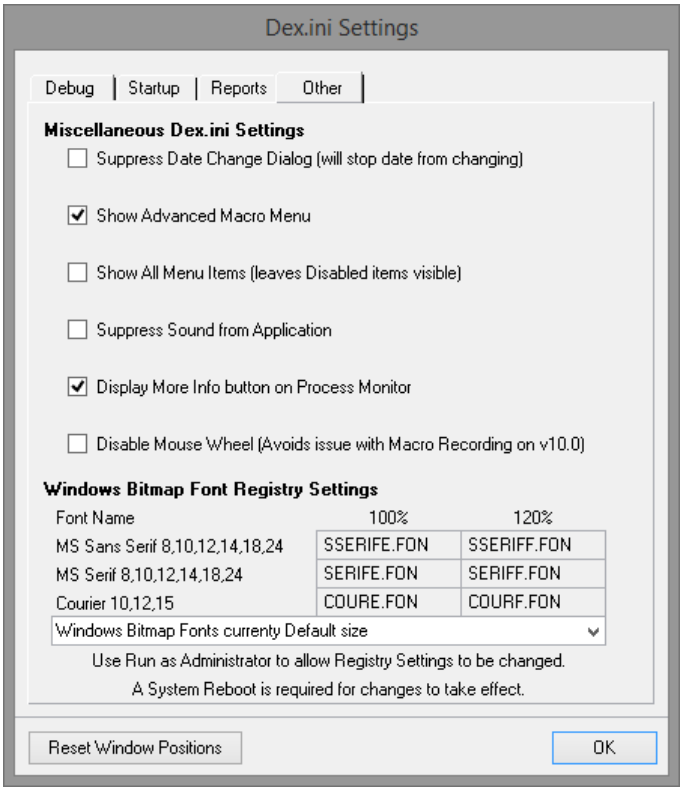
http://support.microsoft.com/kb/870341

Activate Word Template Processing Engine Logging

This option will update the TPELogging and the KeepTemplateTempFiles Dex.ini settings to log the workings of the Template Processing Engine (TPE). The following files will be created in the %TEMP% folder: the TemplateProcessing*.txt file, the document file and the template file.

Other Tab

The Other tab contains other miscellaneous settings for use with Microsoft Dynamics GP.



The following settings are available:

Suppress Date Change Dialog

This option will update the SuppressChangeDateDialog Dex.ini setting to prevent the dialog to change the User Date from being displayed at midnight. Using this option will also stop the date from being changed in Microsoft Dynamics GP.

Show Advanced Macro Menu

This option will update the ShowAdvancedMacroMenu Dex.ini setting to enable the Advanced Macro Menu from the Tools >> Macro menu.

Show All Menu Items

This option will update the ShowAllMenuItems Dex.ini setting to show all menu items, even when the module is not installed, not registered or access has been denied.

Suppress Sound from Application

This option will update the SuppressSound Dex.ini setting to suppress all sound from Microsoft Dynamics GP.

Display More Info button on Process Monitor

This option will update the QueueMoreInfo Dex.ini setting to display the More Info button on the Process Monitor window (Microsoft Dynamics GP >> Process Monitor).

Disable Mouse Wheel

This option will update the MouseWheel Dex.ini setting to disable Mouse Wheel Scrolling in the application. This helps with an issue with Macro Recording in v10.00.

A restart of the application is required for this setting to take effect.

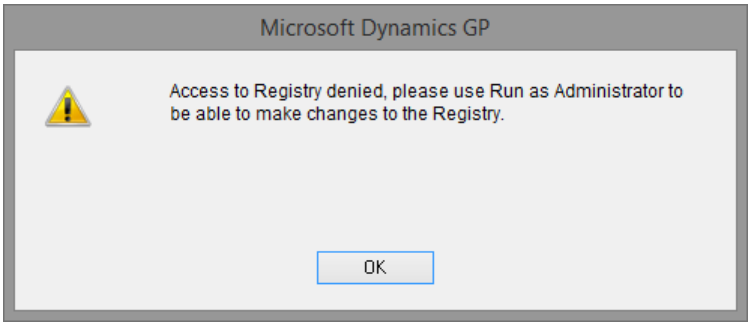
For v10.00: Macro Recording is not enabled until it has been setup using the Macro Recording Settings window.

Windows Bitmap Font Registry Settings

This option will attempt to change the registry to update the font files used for bitmap fonts under Windows 7 and later. These settings are initially created when the operating system is first installed and are not changed when changing the DPI setting for the system. If the fonts in the Microsoft Dynamics GP windows are not being displayed at the correct size, use this option to change the sizes.

A restart of the operating system is required for this setting to take effect.

On an operating system with User Account Control (UAC) enabled, Registry changes are only allowed if the application has been launched using Run as Administrator. If access to the registry is denied the following warning will be displayed:

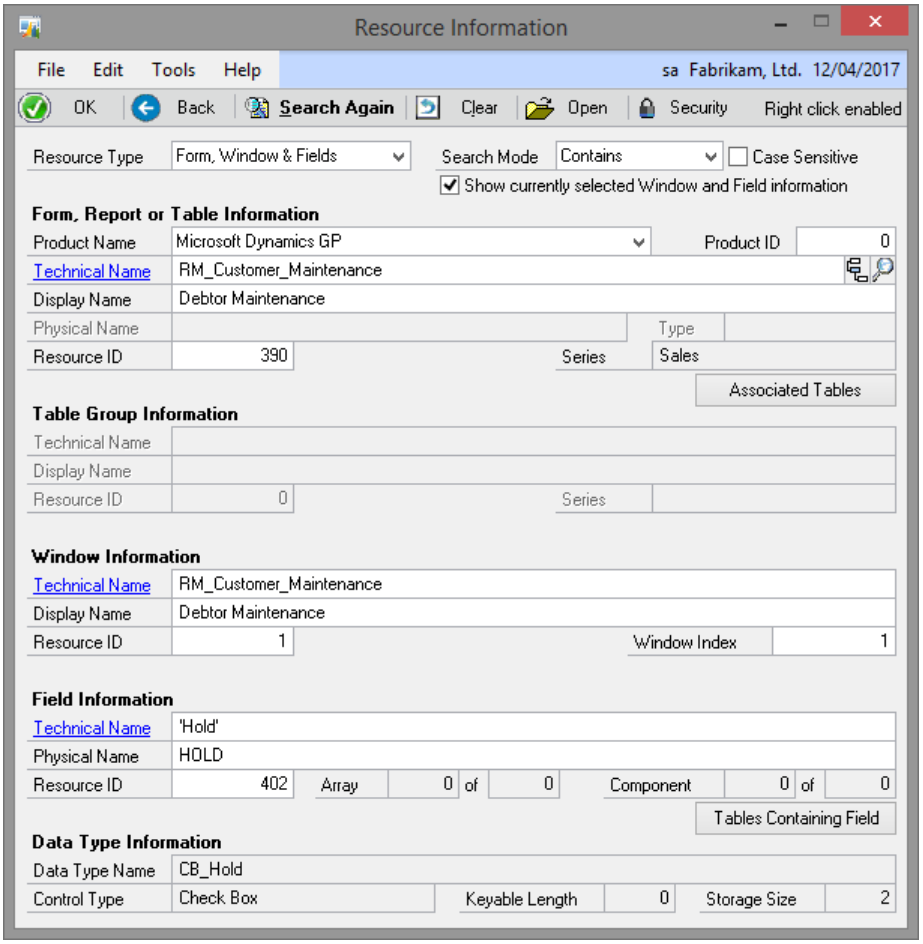


Resource Information

You can open the Resource Information window by selecting Resource Information from the Options button drop list on the main window.

The Resource Information window will display technical, display, and physical names and resource IDs for any form, window, field, table, table group, report or script (procedure or function) resource in the any dictionary currently installed in the Microsoft Dynamics GP application.

It can also provide information about non-dictionary resource Security Objects, such as Customization Tools, Document Access, Letters, Microsoft Dynamics GP Import, Navigation Lists, Series Posting Permissions, and SmartList Objects. If the products are installed, the following objects are also supported, SmartList Builder Permissions and Extender Resources. Security objects from other 3rd party products will show as Unknown Objects.



To use this window, enter the information you know into the appropriate field and the rest of the fields will be populated with the details for that resource.

For example, entering a window's display name will identify the window's technical name and resource ID; or entering a table's physical name as it appears in SQL Server will identify the table's dictionary, technical and display names as well as the resource ID.

This window can be useful when working with table and column names in SQL Server, because it will quickly convert the physical names used in SQL back to the technical names used in Modifier, Report Writer and Dexterity.

Below is a description of the individual fields on the window:

OK Button

This button will close the Resource Information window.

Back Button

This button work backwards through the history of searched resources since the window was opened.

Search Again Button

This button will search for the next resource to match the search criteria. Searching again works for Technical, Display and Physical Names for all resource types. The mode of the search can be controlled by the Search Mode drop down list and the Case Sensitive checkbox.

Clear Button

This button will clear the current search in preparation for a new search.

Open Button

This button will open the selected form or report resource.

Reports opened in this way will not have any options or restrictions applied and might contain unpredictable results. If the report uses a temporary table, this table will contain no data. Opening forms and reports from this window is only for testing purposes.

Security Button

This button will open the Security Information window for the selected resource. See sections below for more information. The Security Button will only be available if the current user has security access to the security windows under Tools >> Setup >> System.

Resource Type

This drop down list controls whether Resource Information window is searching for Forms, Windows & Fields; Tables & Fields; Reports; Security Objects; or Procedures & Functions.

Search Mode

This drop down list controls how text searches will be handled by the Resource Information window. The options are Exact Match, Begins With and Contains. The default setting is Exact Match.

Case Sensitive

This checkbox controls if the text searches on the Resource Information window will be case sensitive or not. The default setting is to be case sensitive.

Show currently selected Window and Field information

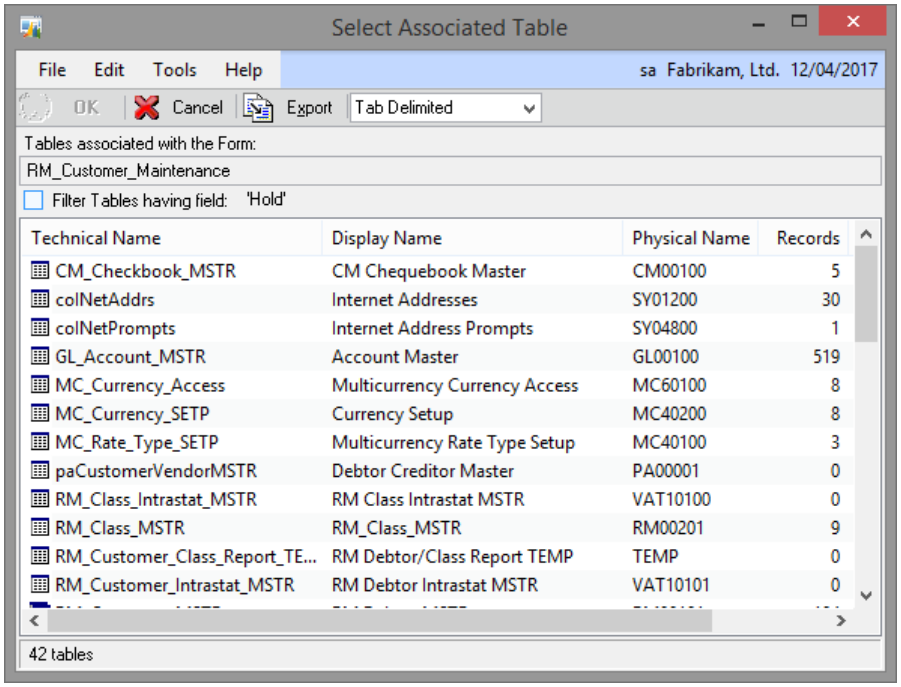
When this checkbox is selected for the Forms, Windows & Fields Resource Type, the Resource Information window will automatically display the details for the currently selected Form, Window and Field.

The Show currently selected Window and Field information feature only works for windows opened while the Resource Information window is open. So open the Resource Information window before opening the windows you want information about.

Associated Tables Button

This button is available when the Resource Information window is in Forms, Windows & Fields mode. It will display a list of tables associated with the currently selected form.

If a field is selected on the Resource Information window, you will have the option to filter the list of tables to only tables having the specified field. If the field is not available in any tables, this option will be disabled.

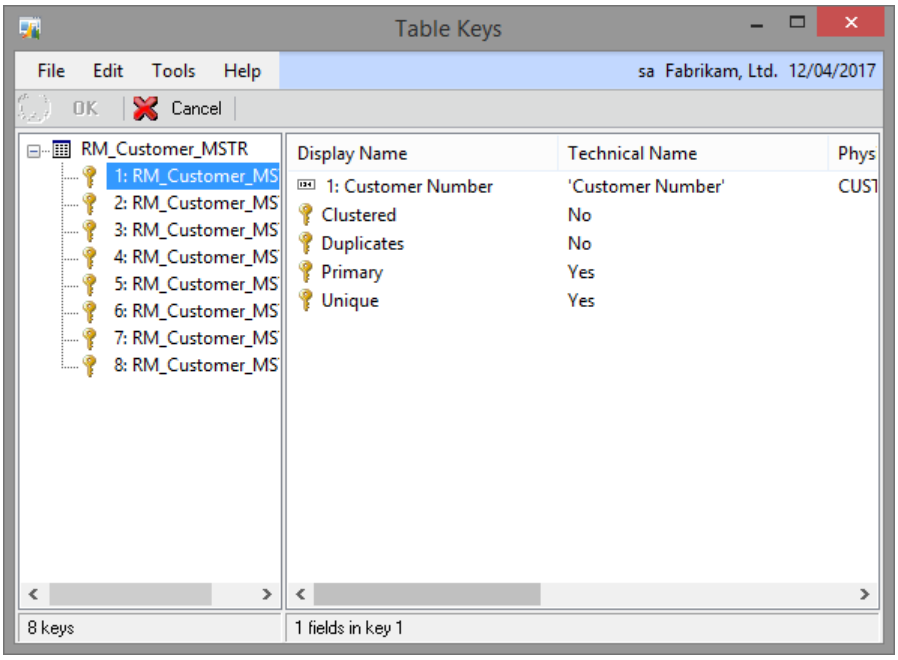


This linked table for the window is highlighted with different icon and a flag in the Linked column in the display.

Selecting a table from this window will change the Resource Information window into Table & Field mode and display the details of the selected table. If filtering on a field, the field will also be selected.

Display Keys Button

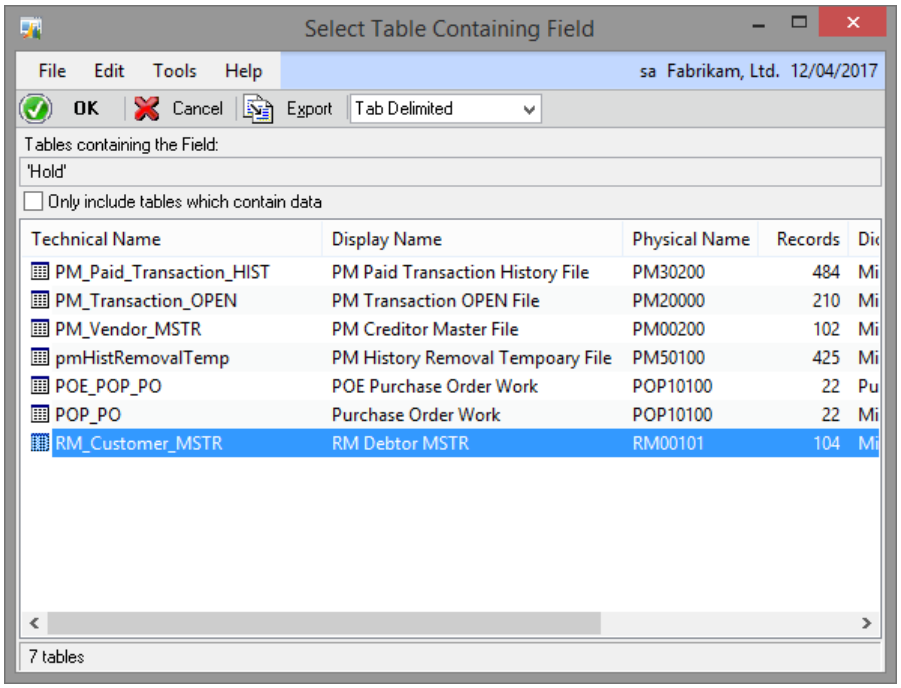
This button is available when the Resource Information window is in Tables & Fields modes. It will display a list of keys (indexes) for the currently selected table. The fields for the key and the key options are displayed.



Selecting a field from this window will display the details of the selected field.

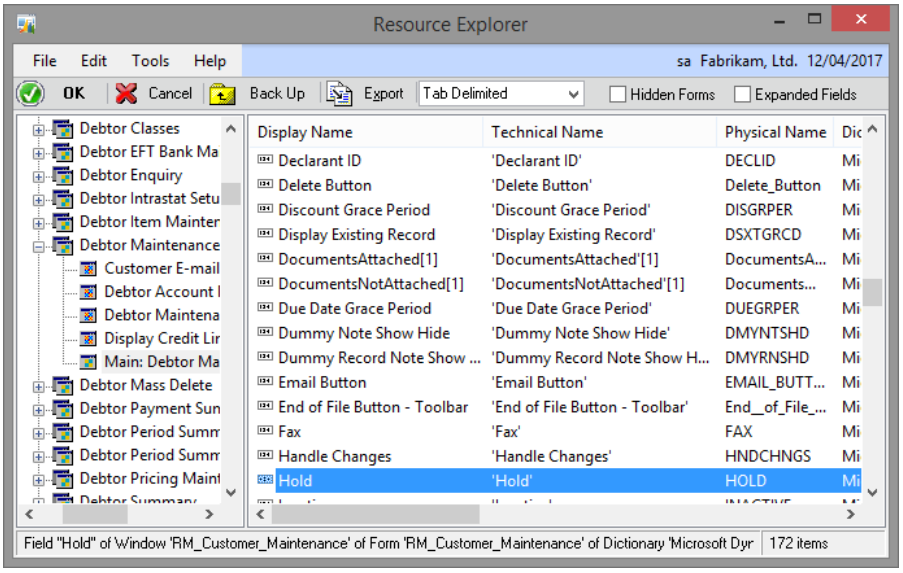
Tables Containing Field Button

This button is available when the Resource Information window is in Forms, Windows & Fields and Tables & Fields modes. It will display a list of tables which contain the currently selected field. You have the option to select to only include tables which contain data.



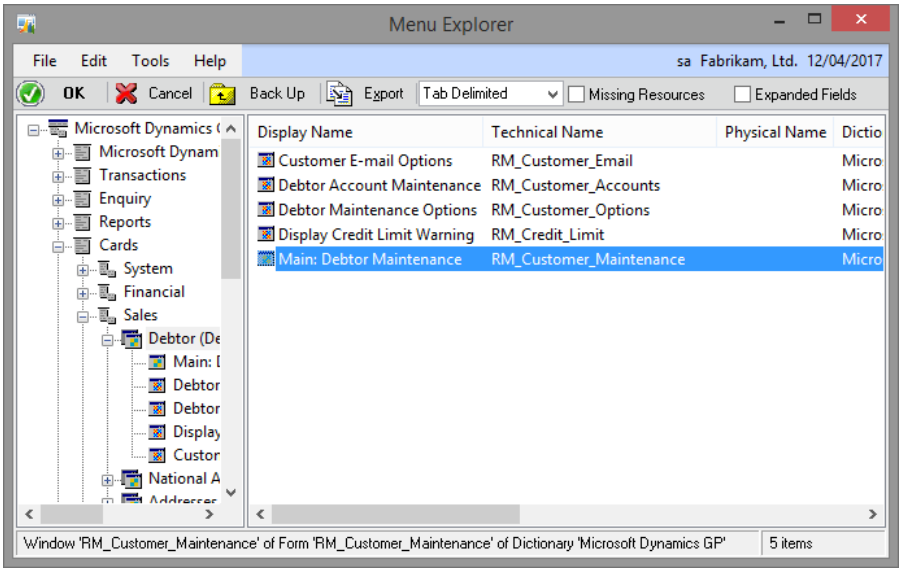
Selecting a table from this window will change the Resource Information window into Table & Field mode and display the details of the selected table and field.

When in Form, Window & Field mode, you can use the lookup button to select a form, window or field resource. Once clicked the Resource Explorer window will open.

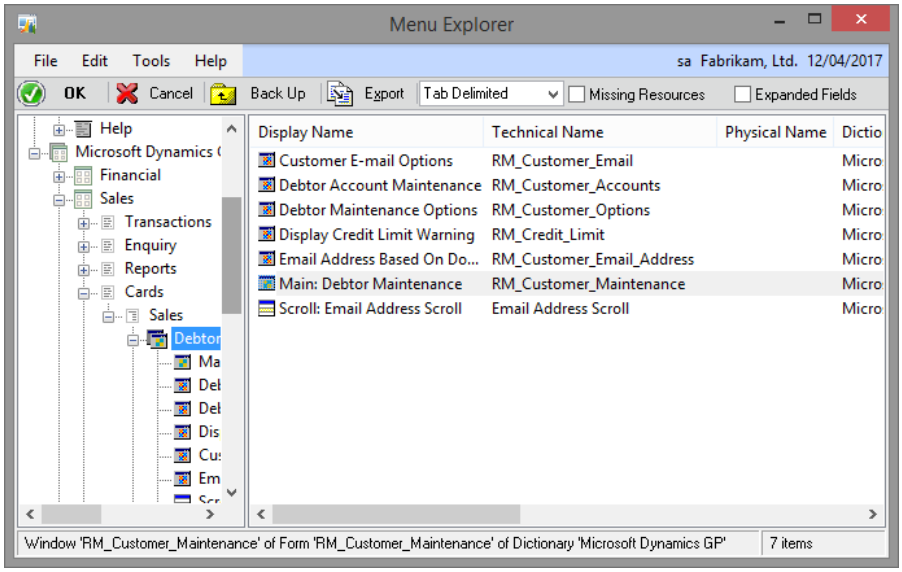


To insert a form name or window name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

You can also use the menu lookup button  to select a form, window or field resource based on the menu navigation model. Once clicked the Menu Explorer window will open.

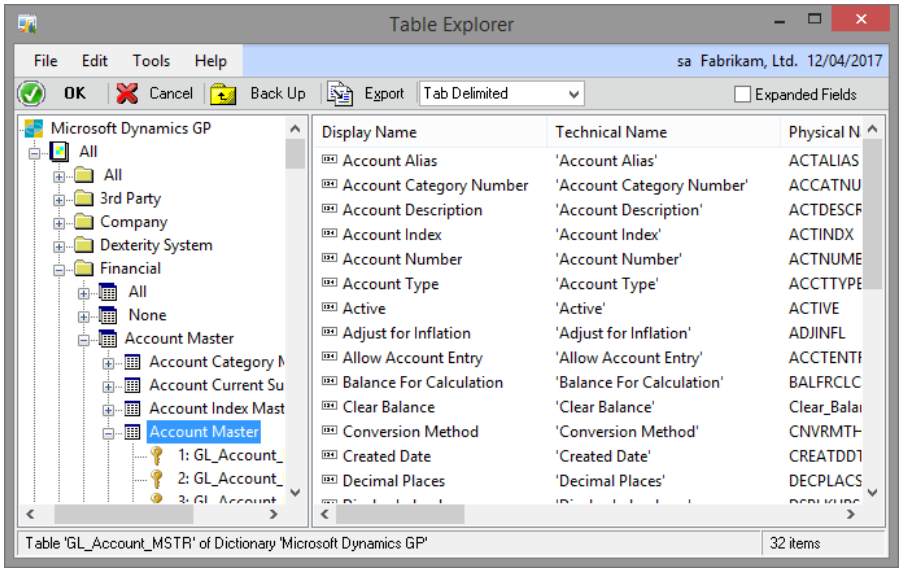


The Menu Explorer window has the option to navigate via the application menus (top of left pane) or by the Area Pages (bottom of left pane).



To insert a form name or window name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

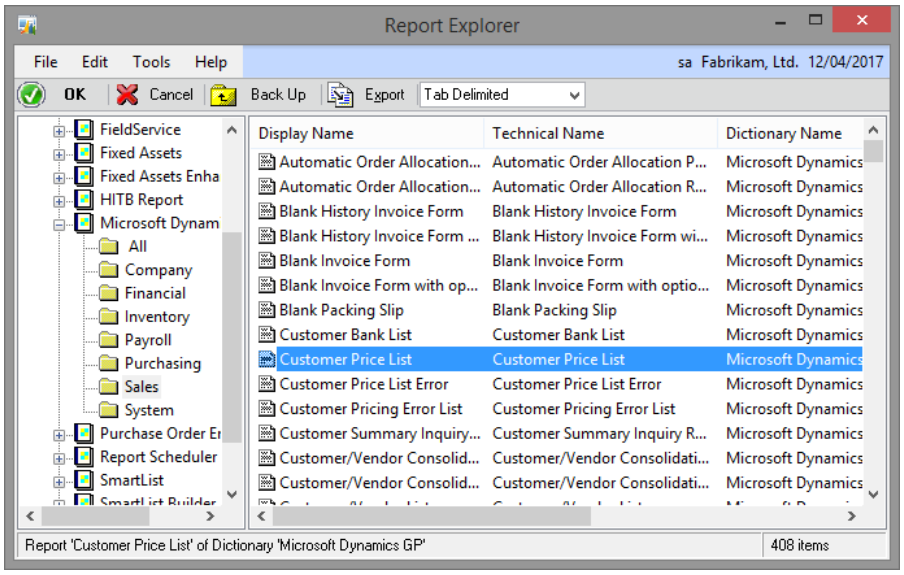
When in Table & Field mode, you can use the lookup button to select a table or field resource. Once clicked the Table Explorer window will open.



To insert a table name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

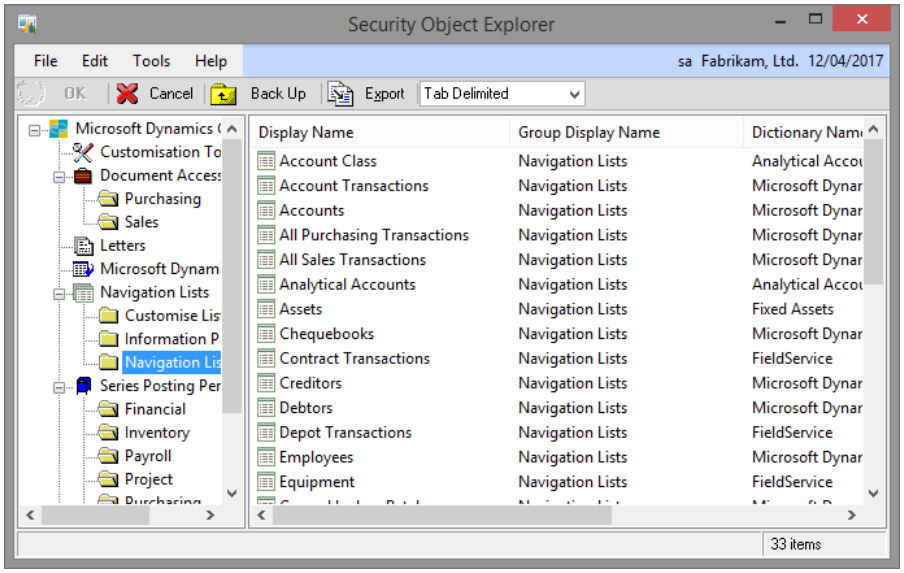
The Table Explorer window can also display the key (index) information for a table. Expand the table node in the tree to display the keys; selecting an individual key will display the key fields and the key options.

When in Report mode, you can use the lookup button to select a report resource. Once clicked the Report Explorer window will open.



To insert a report name, select the resource in the right hand pane and click OK. Custom Reports are shown with a different icon in the right hand pane.

When in Security Object mode, you can use the lookup button to select a security object. Once clicked the Security Object Explorer window will open.



To insert a security object, select the desired security object in the right hand pane and click OK. Security objects from other 3rd party products will show as Unknown Objects.

Below is a description of the individual fields on the Explorer windows:

OK Button
This button will return the selected resource and close the window.

Cancel Button
This button will close the window without making a selection.

Back Up Button
This button will change the current selection to its parent on the tree.

Export Button
This button will allow the resources displayed in the list view to be exported to a file or directly to an email. The default email settings can be set up in the Administrator Settings window.

Export Mode
Use this drop down list to select the format for the exported file. The file can be exported as Tab Delimited, Comma Delimited or as a HTML Table.

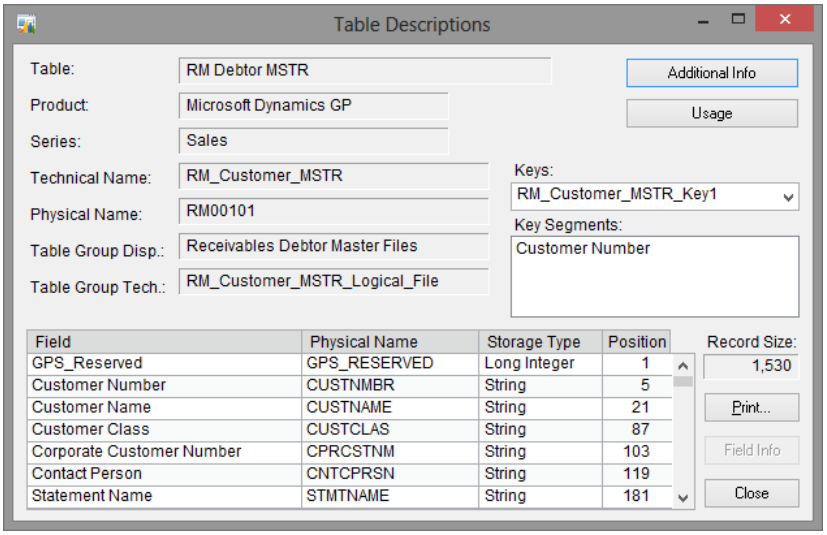
Hidden Forms
Use this check box to show forms which are normally hidden from the security system.

Missing Resources
Use this check box to show menu items which point to external or missing resources.

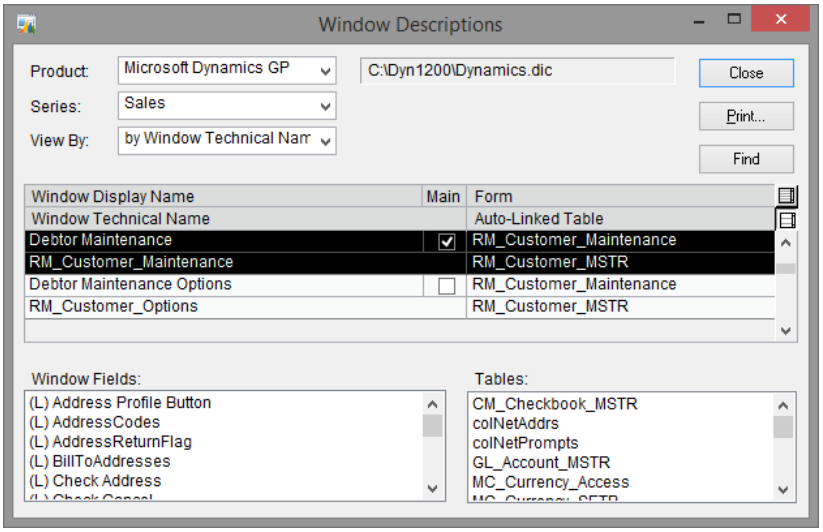
Expanded Fields
Use this check box to expand composite and array fields into the component parts.

The Resource Explorer windows which have two panes are Splitter enabled which allows the ratio between the left and right hand panes to be adjusted. When running on the Web Client, the splitter functionality is disabled..

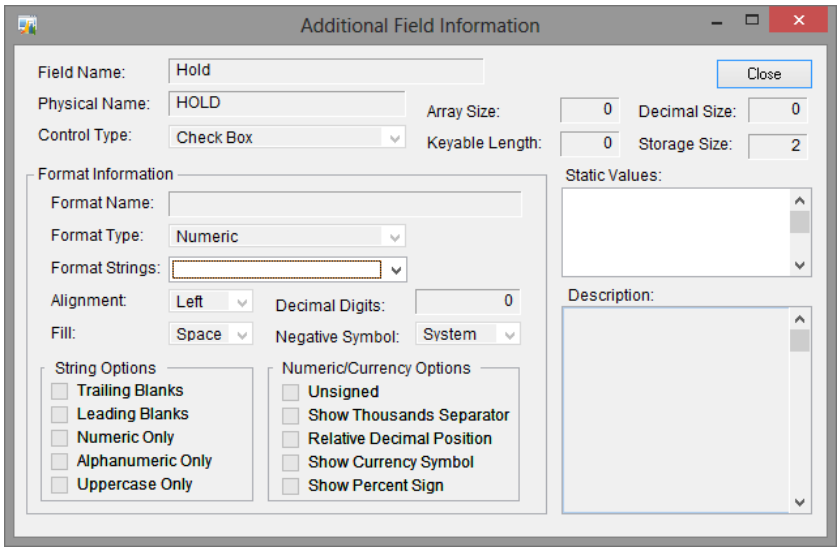
On the Resource Information window, you can click the Open Button or the Technical Name hyperlink to open the current resource. If the resource is a form or report, it will open. If the resource is a table, the standard Table Descriptions window will open.



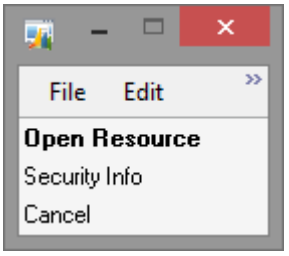
If you click Window Technical Name hyperlink, the standard Window Descriptions window will open.



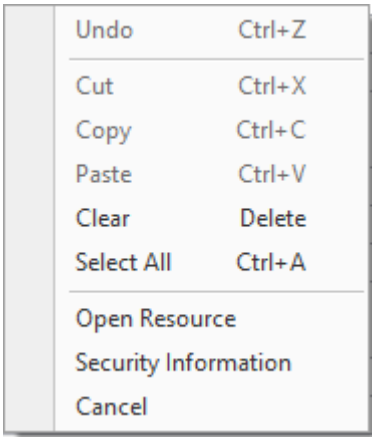
If you click Field Technical Name hyperlink, the standard Field Information window will open.



The Resource Information window is Right click enabled. If you right mouse click on any of the fields you can select Open Resource (same as Open Button), Security Info (same as Security Button) or Cancel from the context sensitive menu. The Security Info option will only be available if the current user has security access to the security windows under Tools >> Setup >> System.



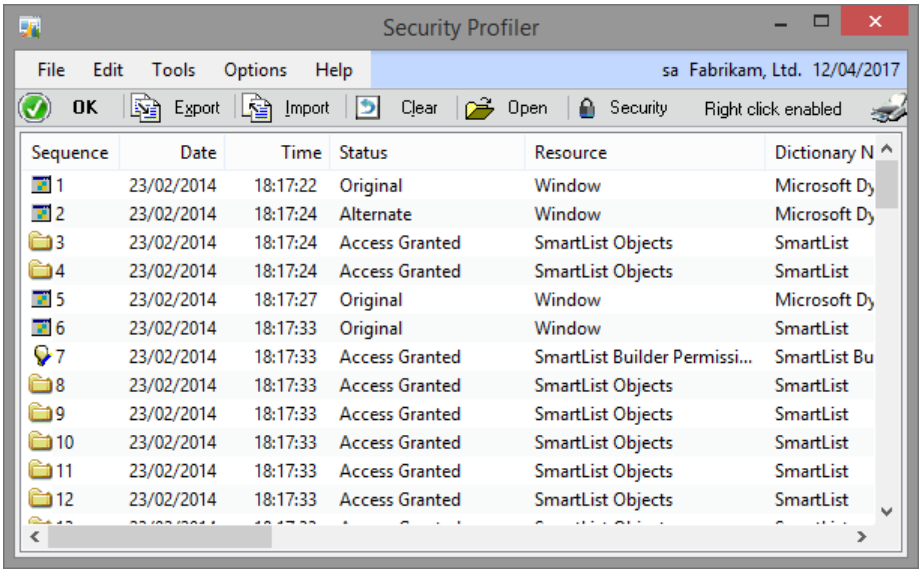
or



Security Profiler

You can open the Security Profiler window by selecting Security Profiler from the Options button drop list on the main window.

After it has been opened, the Security Profiler window will monitor all application-level security requests and display the results.



Sequence	Date	Time	Status	Resource	Dictionary N
1	23/02/2014	18:17:22	Original	Window	Microsoft Dy
2	23/02/2014	18:17:24	Alternate	Window	Microsoft Dy
3	23/02/2014	18:17:24	Access Granted	SmartList Objects	SmartList
4	23/02/2014	18:17:24	Access Granted	SmartList Objects	SmartList
5	23/02/2014	18:17:27	Original	Window	Microsoft Dy
6	23/02/2014	18:17:33	Original	Window	SmartList
7	23/02/2014	18:17:33	Access Granted	SmartList Builder Permissi...	SmartList Bu
8	23/02/2014	18:17:33	Access Granted	SmartList Objects	SmartList
9	23/02/2014	18:17:33	Access Granted	SmartList Objects	SmartList
10	23/02/2014	18:17:33	Access Granted	SmartList Objects	SmartList
11	23/02/2014	18:17:33	Access Granted	SmartList Objects	SmartList
12	23/02/2014	18:17:33	Access Granted	SmartList Objects	SmartList

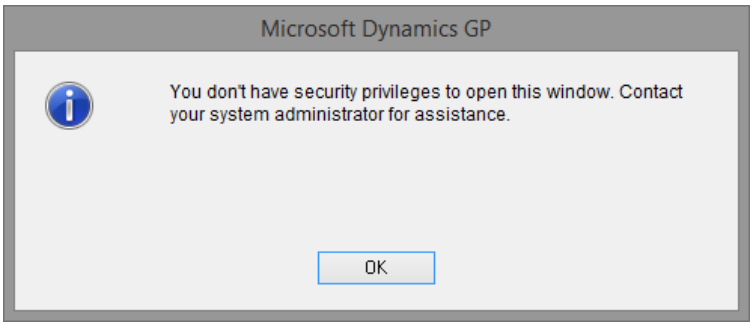
Whenever a form or report is opened, the application-level security is checked to confirm that the current user has access. Security is also checked to find out whether a customized version (modified, alternate or modified alternate) of the form or report is to be used.

When a report is opened, access is checked for all of the tables linked to the report. To be able to print the report, access must be permitted for the report itself and all the tables linked to the report.

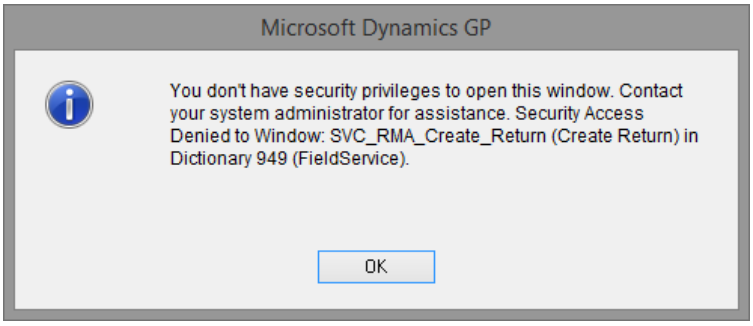
The Security Profiler will also track access to non-dictionary resource Security Objects, such as Customization Tools, Document Access, Letters, Microsoft Dynamics GP Import, Navigation Lists, Series Posting Permissions, and SmartList Objects. If the products are installed, the following objects are also supported, SmartList Builder Permissions and Extender Resources. Security objects from other 3rd party products will show as Unknown Objects.

The Security Profiler window displays each of the queries to the application-level security system and displays the results with all the relevant details of the resources involved.

The Security Profiler window can be used to identify which form or report is causing unexpected security privileges or access denied errors. Just open the Security Profiler and then perform the action in Microsoft Dynamics GP that causes the error to appear. The details of the resource causing the error will be displayed.



By default, the Support Debugging Tool will append additional details to the dialog to identify the resource. This functionality can be disabled from the Administrator Settings window, if desired.



The Security Profiler window is monitoring only application-level security. It will not display security issues caused by Windows security or SQL Server security.

Below is a description of the individual fields on the window:

OK Button

This button will close the Security Profiler window.

Export Button

This button will allow a Security Profiler log to be exported to a file or directly to an email. The default email settings can be set up in the Administrator Settings window. This allows a user to provide all of the details of a security issue to the administrator for their analysis.

Import Button

This button can be used to import a previously exported Security Profiler log. This allows an administrator to view a log of security issues provided by a user.

Clear Button

This button can be used to clear the current contents of the Security Profiler window.

Open Button

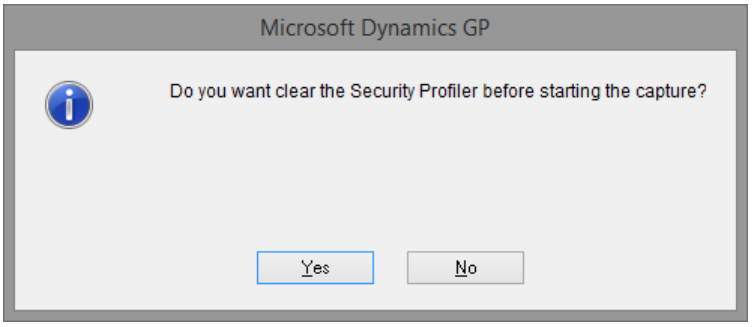
This button will open the selected form or report resource.

Reports opened in this way will not have any options or restrictions applied and might contain unpredictable results. If the report uses a temporary table, this table will contain no data. Opening forms and reports from this window is only for testing purposes.

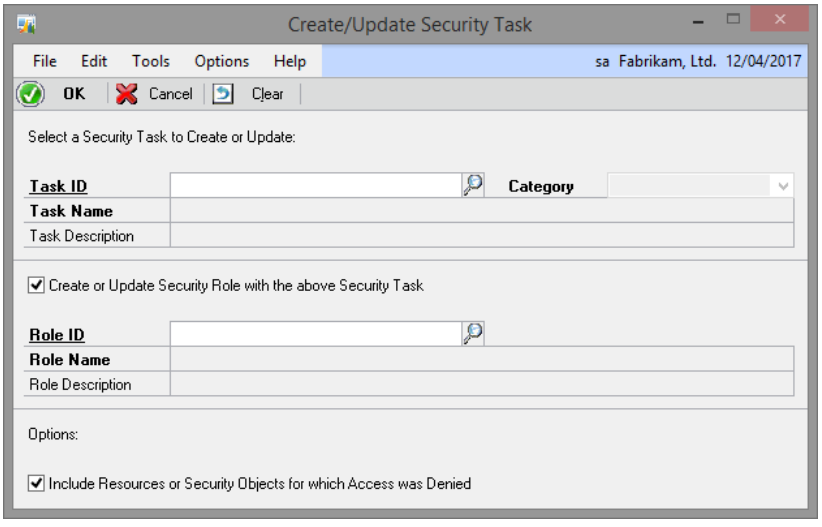
Security Button Drop List

This button Drop List has the option to open the Security Information window for the selected resource. See sections below for more information. The Security Button will only be available if the current user has security access to the Security Information window.

If the current user has access to the Security Task Setup window, the option to Start Capture of Resources and Security Objects will be available. This option will offer to clear the Security Profiler if it is not empty:



If the current user has started the capture of Resources and Security Objects, the option to Stop Capture and create/update Security Task will be available. When this option is selected it will open the Create/Update Security Task window.



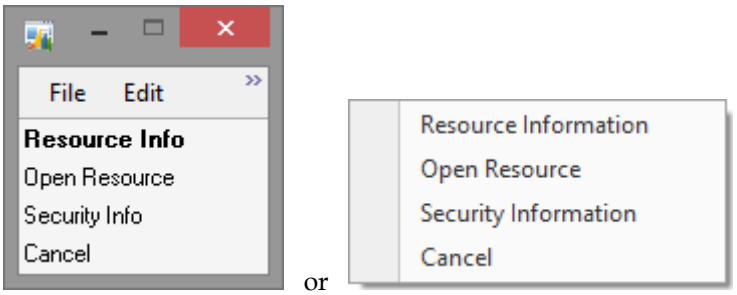
This window can be used to create a new Security Task or update an existing Security Task with the items listed in the Security Profiler. If the user has access to the Security Role Setup window, the option to create a new Security Role or update an existing Security Role with the Security Task ID will be available.

Use the options to capture Resources and Security Objects and then create or update a Security Task based on the captured items to quickly build Security Tasks for specific activities within Microsoft Dynamics GP.

Print Button

This button will allow a report of the contents of the Security Profiler window to be printed.

The Security Profiler window is Right click enabled. If you right mouse click on an item in the list you can select Resource Info (same as double click), Open Resource (same as Open Button), Security Info (same as Security Button) or Cancel from the context sensitive menu. The Security Info option will only be available if the current user has security access to the security windows under Tools >> Setup >> System.



The Security Profiler window can be configured to open automatically when there is a security issue. This option is controlled from the Administrator Settings window.

The Security Profiler window has an Options Menu which can be used to Refresh Application Navigation. This option can be used by a user to update the application's navigation menus to reflect changes made to security without having to exit and re-launch the application.

Security Information

You can open the Security Information window by selecting Security Information from the Options button drop list on the main window. Once opened, you can use the drop down menu on top of the left pane to select a resource. You may select a Form (by Dictionary or By Menu) as well as a Table or Report resource, or a Security Object.

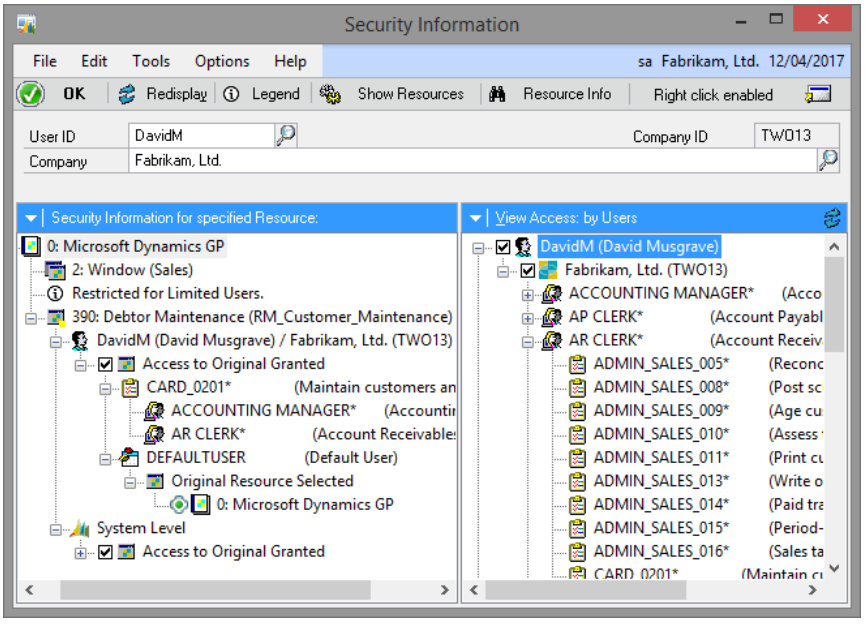
You can also open the Security Information window from the Resource Information window or the Security Profiler window. From these windows use the Security Button or the Security Info option from the local context (right click) menu to show security information for the selected resource.

For a user to have access to the Security Information window they must have access to the security windows under Tools >> Setup >> System.

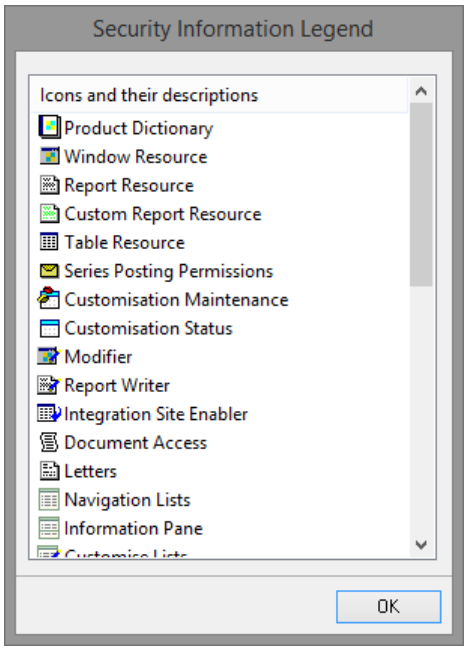
The user must have access to User Security Setup, Security Task Setup, Security Role Setup and Alternate/Modified Forms and Reports.

The Security Information window is designed to display the security settings for the selected resource for a particular user and company combination. Once the information is displayed the administrator can use the Go To Button or double click to open the appropriate security administration window to make changes if necessary.

Below is an example of the Security Information window. It shows the security settings for the user including the security tasks that belong to security roles assigned to that user. Also shown is the alternate/modified form and report ID to show which version of a resource the user has access to. Under the System Level node, all security tasks, security roles and alternate/modified form and report IDs which reference the selected resource are displayed.



If you click the Legend Button the following window will be displayed to show the meanings of the different icons used.



The tree in the left hand pane is used to display the security status for the currently selected user and company for the selected security resource. The first 3 nodes of the tree describe the product dictionary, resource type (and series) and resource by Display and Technical Name.

If a resource is not available on the Web Client or not available to Limited Users, this will be displayed on an information node on the tree. Also a Limited User will be highlighted with a yellow dot on the icon and Limited User in the description.

The next section is the User node which shows if the user has access to the current resource and which Security Tasks and Security Roles provided that access. If the resource is a Form or Report, the Alternate Modified Form and Report ID will be shown to define which version of the resources access is granted to.

The third section is the System node which shows all Security Tasks and Security Roles which reference the current resource and all Alternate Modified Forms and Report IDs that reference the current resource. Security Tasks, Security Roles and Alternate Modified Forms and Reports IDs in this view will have a green or red indicator to show whether the current user and company has access.

Below is a description of the individual fields on the window:

User ID

This is the User ID for which security is being checked.

Company

This is the company for which security is being checked.

OK Button
This button will close the Security Information window.

Redisplay Button
This button will re-populate the security information tree. Use this button after making security changes to see the new updated security.

Legend Button
This button will open the Security Information Legend window.

Show Resources Button
This button will open the Security Information Resources window.

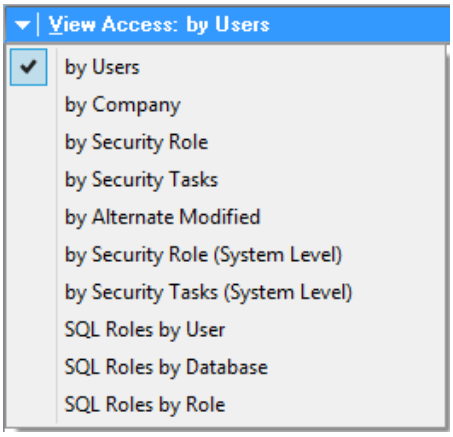
Resource Info Button
This button will open the Resource Information window.

Go To Button
This button allows the user to open a system security window.

You can double click on the User ID/Company node to open the User Security Setup window; a Security Task ID to open the Security Task Setup window; a Security Role ID to open the Security Role Setup window; and an Alternate/Modified Forms and Reports ID to open the Alternate/Modified Forms and Reports window.

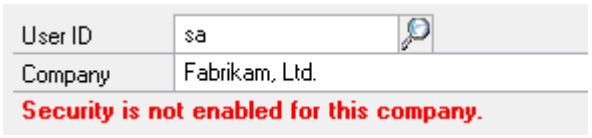
The right hand pane on the Security Information window displays a number of different views into the company access and security information. Use the View Access button drop down list to change view. When changing views the currently selected object will remain selected if possible. This pane can be used even when no resource is selected before opening the Security Information window.

Below are the views available.



These views will provide a visual representation of the relationships between Security Tasks, Security Roles, Alternate Modified Forms and Report IDs, Users and Companies.

The Security Information window will highlight when security is not activated for the selected company. This can be enabled from Company Setup window (Microsoft Dynamics GP >> Tools >> Setup >> Company >> Company).



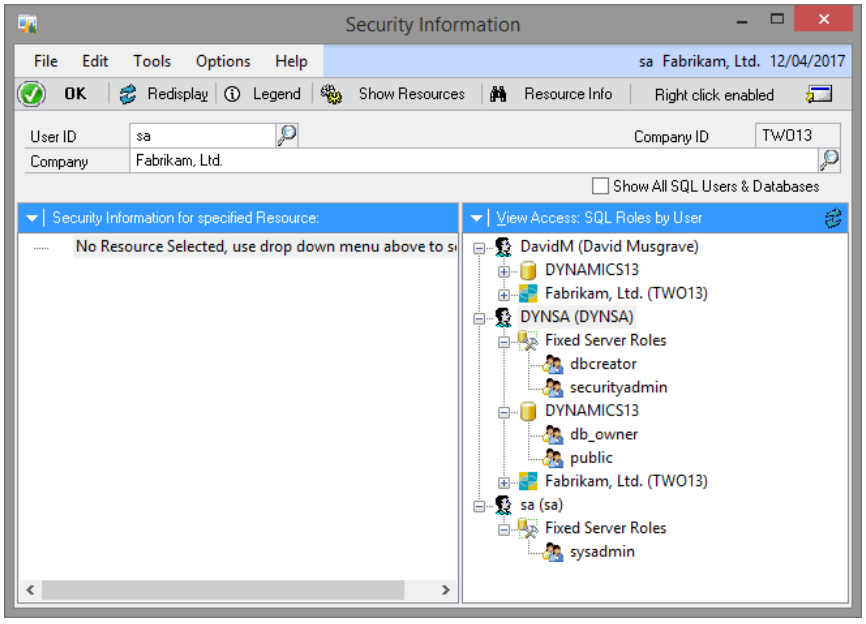
The Security Information window is Splitter enabled which allows the ratio between the left and right hand panes to be adjusted. When running on the Web Client, the splitter functionality is disabled..

Security Information SQL Role Views

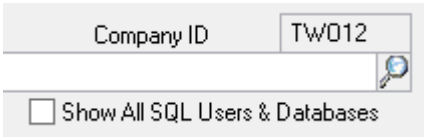
The Security Information window can also be used to show the SQL Server Roles assigned to users at the SQL Server level as well as for each database. There are three views available to view the data by Users, by Database and by Role.

The SQL Role information is read from the SQL Server the first time one of the three SQL Role views is selected. On a large system, there might be a small delay while the data is read from the SQL Server. To force the data to be read again, close and re-open the Security Information window.

Below is an example screenshot.



The data shown in the three views is restricted to only include Dynamics GP users and databases by default. To show all users and database, select the Show All SQL Users & Databases checkbox.



Once the option has been selected, the view will be refreshed to include the additional data for non-Dynamics GP users and databases.

Security Information Resources

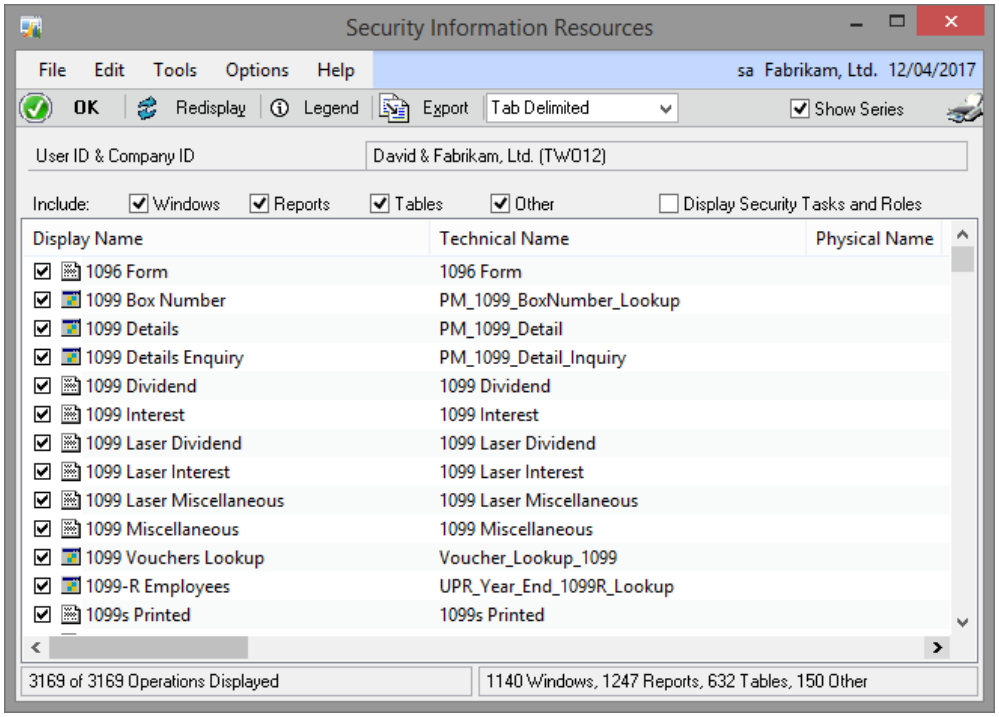
When the Show Resources Button is clicked, the Security Information Resources window will open.

This window will display the resources associated with the currently selected User ID/Company combination, Security Role ID, Security Task ID or Alternate/Modified Forms and Report ID in the right hand pane of the Security Information window. Changing the selection will cause the window to refresh.

You can use the check boxes to decide which resource types (Forms, Reports, Tables and Other) to include in the displayed resources. These selections can be changed while the window is populating.

The resources displayed are those for which the selected User ID/Company combination, Security Role ID, Security Task ID or Alternate/Modified Forms and Report ID has access to.

If the selected node in the right hand pane of the Security Information window has a User ID and/or Company ID parent node, the system will be able to identify which Alternate/Modified Forms and Report ID to apply and so will display when an alternate and/or modified version of the resources has been selected.



Below is a description of the individual fields on the window:

OK Button
This button will close the Security Information Resources window.

Redisplay Button
This button will re-populate the window. Use this button after making security changes to see the new updated security.

Legend Button
This button will open the Security Information Legend window.

Export Button
This button will allow the resources displayed in the list view to be exported to a file or directly to an email. The default email settings can be set up in the Administrator Settings window.

Export Mode
Use this drop down list to select the format for the exported file. The file can be exported as Tab Delimited, Comma Delimited or as a HTML Table.

Show Series
Use this checkbox if you want the series information included in the resource list.

Display Security Tasks and Roles
Use this checkbox if you want the Security Tasks and Security Roles displayed in the resource list. When this option is selected, multiple lines will be displayed for resources if there are more than one Security Task or Security Role which provides access to the resource.

Print Button
A report of the contents of the resource list can be printed using this button.

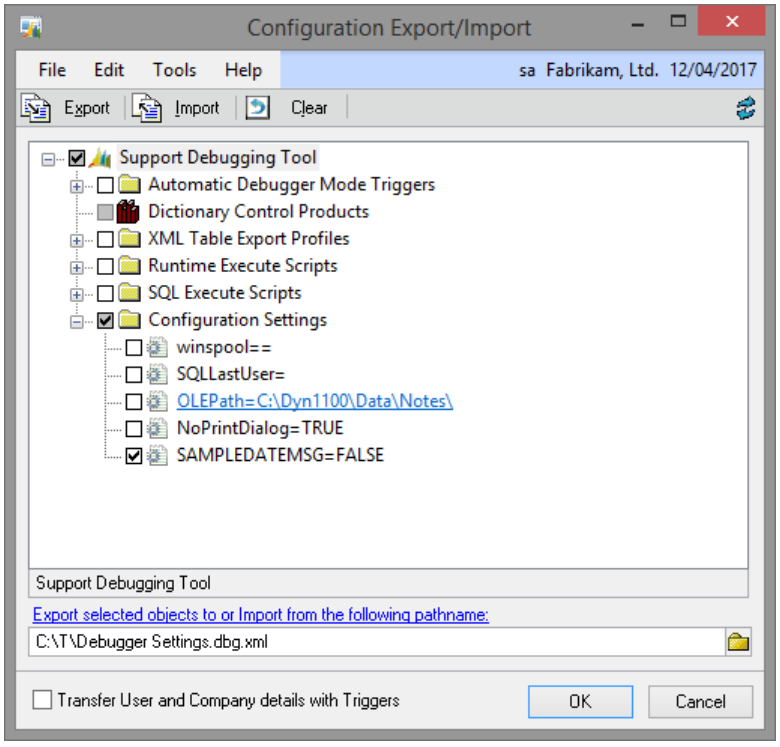
When opening the Security Information window a background process is launched to check if all the dictionary resources and security objects have been added to the syCurrentResources (SY09400) table. If information is found to be missing, the Support Debugging Tool will generate the additional data. The Support Debugging Tool will also add the additional data when the table is cleared using the Clear Data window.

The Security Information window has an Options Menu which can be used to Refresh Resource Information Table. This option can be used by a user to clear and then update the syCurrentResources (SY09400) table without having to use the Clear Data window.

Configuration Export/Import

You can open the Configuration Export/Import window by selecting Configuration Export/Import from the Options button drop list on the main window.

The Configuration Export/Import window can be used to export and import selected Support Debugging Tool settings.



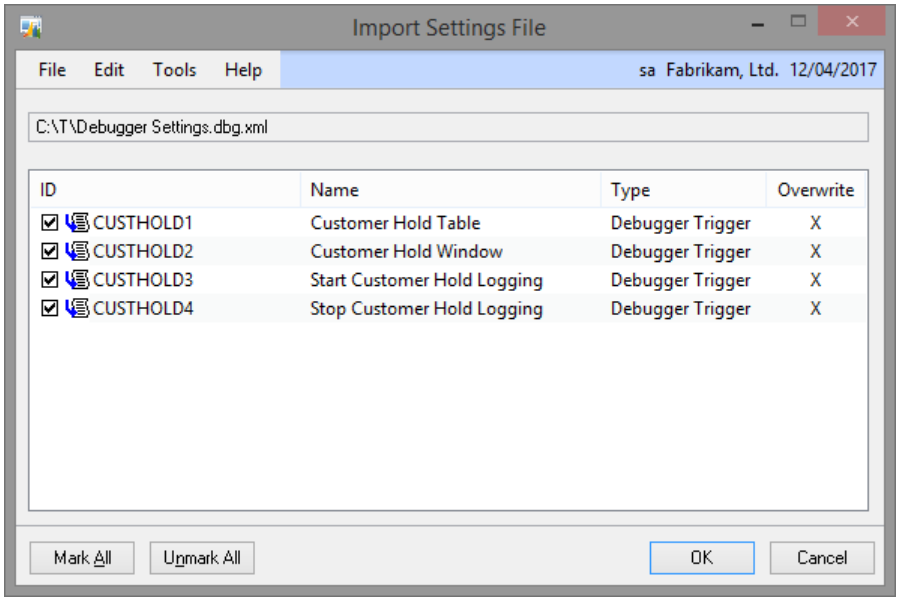
Below is a description of the individual fields on the window:

Export Button

This button will export the selected settings to the file name selected.

Import Button

This button will import the contents of the selected file name. It will open the Import Settings File window to display the contents of the settings file. You can then select the objects that will be imported from the settings file.



Clear Button

This button will clear any selections and reset the File Name and Transfer User and Company Details with Triggers checkbox.

File Name

This is the file name used for exporting and importing. The file should use the extension .dbg.xml.

Transfer User and Company details with Triggers

This checkbox selects whether the user and company selection for triggers is exported when the trigger is exported.

ScreenShot

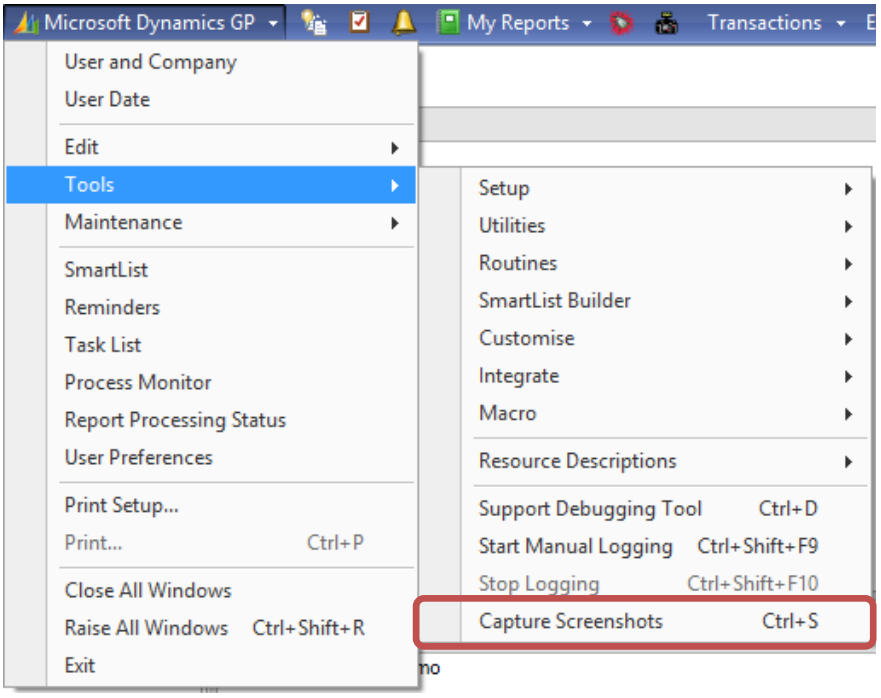
ScreenShot is a tool which can capture screenshots of all the open windows in the application as well as a System Status report and then either email or save the files. ScreenShot is based on code originally developed by eOne Integrated Business Solutions.

Screenshot creates reduced color bitmaps (4 bits per pixel, 16 colors) to ensure that the size of the email is kept to a minimum. It can capture all open windows regardless of whether they are overlaid by other windows.

The System Status report contains information about the system including registration information, current login information, environmental information (such as operating system, database and ODBC versions), product information (including all version and build numbers) and a list of the attached screenshots.

You can open the Support Debugging Tool ScreenShot window by selecting Capture Screenshots from the Options button drop down list on the main window.

You can open it directly from the Tools menu underneath the Microsoft Dynamics GP menu (highlighted below). It also has the keyboard shortcut Ctrl+S assigned to it.

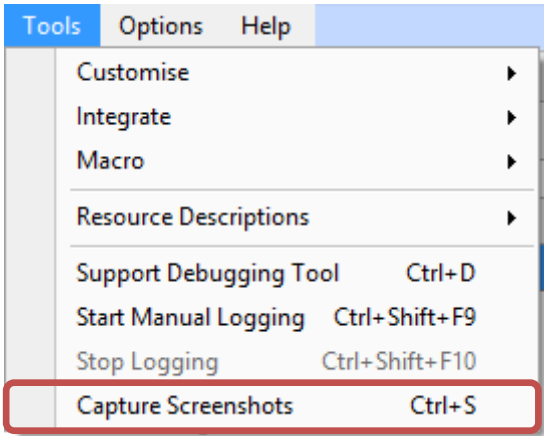


If the Standard Toolbar is displayed, you can launch ScreenShot from the Capture Screenshots button (highlighted below).

You can also use the Capture Screenshots option on Quick Links on the Home Page

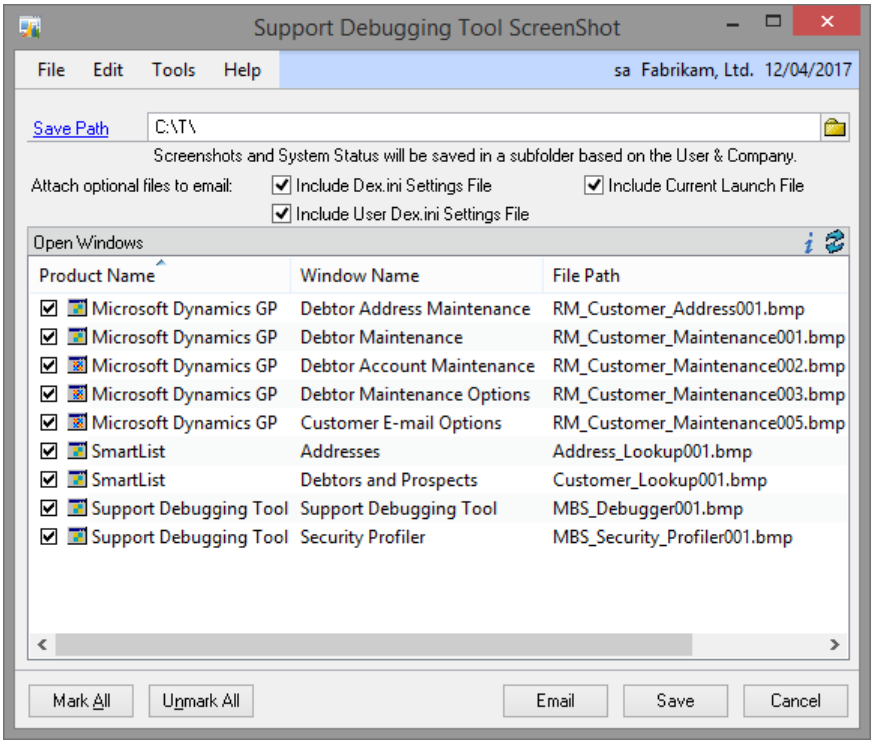
When running on the Web Client, use the Quick Links on the Home Page to open the ScreenShot as the other navigation options are not available.

In addition, the ScreenShot is also found under the Tools menu on each individual window of Microsoft Dynamics GP (highlighted below).



You may need to press and release the Alt key on the keyboard to allow the window menu bar to activate before the shortcut keys work.

Once ScreenShot is activated, the following window will be displayed.



Below is a description of the individual fields on the window:

Save Path

This is the root path that will be used when saving screenshots. The actual path used will be a subfolder based on the user ID and company ID code.

Include Dex.ini Settings File

This checkbox tells ScreenShot whether to include the Global level Dex.ini settings file as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Include User Dex.ini Settings File

This checkbox tells ScreenShot whether to include the User level Dex.ini settings file as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Include Current Launch File

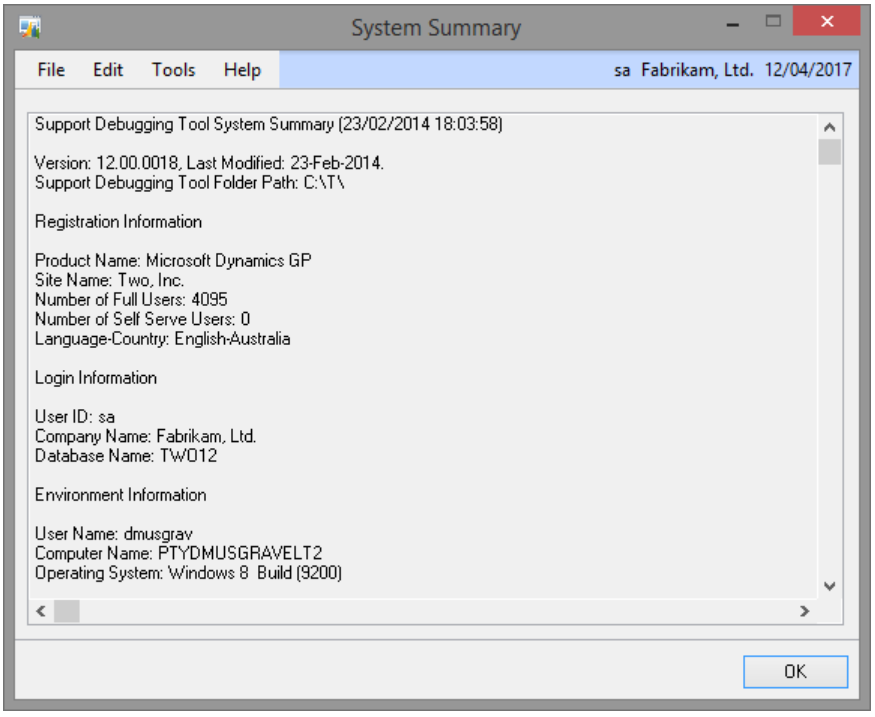
This checkbox tells ScreenShot whether to include the launch file, usually Dynamics.set, as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Open Windows

This is a list of windows currently open on the system. It is automatically updated when a form is open or closed. If you open a secondary window on a form, you can refresh the list with the Refresh Button. You can use the checkboxes to select which screenshots should be included.

Info Button

This button can be used to preview the System Status report. You can use Ctrl-A to select the contents of the report and then Ctrl-C to copy it to the clipboard.



Refresh Button

This button will refresh the window list with the currently open windows.

Mark All Button

This button will select all windows to be emailed or saved. This button will be disabled when running on the Web Client.

Unmark All Button

This button will de-select all windows so that individual windows can be selected. This button will be disabled when running on the Web Client.

Email Button

This button will create an email with the selected screenshots and System Status report attached. The System Status will also be included as the body of the email. All that the user needs to do is add a recipient and click Send. The default email settings can be set up in the Administrator Settings window.

Save Button

This button will save the selected screenshots and System Status report to a folder based on the Save Path and the current user ID and company ID code.

Cancel Button

This button will close ScreenShot.

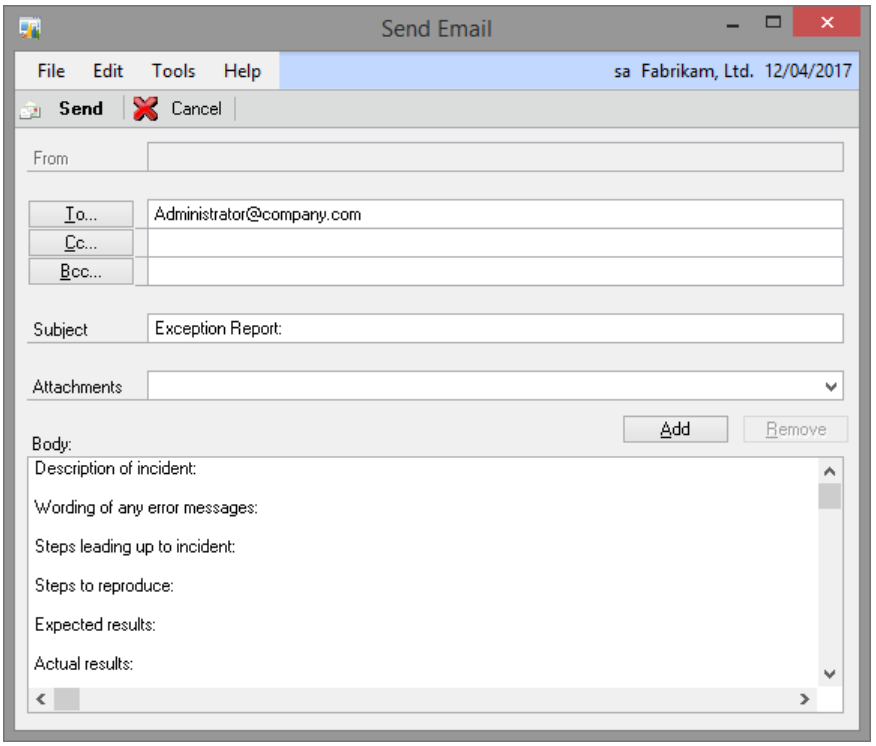
When running on the Web Client, ScreenShot is unable to create the bitmap images and so this functionality is disabled..

Send Email

You can open the Send Email window by selecting Send Email from the Options button drop list on the main window.

The Send Email window can be used to create and send email messages from inside the Microsoft Dynamics GP application. This window will also appear to the user when other features in the Support Debugging Tool are configured to send emails and the option to Preview emails is enabled in the Administrator Settings window.

The default email settings can be set up in the Administrator Settings window. This includes the Administrator Email address to use in the To address and the Default Subject and Default Body Text.



Email addresses can be in the following formats and multiple addresses should be separated by a semi-colon (;):

- name@domain.com
- Full Name<name@domain.com>
- Full Name (when in Microsoft Outlook mode only)

Below is a description of the individual fields on the window:

From Field

This is a single email address used as the sender's email when sending via SMTP mode. The default value is set up in the Administrator Settings window as the Sender's Email.

To Field

This is the list of email addresses to be used as the To value when sending the email. The To Button is available when a MAPI compliant email client is installed and allows the selection of addresses from an address book. The default value is set up in the Administrator Settings window as the Administrator Email.

Cc Field

This is the list of email addresses to be used as the Cc (Carbon Copy) value when sending the email. The Cc Button is available when a MAPI compliant email client is installed and allows the selection of addresses from an address book.

Bcc Field

This is the list of email addresses to be used as the Bcc (Blind Carbon Copy) value when sending the email. The Bcc Button is available when a MAPI compliant email client is installed and allows the selection of addresses from an address book.

Subject

This is the Subject line to be used when sending the email. The default value is set up in the Administrator Settings window as the Default Subject.

Attachments

This is a drop down list containing the paths to the files to be attached when sending the email.

Add Button

This button opens a dialog to select a file to be added to the list of attachments.

Remove Button

This button removes the currently selected attachment from the list.

Body

This is the Body text to be used when sending the email. The default template can be set up in the Administrator Settings window as the Default Body Text.

Send Button

This button will process the email and send it. The transport protocols and other email settings can be set up in the Administrator Settings window.

Cancel Button

This button will abort the email and close the window.

When the Send Email window is manually opened, it behaves as though Preview and Auto Send options are enabled in the Administrator Settings window. This is to ensure that the Send Email window is the only user interface seen when manually sending emails.

Chapter 4: Advanced Mode Features

This chapter includes the following sections:

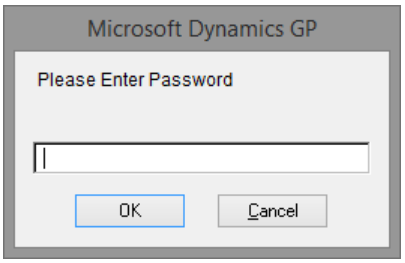
- *Advanced Mode Access*
- *Automatic Debugger Mode*
- *Dictionary Control*
- *XML Table Export*
- *XML Table Import*
- *Runtime Execute*
- *SQL Execute*
- *Configuration Maintenance*
- *Administrator Settings*
- *Dex.ini Configuration*

Advanced Mode Access

To be able to access the Advanced Mode features of the Support Debugging Tool you will need to mark the Enable Debugger Advanced Mode Features checkbox of the Dex.ini Settings window. This will add the additional items to the menus and the Options button drop list.

Once the Advanced Mode features are available on the menus, the ability to use them will be restricted. This restriction is based on the current Microsoft Dynamics GP User ID having either the SQL Server sysadmin fixed server role or membership of the db_owner role on the system database (DYNAMICS) and the current company database.

If the Microsoft Dynamics GP system password is configured, you will need to enter this password before the window will open.



Advanced Mode features are protected because they should be used only by Microsoft Dynamics GP system administrators, partner consultants or Microsoft support engineers.

Some Advanced Mode features allow direct access to data stored on the SQL Server. Other features can be used to disable functionality of Microsoft Dynamics GP.

Automatic Debugger Mode

The primary function of the Support Debugging Tool is the Automatic Debugger Mode. This mode uses the logging options and Dexterity triggers to log application and SQL activity up to a specific event and exception condition. The Support Debugging Tool can look for multiple issues.

Introduction

The Automatic Debugger Mode of the Support Debugging Tool came about as a result of a specific support incident. The Dynamics support team was assisting a customer with a situation that produced invalid data in a table, but no cause could be replicated. Looking at the customer's data it was verified that there was an incorrect value in the table. No one was able to identify when the previously correct value in the table was being changed to the incorrect value. The Support Debugging Tool was used to monitor the table field in question and log the steps which led up to the field changing to the incorrect value. It was able to identify the situation and provide the exact scripts being executed up to the point the exception occurred. This information allowed the code issue to be identified and fixed.

How To Setup

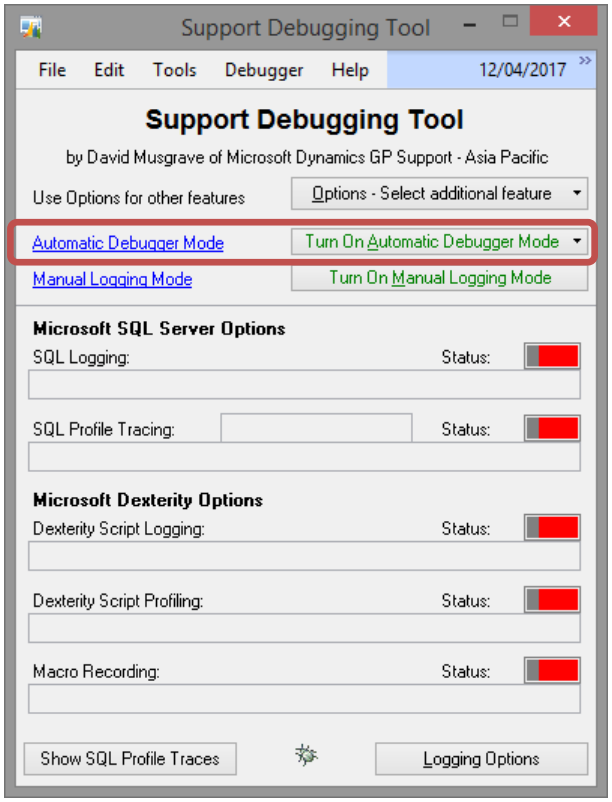
To use Automatic Debugger Mode you must create a trigger ID for each issue or exception condition being monitored. For each trigger ID, an event must be identified which can be used to look for the exception condition. For example, if the exception condition involves data in a table, the trigger event used could be when the table in question is saved. If the exception condition involves a field on a window, the trigger event could be when the field in question is changed.

After the trigger event is selected, a conditional script is written using Dexterity sanScript to check whether the exception condition has actually occurred. Scripts written for this purpose will require the assistance of an experienced Dexterity developer.

Finally, the actions to perform when the exception condition occurs are defined. The trigger ID can be marked to start automatically. When the Start Trigger Automatically on Login checkbox is selected, it is possible to limit the Trigger ID to only automatically start for specified users and/or companies as well as a specified date range.

Registration

When Automatic Debugger Mode is started either manually from the Support Debugging Tool main window or automatically on login, the Support Debugging Tool registers Dexterity triggers based on the trigger IDs being activated. Once the triggers are registered all logging options are activated. The Support Debugging Tool then waits for one of the triggers to fire.

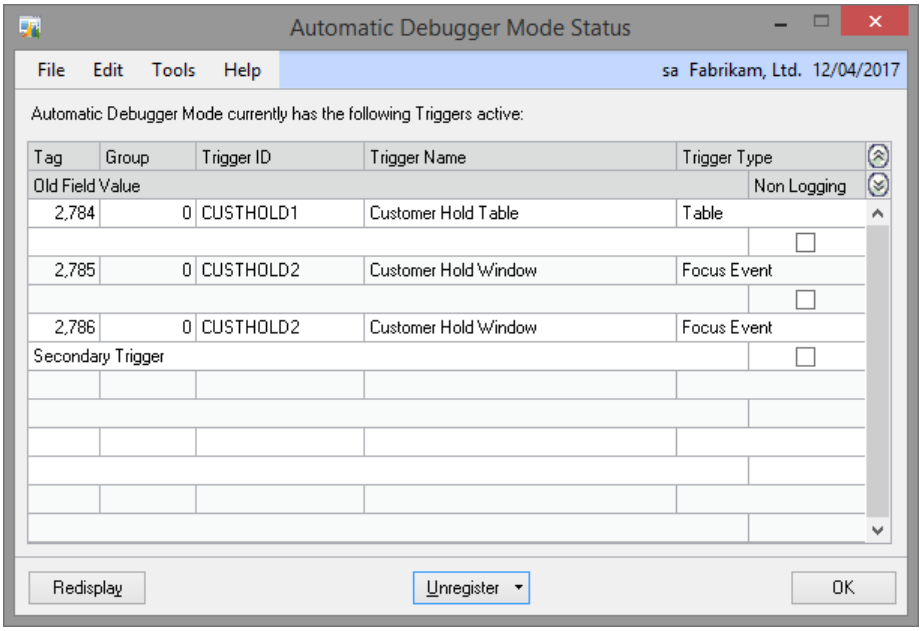


When manually activating the Automatic Debugger Mode, you can select whether to activate:

- just the DEFAULT trigger ID only,
- the logging trigger IDs marked to start automatically only,
- all logging trigger IDs in the system (except those marked as disabled),
- the non-logging trigger IDs marked to start automatically only, or
- all non-logging trigger IDs in the system (except those marked as disabled).

Non-logging triggers are triggers that can be registered to perform actions independently of the normal Automatic Debugger Mode triggers. They will not activate Automatic Debugger Mode and will not start the system logging. Non-logging triggers can be used to store system values prior to other triggers or used to prototype possible changes to fix an issue without the creation of a Dexterity chunk-based trigger.

Clicking on the Automatic Debugger Mode hyperlink will open the Automatic Debugger Mode Status window which displays the Dexterity triggers are currently registered by the Support Debugging Tool. If the trigger needs to store a previous value for a field, it will also be shown on this window.



You can also open the Automatic Debugger Mode Status window by selecting Automatic Debugger Mode Status from the Options button drop list on the main window.

From the Automatic Debugger Mode Status window you can use the Unregister button to unregister single or multiple triggers of either the logging or non-logging type.

Triggering

When an event being monitored occurs and the Dexterity trigger is initiated or "fired" the Support Debugging Tool looks up the trigger ID and runs the associated script to check if the exception condition has actually happened.

If the issue or exception condition is identified to have occurred by the associated script, the Support Debugging Tool will log the results and save the log files as described in the Manual Logging Mode section. The Support Debugging Tool then restarts the logging and continues to wait for the next trigger to fire.

If the actions to export the table record or the entire table were selected, the following files will be created:

- Record_<User>_<Company>_<Date>_<Time>.xml

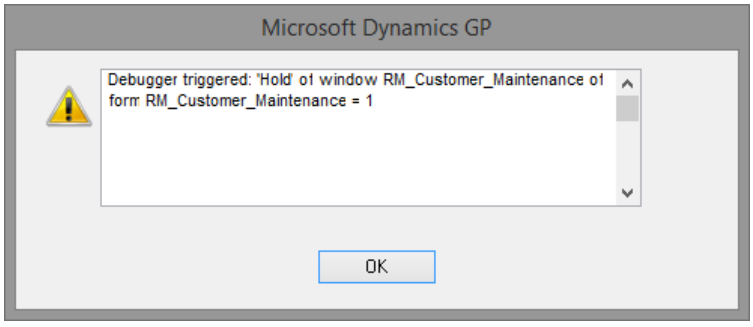
  This file will contain the exported table record.

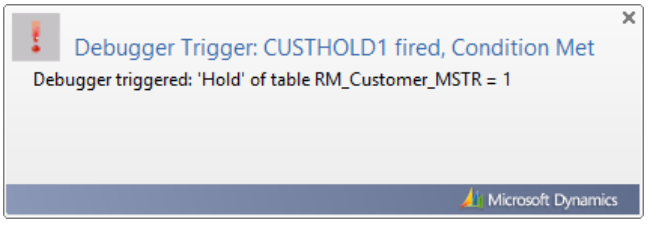
- Table_<User>_<Company>_<Date>_<Time>.xml

  This file will contain the exported records for the entire table.

These export files can be found in the folder where the Support Debugging Tool is storing its data files. The default location is the data subfolder beneath the Microsoft Dynamics GP application folder. The location can be changed from the default path using the Pathname location for Debugger Setup files, exports and logs option on the Dex.ini Settings windows (see section the previous chapter).

If the action to display a message or desktop alert was selected, a dialog and/or alert with the display message specified will be shown.

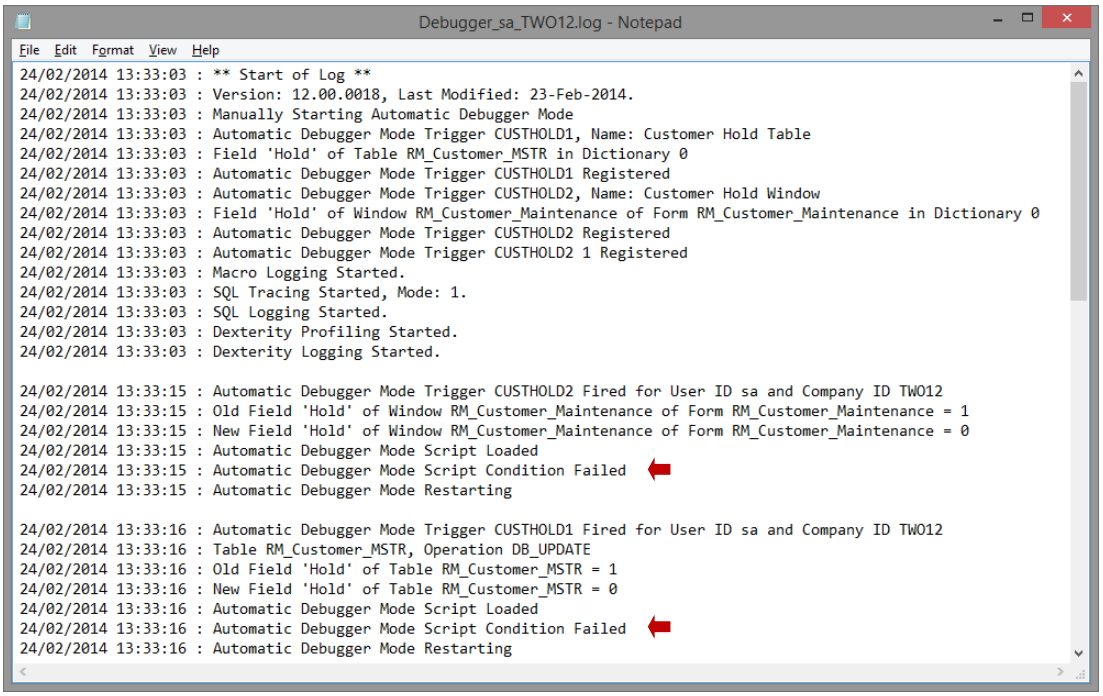




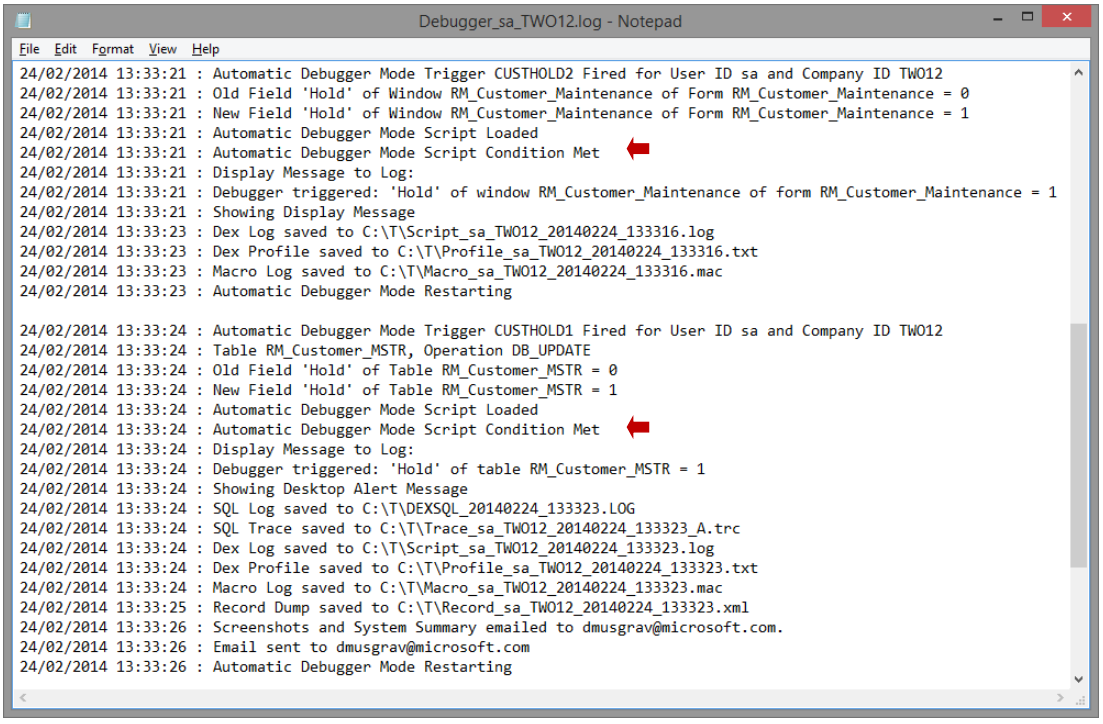
If the exception condition has not occurred, then the Support Debugging Tool resets and continues to wait for the next trigger to fire.

Log File

All actions by the Support Debugging Tool are logged in the Debugger_<User>_<Company>.log file. Below is an example log when the conditions failed.



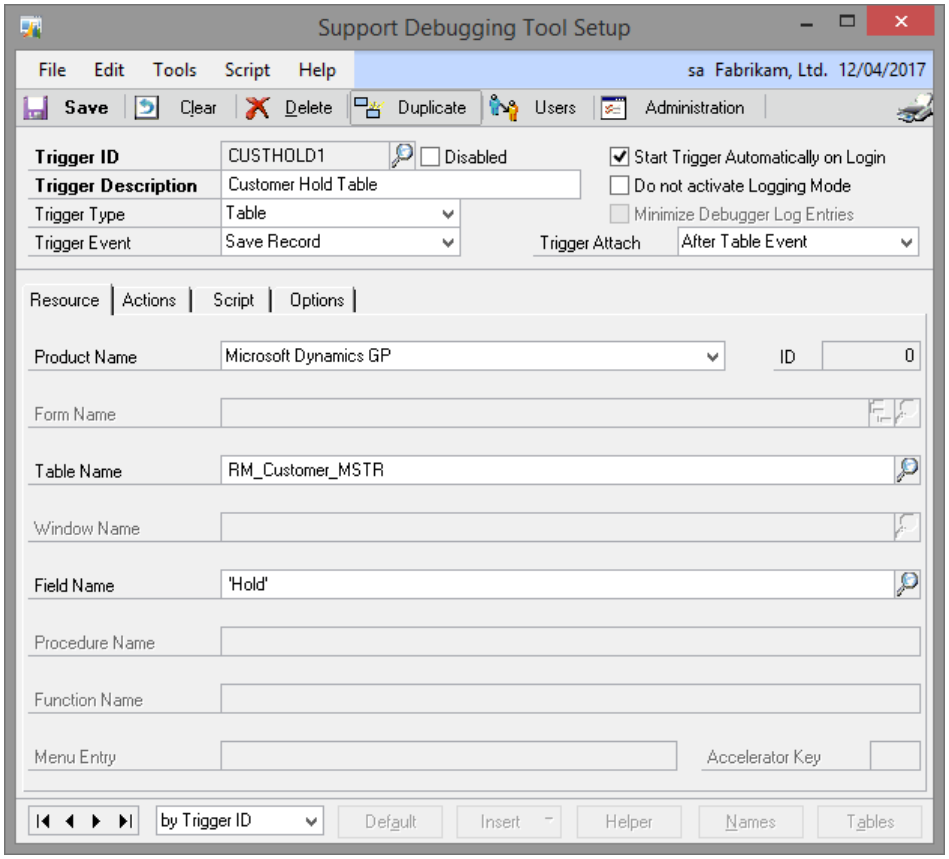
Below is an example log when the conditions were met.



Setup

You can open the Support Debugging Tool Setup window by selecting Setup Automatic Debugger Mode from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The Support Debugging Tool Setup window is used to define the Dexterity triggers that will be used to look for the exception conditions.



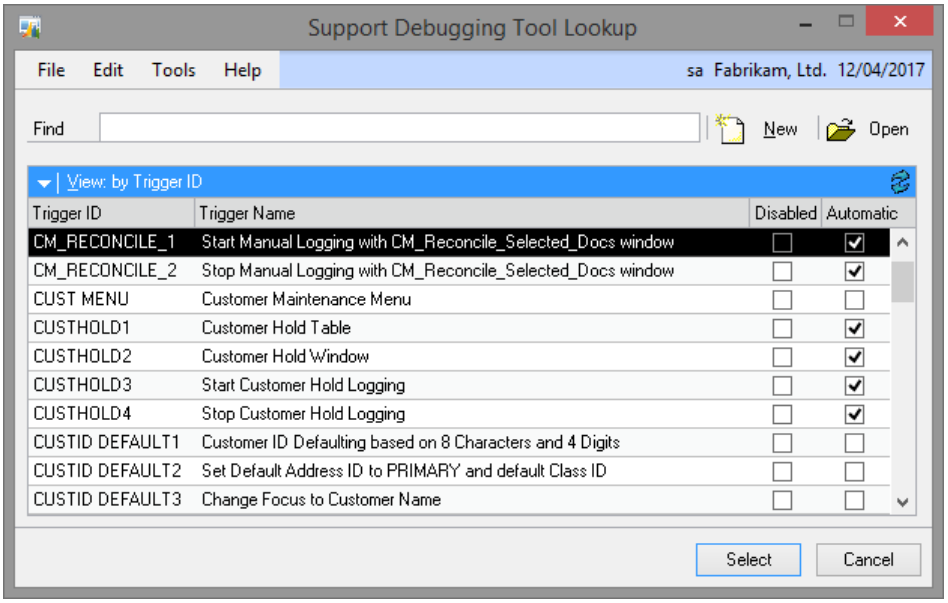
The window is divided into a header section and four tabs; the Resource Tab, the Actions Tab, the Script Tab and the Options Tab.

The system will always have a trigger ID named DEFAULT. If this trigger ID is deleted, it will be added back automatically as a blank trigger ID. The use of this trigger ID is optional.

Below is a description of the individual header fields on the window:

Trigger ID

This field contains a unique identifier for each trigger in the system. The lookup button can be clicked to select from existing trigger IDs.



Note that the Trigger IDs starting with the prefix character of tilde (~) are reserved for use by Microsoft Support.

Trigger Description

This field contains a description for the trigger.

Trigger Type

This drop down list specifies the type of trigger being defined. The following objects can be selected.

- Table
- Table restricted to Form
- Procedure
- Function
- Focus Event
- Focus Event with Table
- Warning Dialog
- Timed Event
- Add Form Menu

Trigger Event

This drop down list specifies the event for the selected object. The following events can be selected depending on the trigger type selected:

- Table
  - Save Record
  - Delete Record
  - Read Record
- Table restricted to Form
  - Save Record
  - Delete Record
  - Read Record
- Procedure
  - Global Level
  - Form Level
- Function
  - Global Level
  - Form Level
- Focus Event
  - Form Pre
  - Form Post
  - Window Pre
  - Window Post
  - Window Activate
  - Scroll Fill
  - Scroll Pre
  - Scroll Change
  - Scroll Post
  - Scroll Insert
  - Scroll Delete
  - Field Pre
  - Field Change
  - Field Post
- Focus Event with Table
  - Form Pre
  - Form Post
  - Window Pre
  - Window Post
  - Window Activate
  - Scroll Fill
  - Scroll Pre
  - Scroll Change
  - Scroll Post
  - Scroll Insert
  - Scroll Delete
  - Field Pre
  - Field Change
  - Field Post

- Warning Dialog
  - Warning Dialog
- Timed Event
  - Every 1 Minute
  - Every 5 Minutes
  - Every 10 Minutes
  - Every 15 Minutes
  - Every 30 Minutes
  - Every 60 Minutes
- Add Form Menu
  - Form Level

While the Support Debugging Tool can trigger against global and form level procedures and functions, it is unable to obtain the parameter lists for those procedures and functions.

Trigger Attach

This drop-down list specifies when the code for the Dexterity trigger is run when the selected event for the selected object occurs. The following attach modes can be selected depending on the trigger type selected:

- Table
  - After Table Event
- Table restricted to Form
  - After Table Event
- Procedure
  - Before Original
  - After Original
- Function
  - Before Original
  - After Original
- Focus Event
  - Before Original
  - After Original
- Focus Event with Table
  - Before Original
  - After Original
- Warning Dialog
  - Before Original
  - After Original
- Timed Event
  - After Timed Event
- Add Form Menu
  - After Menu Selected

When using table trigger type, it is possible to trigger only after a successful table event. This means this option cannot be used to capture a failed save, delete or read event.

Disabled

When this checkbox is marked, the current trigger is disabled and will never be activated.

Start Trigger Automatically on Login

When this checkbox is marked, the current trigger will be activated automatically after logging into Microsoft Dynamics GP. Use the Users button to specify the individual user and companies to limit for whom the trigger is used.

Do not activate Logging Mode

When this checkbox is marked, the trigger will not start logging and will not activate the Automatic Debugger Mode. It allows a trigger to be registered and used without the overhead of maintaining the log files.

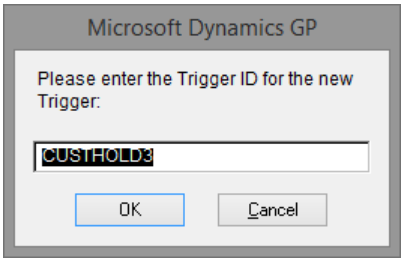
Non-logging triggers can be started automatically on login or started manually from the Automatic Debugger Mode Turn On button. To stop a non-logging trigger, use the Unregister button on the Automatic Debugger Mode Status window.

Minimize Debugger Log Entries

When using a Non-logging trigger, this option can be enabled to prevent the trigger generating entries in the Debugger_<User>_<Company>.log file unless an error occurs.

Duplicate Button

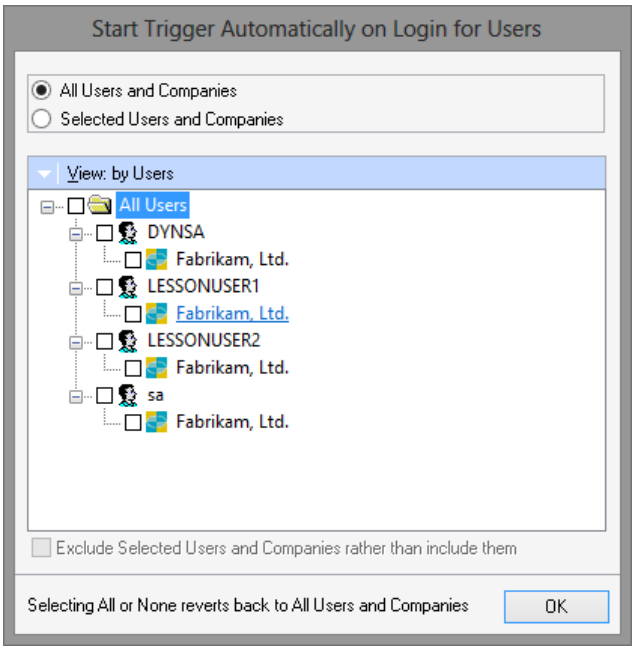
Use this button to duplicate the current trigger ID and create a new trigger ID. This is useful when an existing trigger ID is very similar to the new one you want to create.



A new trigger ID must be specified in the dialog which opens.

Users Button

Use this button to specify which users and companies should have the current trigger start automatically. Once clicked the Start Trigger Automatically on Login for Users window will open.



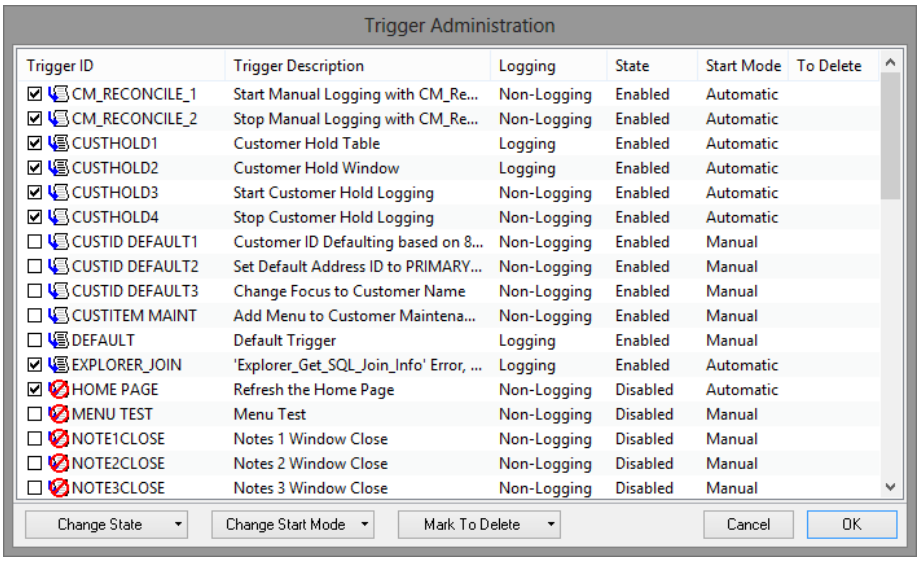
You can view this window by users or by companies and navigate the tree to select the user and company combinations as required.

If all users are selected on the tree, the tree selections will be cleared and the mode will change from Selected Users and Companies to All Users and Companies. If no users are selected on the tree, the mode will change to All Users and Companies.

The Exclude Selected Users and Companies rather than include them option allows you to invert the behavior of the window. This is handy when it is easier to specify the users and companies for whom the trigger should not be activate.

Administration Button

Use this button to administer multiple Automatic Debugger Mode Trigger IDs at the same time. Once clicked the Trigger Administration window will open.



When the Trigger Administration window is opened, the current Trigger ID is saved automatically. The Trigger Administration window is modal and must be closed before continuing to use other windows.

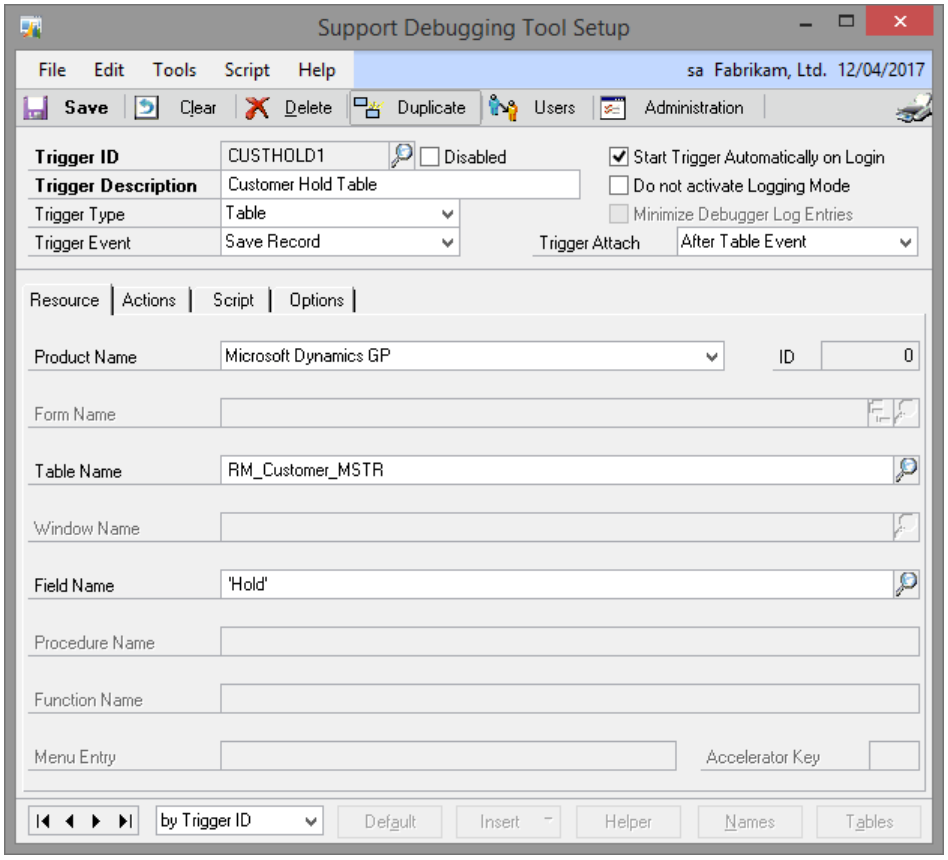
The window shows the current status of the Trigger IDs in the system. Triggers can be Enabled or Disabled, have their start mode changed between Manual and Automatic, or be deleted in bulk from this window.

To make changes, select the Trigger IDs (use control and shift keys to multi-select) and use the Change State, Change Start Mode, and Mark To Delete Buttons.

The selected changes will be made when OK is clicked. Clicking Cancel will close the window without applying any pending changes.

Resource Tab

The Resource tab contains the definition of the resource to apply the trigger against.



The following is a description of the individual resource selection fields on the tab. The actual fields available depend on the settings for Trigger Type and Trigger Event fields. The lookup button can be used to open the Resource Explorer or the Table Explorer to select the required resource:

Product Name

This drop-down list contains a list of products currently installed on the Microsoft Dynamics GP workstation.

Product ID

This non-editable field displays the product ID (as known as dictionary ID) for the selected product name.

Form Name

This field contains the technical name for the form selected.

Table Name

This field contains the technical name for the table selected.

Window Name

This field contains the technical name for the window selected.

Field Name
This field contains the technical name for the field selected.

Procedure Name
This field contains the technical name for the procedure selected.

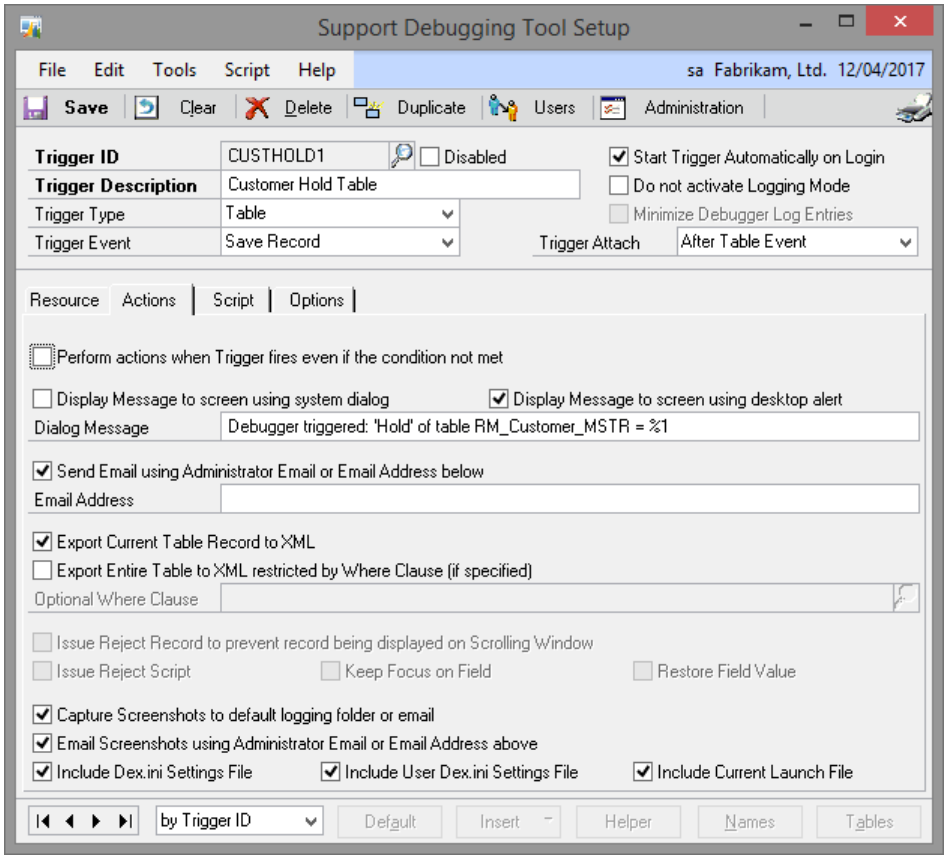
Function Name
This field contains the technical name for the function selected.

Menu Entry
This field contains the description to be displayed on the Form Menu created by this trigger.

Accelerator Key
This field contains an optional accelerator shortcut key (used with Control) for the menu entry.

Actions Tab

The Actions tab contains the actions to perform when the trigger has fired.



The following is a description of the individual action fields on the tab. These actions will be processed when the trigger fires and the conditional script returns true or if the Perform actions when Trigger fires even if the condition not met checkbox is selected.

Perform actions when trigger fires even if the condition not met

Check this checkbox when you want the actions to be processed when the trigger is fired regardless of whether the conditional script returns true.

Display Message to screen using system dialog

Select this checkbox if you want the message displayed to the screen in a system dialog box.

Display Message to screen using desktop alert

Select this checkbox if you want the message displayed to the screen in a desktop alert.

Dialog Message

This field contains the message which will be logged and displayed if the Display Message checkbox is selected. When the Field Name is specified, the message can contain the %1 placeholder which will be substituted with the field value when the message is displayed.

Send Email using Administrator Email or Email Address below
When this checkbox is selected, an email with the log details of the trigger will be sent to the Administrator Email address as setup in the Administrator Settings window, or to the specified Email Address.

Email Address
This field can be used to specify an email address to use instead of the default Administrator Email.

Export Current Table Record to XML
When this checkbox is selected, the current table buffer contents will be exported as an XML file. This action is only valid for Table triggers.

Export Entire Table to XML restricted by Where Clause
When this checkbox is selected, the entire table contents will be exported as an XML file. This action is only valid for Table triggers. A SQL Where Clause can be specified to restrict the records exported.

Optional Where Clause
This field can be used with the Export Entire Table option to define a SQL Where Clause to restrict the records exported to XML. This field is only valid for Table triggers.

Issue Reject Record
When this checkbox is selected, a reject record command will be issued to prevent the current record being shown in a scrolling window. This action is only valid for Non-logging Focus Event triggers attached to the Scroll Fill Event.

Issue Reject Script
When this checkbox is selected, a reject script command will be issued to abort the original code from executing. This action is only valid for Non-logging Focus Event triggers running before the original code.

Keep Focus on Field
When this checkbox is selected, the focus will be kept on the current field. This action is only valid for Non-logging Focus Event triggers running before the original code when the Reject Script option is used.

Restore Field Value
When this checkbox is selected, the original value of the current field will be restored. This action is only valid for Non-logging Focus Event triggers running before the original code when the Reject Script option is used.

Capture Screenshots to default logging folder or email
When this checkbox is selected, the ScreenShot utility will be used to capture screenshots of all open windows and either save them to the logging folder or email them.

Email Screenshots using Administrator Email or Email Address below
When this checkbox is selected, an email with the captured screenshots will be sent to the Administrator Email address as setup in the Administrator Settings window, or to the specified Email Address.

Include Dex.ini Settings File

This checkbox tells the ScreenShot utility whether to include the Global level Dex.ini settings file as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Include User Dex.ini Settings File

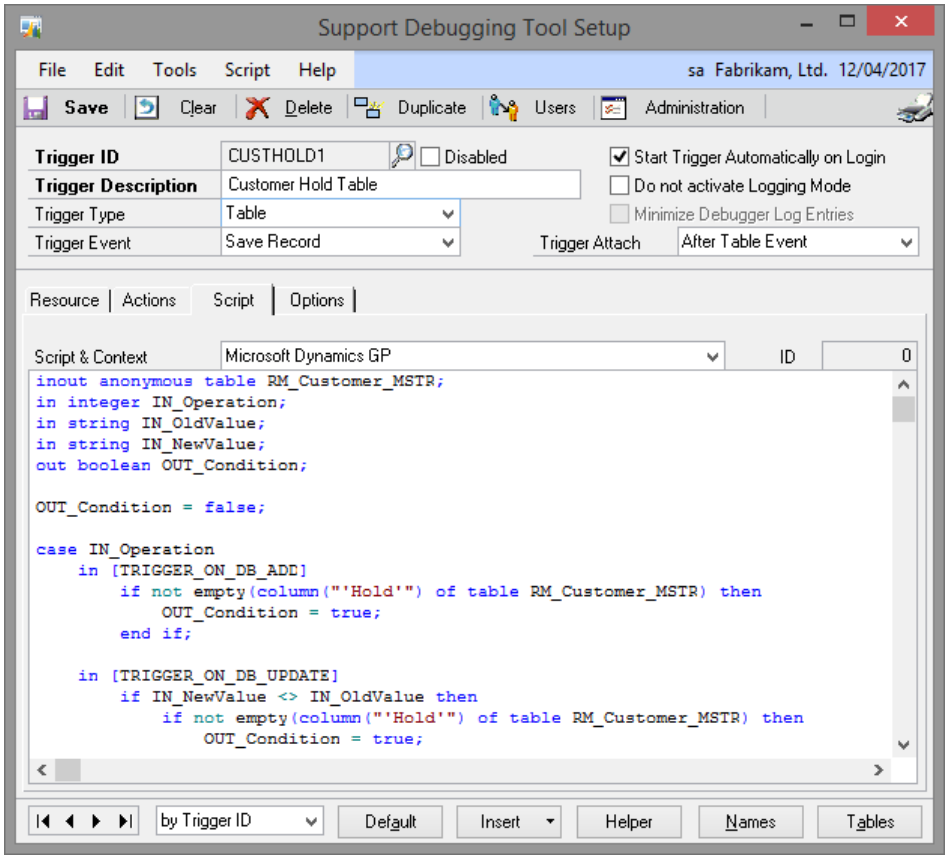
This checkbox tells the ScreenShot utility whether to include the User level Dex.ini settings file as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Include Current Launch File

This checkbox tells the ScreenShot utility whether to include the launch file, usually Dynamics.set, as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

Script Tab

The Script tab contains the Conditional script to be executed when the trigger fires.



The following is a description of the individual script fields on the tab.

Script Context

This drop-down list contains a list of products currently installed on the Microsoft Dynamics GP workstation. It is used to select the dictionary context that the conditional script will be executed in. The script context is usually the same as the dictionary ID, but can be changed if the script needs to be executed in a different dictionary to where the trigger is registered.

Script Context ID

This non-editable field displays the script context ID (as known as dictionary ID) for the selected script context.

Conditional Script

This text field contains the script to be executed when the trigger fires. The script will be populated with a default script when the trigger type, trigger event and resource information are selected. The script will have the required parameters, including a boolean OUT_Condition. The script can be used to check for the exception condition being targeted and then set OUT_Condition to true if the condition has occurred. The script is checked for syntax errors when saved.

Using the Helper Functions (see below), a script created in the Runtime Execute window or the SQL Execute window can be loaded and executed from within a conditional script of a trigger.

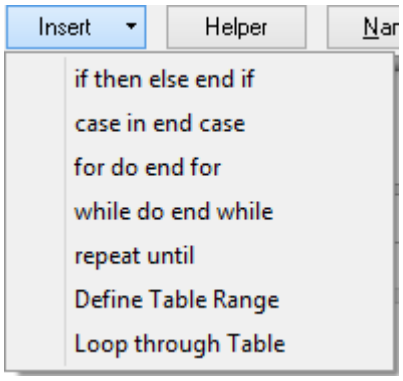
The following is a description of the additional buttons on the tab:

Default Button

Use this button to reset the Message and Conditional Script fields to the default settings based on the trigger and resource settings.

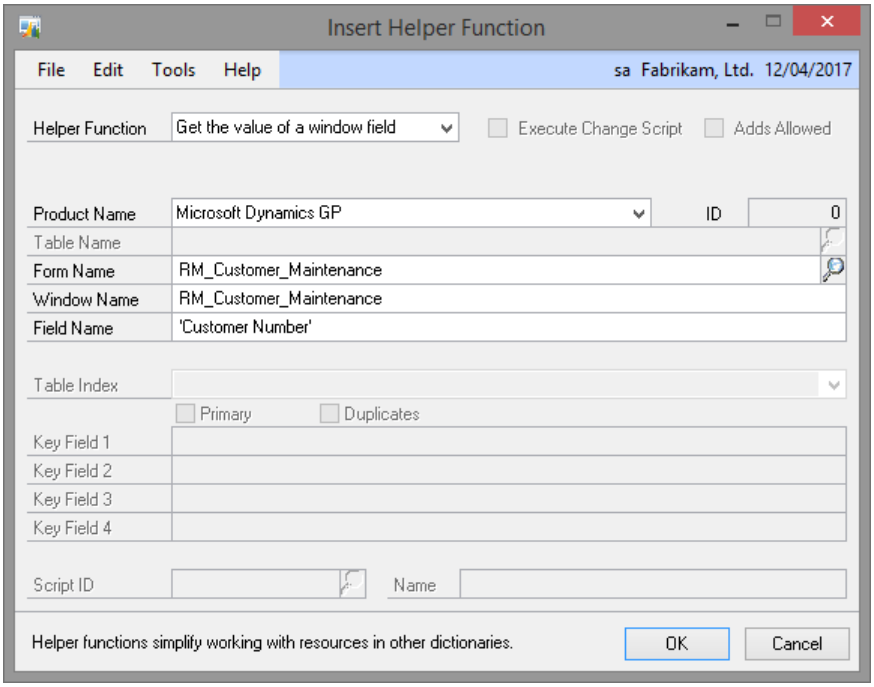
Insert Button

Use this button to insert a Dexterity sanScript code construct. The available constructs are shown below:



Helper Button

Use this button to open the Insert Helper Function window. The appropriate code for the selected helper function will be inserted into the script.

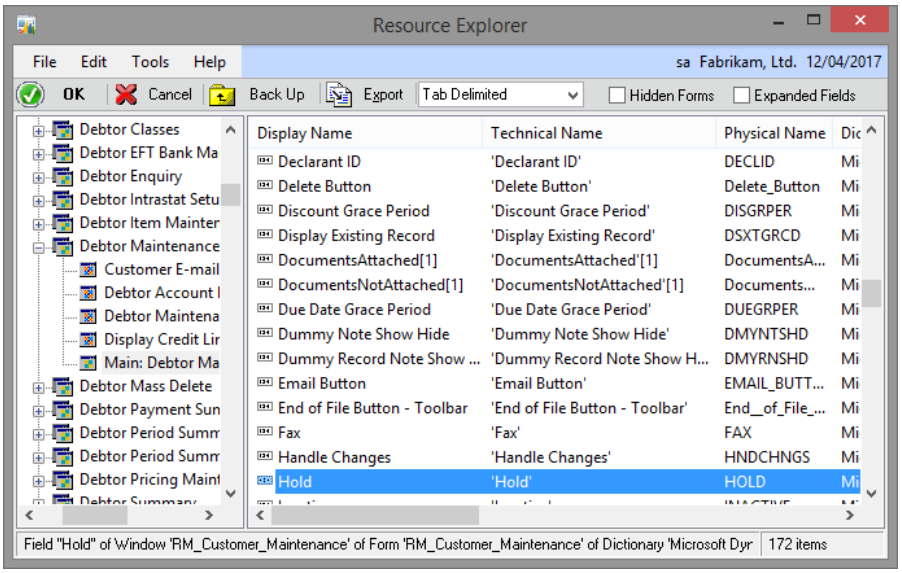


Helper functions can be used to read or write a window or table field in any window or table from any loaded dictionary. When setting a window field you can select whether to execute the field's change script. When setting a table field you can select whether adding a new record is allowed.

The table-based help functions currently support up to four key fields. The individual helper functions are covered in more detail in a later chapter.

Names Button

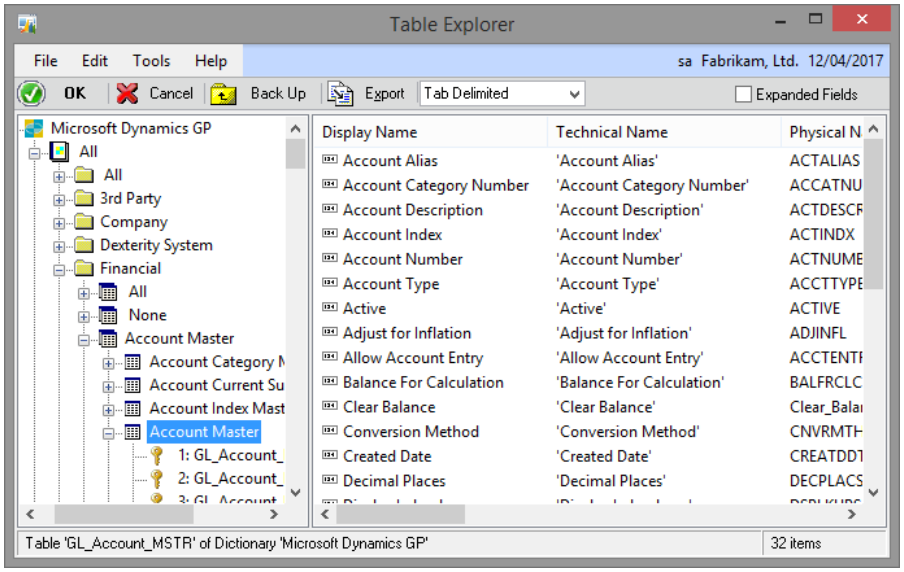
Use this button to insert a form, window or field resource into the script. Once clicked the Resource Explorer window will open.



To insert a form name or window name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

Tables Button

Use this button to insert a table or field resource into the script. Once clicked the Table Explorer window will open.

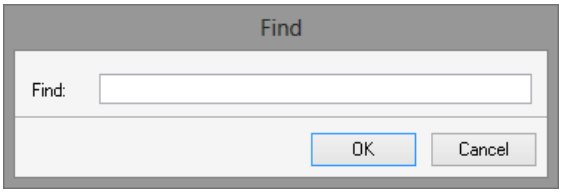


To insert a table name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

The following is a description of the Script menu available for the tab:

Find …

Use this menu option to open the script editor Find window to search for text. Control-F can be used as a shortcut.

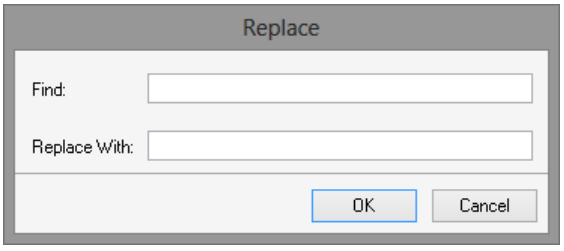


Find Next

Use this menu option to find the next occurrence. Control-G can be used as a shortcut.

Replace …

Use this menu option to open the script editor Replace window to search and replace text. Control-R can be used as a shortcut.



Replace and Find Next

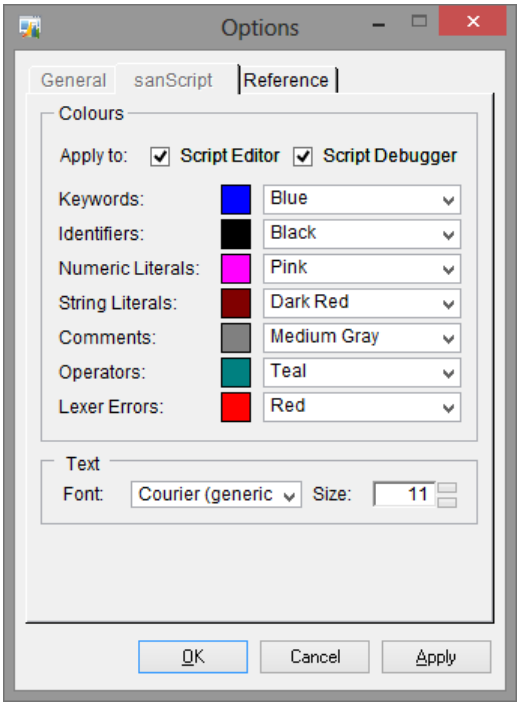
Use this menu option to replace and find the next occurrence. Control-B can be used as a shortcut.

Check Syntax

Use this menu option to check the syntax of the current script. Any errors will be displayed in a dialog window. Control-S can be used as a shortcut.

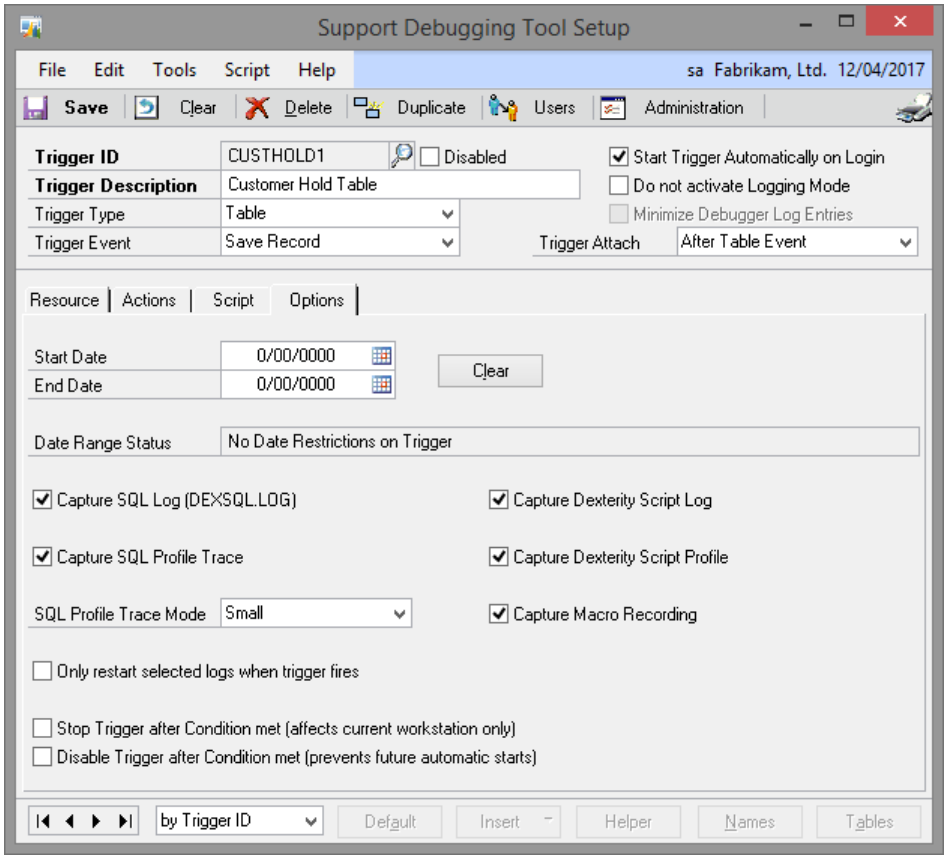
Options

Use this menu option to open the Dexterity options window to allow the syntax highlighting colors, font style, and size to be changed. Control-O can be used as a shortcut.



Options Tab

The Options tab contains optional settings which change the behavior of the trigger.



The following is a description of the individual script fields on the tab.

Start Date

You can specify a Start Date to restrict the dates that a trigger will automatically start.

End Date

You can specify an End Date to restrict the dates that a trigger will automatically start.

If the Start Date and the End Date are the same, the trigger will only be active for a single day. If the End Date is before the Start Date, then the trigger will be inactive during the date range. The status field will show the behavior based on the selected dates. If a Start Date is not specified the trigger will be active up to the End Date. If an End Date is not specified the trigger will be active from the Start Date.

Capture SQL Log

You can select which of the logging modes to enable, this option enables the SQL Logging when this trigger is active. This option is not valid for Non-logging triggers.

Capture SQL Profile Trace

You can select which of the logging modes to enable, this option enables the SQL Profile Tracing when this trigger is active. This option is not valid for Non-logging triggers.

SQL Profile Tracing is not enabled until it has been setup using the SQL Profile Trace Settings window under the Administrator Settings.

SQL Profile Trace Mode

When using SQL Profile Tracing, you can use this option to select the type of SQL Profile Trace created. You can select between Small, Medium, Large and Performance. The Other mode can be used in conjunction with a customized MBS_SQL_Tracing_API_5 stored procedure in the DYNAMICS database. This option is not valid for Non-logging triggers.

Capture Dexterity Script Log

You can select which of the logging modes to enable, this option enables the Dexterity Script Logging when this trigger is active. This option is not valid for Non-logging triggers.

Capture Dexterity Script Profile

You can select which of the logging modes to enable, this option enables the Dexterity Script Profiling when this trigger is active. This option is not valid for Non-logging triggers.

Capture Macro Recording

You can select which of the logging modes to enable, this option enables the Macro Recording when this trigger is active. This option is not valid for Non-logging triggers.

For v10.00: Macro Recording is not enabled until it has been setup using the Macro Recording Settings window under the Administrator Settings.

Macro Recording can only work when a single instance of Microsoft Dynamics GP is running on a workstation, or if multiple instances are running, Macro Recording will only work on the first instance launched.

Only restart selected logs when trigger fires

Using this checkbox, you can control which logging modes are restarted when the trigger fires. By default all active logging modes are restarted each time a trigger fires. If this checkbox is enabled, only the logging modes selected for this trigger will be restarted when this trigger fires. This option is not valid for Non-logging triggers.

Stop Trigger after Condition met

Using this checkbox, you can specify that a trigger should only be used once per session. When the Trigger fires and the condition is met, the trigger will be stopped until next login or manual restart.

Disable trigger after Condition met

Using this checkbox, you can specify that a trigger should only be used once. When the trigger fires and the condition is met, the trigger will be disabled preventing it from starting until it is re-enabled.

Restriction of Scope

The Support Debugging Tool has a restriction which must be taken into account when using the Automatic Debugger Mode.

When using a table trigger type, the Support Debugging Tool uses a Dexterity database trigger. A Dexterity database trigger is only capable of tracking changes made to the tables using Dexterity commands.

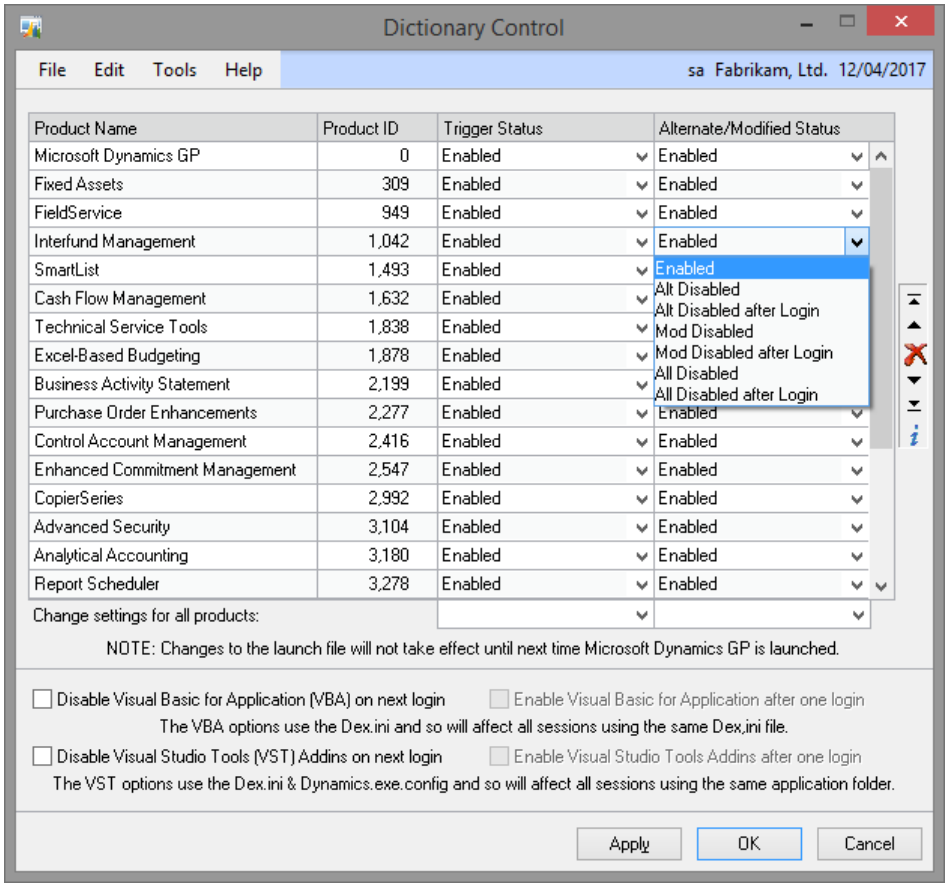
Changes made to tables using tools or applications other than Dexterity will not be picked up by the Support Debugging Tool. This can include table changes made directly by SQL Query Analyzer, pass-through SQL commands, SQL stored procedures, SQL triggers, or updates from eConnect, Integration Manager's SQL Optimized or Microsoft Dynamics GP eConnect adapters, ADO (ActiveX Data Objects) from VBA (Visual Basic for Applications) or any other external application.

Dictionary Control

You can open the Dictionary Control window by selecting Dictionary Control from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

Dictionary Control can be used to troubleshoot issues with third party dictionaries. You can effectively remove dictionaries from the system one-by-one until the issue stops. Then the last dictionary to be removed can be investigated further.

You can use Trigger Status to disable Dexterity triggers for a specific Product in a similar fashion to the Customization Status window in Microsoft Dynamics GP. The added benefit of Dictionary Control is that it can remember the settings and automatically disable the product on the next login.



Dictionary Control can also disable alternate and/or modified windows for third party dictionaries using the Alternate Status option. This does not change any security settings.

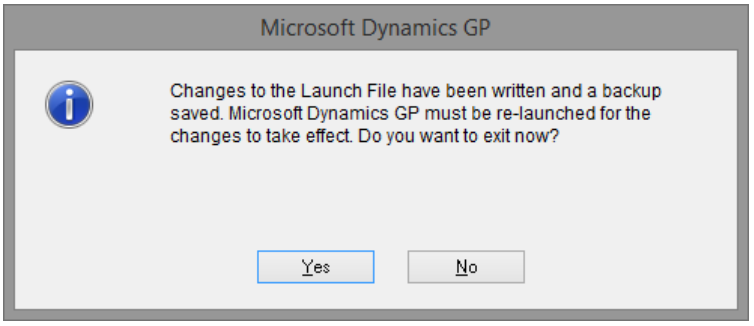
The drop-down lists at the bottom of the window can be used to change settings for all dictionaries.

Using Dictionary Control to disable the triggers and alternate windows for a third party dictionary can produce the same effect as removing the dictionary from the Dynamics.set launch file without requiring any backups or manual editing.

Sometimes issues can occur because of the order of the dictionaries in the system. Different dictionaries adding triggers for the same event in the application can sometimes clash causing unexpected or undesirable results. The order that triggers from different products will execute is affected by the order of the products in the Dynamics.set Launch File. By changing the order of the products, you can change the order of the triggers and avoid the issues.

Under most circumstances having two or more dictionaries triggering from the same event would not cause any problems regardless of the order the triggers are executed in. Sometimes, a trigger from one dictionary can make changes to data which affect the behaviour of a trigger from a second dictionary thus causing the code to fail. It is this type of situation which can often be fixed by re-ordering the dictionaries.

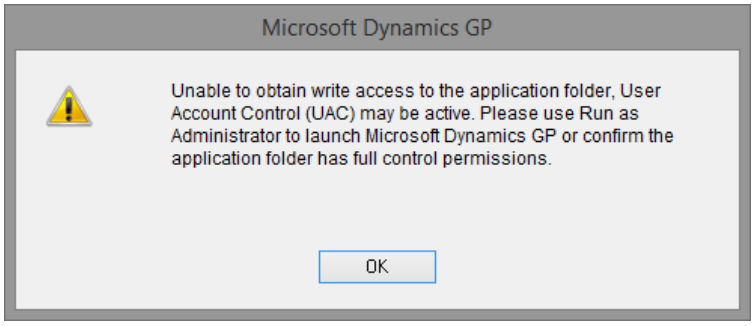
Dictionary Control allows the order of the products to be changed using the Top, Up, Down and Bottom buttons. You can also remove a product with the Delete Button. Any changes to the Dynamics.set launch file will be saved when OK is clicked. You will be requested to restart Microsoft Dynamics GP after the changes have been saved.



A backup of the original file will be saved as Backup X of Dynamics.set, where X will be a number starting at 1.

If using the Dictionary Control window to disable access to modified windows be aware that by displaying the original window, users might have access to fields previously hidden or disabled on the modified version of the window.

If User Account Control (UAC) is preventing write access to the application folder, you will see the following dialog displayed. You will need to use Run as Administrator to allow access and complete the changes.



Dictionary Control now has the ability to disable Visual Basic for Applications (VBA) and Visual Studio Tools (VSTools) on next login.

The following is a description of the additional checkboxes on the window.

Disable Visual Basic for Applications (VBA) on next login

This checkbox disables Visual Basic for Applications (VBA) when the application is restarted. This option uses the VBADisable Dex.ini setting.

Enable Visual Basic for Applications after one login

This checkbox automatically re-enables Visual Basic for Applications (VBA) for the application after the first restart. This option uses the MBS_Debug_VBADisableReset Dex.ini setting.

Disable Visual Studio Tools (VST) Addins on next login

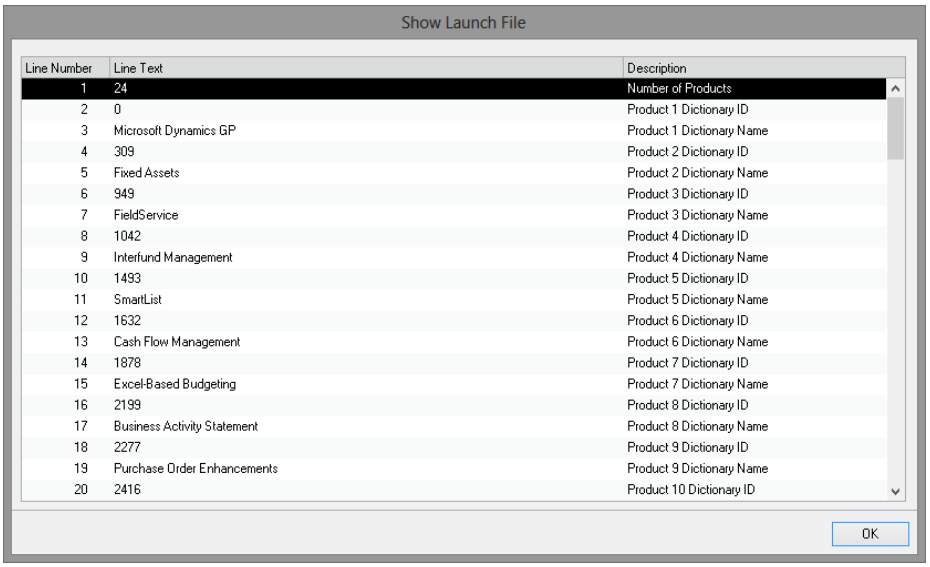
This checkbox disables Visual Studio Tools (VST) Addins when the application is restarted. This option uses the MBS_Debug_VSTDisable Dex.ini setting.

Enable Visual Studio Tools Addins after one login

This checkbox automatically re-enables Visual Studio Tools (VST) Addins for the application after the first restart. This option uses the MBS_Debug_VSTDisableReset Dex.ini setting.

The Visual Basic for Applications and Visual Studio Tools options are not available if running on the Web Client. The Visual Studio Tools options will be disabled if User Account Control (UAC) is preventing write access to the application folder. This is because the Dynamics.exe.config file must be renamed as part of the process of disabling Visual Studio Tools Addins.

If you want to check exactly what is contained in the Dynamics.set launch file and confirm that each line is in the correct position you can click the Info button to open the Show Launch File window.



The Description column in this window describes what information should be on the current line of the file for the file to be valid.

To use Dictionary Control, a user must have security access to the Customization Status window. This window may have access disabled automatically on each login when using Field Level Security and Field Security IDs are active for the current user and company.

Dictionary Control cannot be used to disable alternate windows and forms or triggers in the Support Debugging Tool. As the core Microsoft Dynamics GP dictionary cannot have alternate windows, Dictionary Control cannot be used to disable alternate windows.

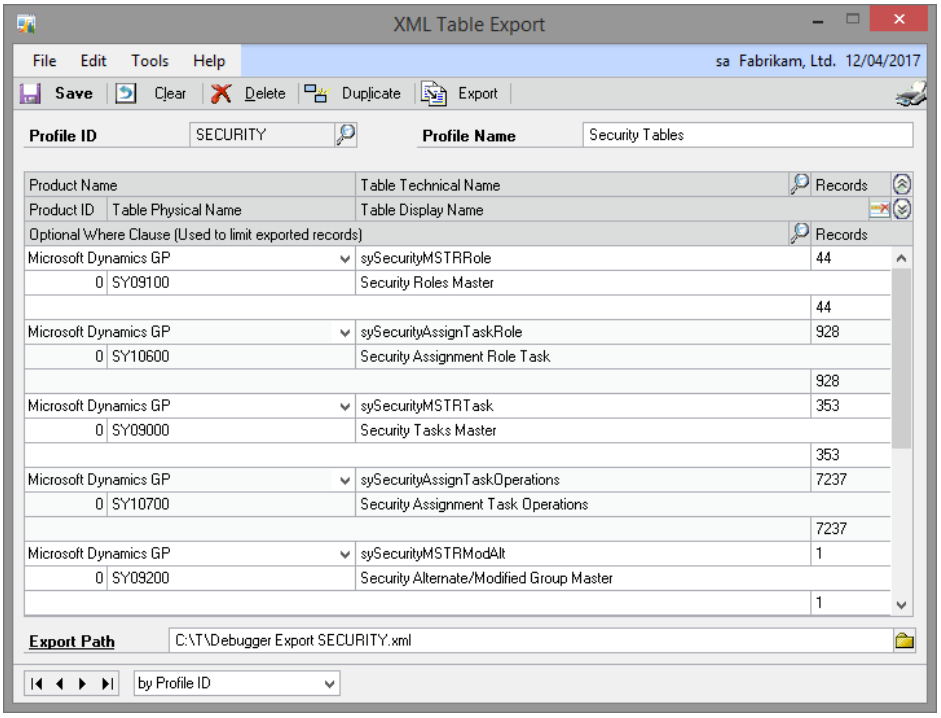
When running on the Web Client, Dictionary Control cannot be used to modify the launch file and so the movement and delete buttons are disabled.

XML Table Export

You can open the XML Table Export window by selecting XML Table Export from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The XML Table Export window can be used to copy the contents of one or more tables residing in any product into an XML file. All tables selected will be exported into the single XML file listed on the Export Path.

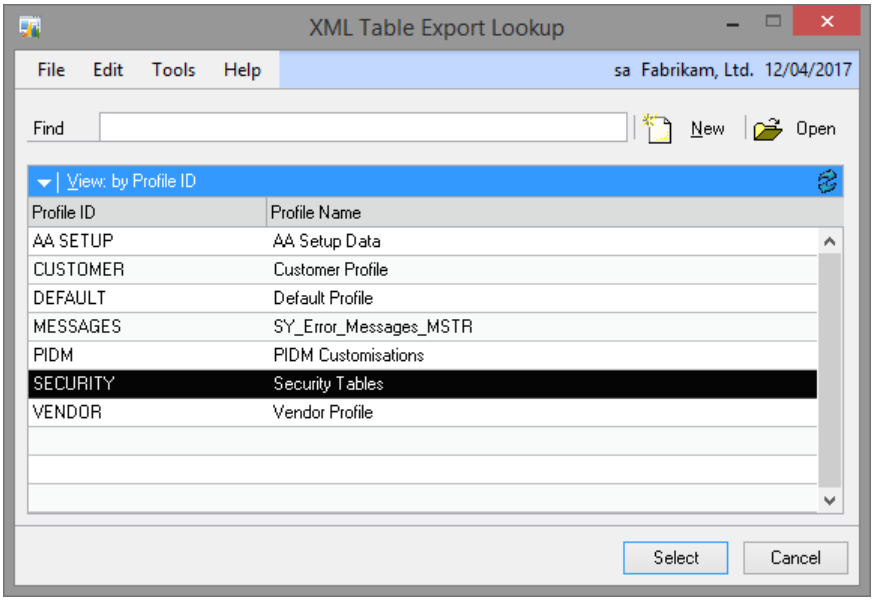
Using separate Profile IDs allows multiple sets of tables to be stored for particular related groups.



The following is a description of the individual fields on the window:

Profile ID

This field contains a unique identifier for each XML Table Export profile in the system. The lookup button can be clicked to select from existing script IDs.



Note that the Profile IDs starting with the prefix character of tilde (~) are reserved for use by Microsoft Support.

Profile Name

This field contains a description for the XML Table Export profile.

Table List

Select the tables you want to export and add them to the list. You can use the lookup or manually enter the Table Technical Name or Table Physical Name fields.

Export Path

This field contains the path of the file name to which the tables will be export as XML.

The following is a description of the additional buttons on the window:

Duplicate Button

Use this button to duplicate the current profile ID to a new profile ID. This is useful when an existing profile ID is very similar to the new one you want to create.

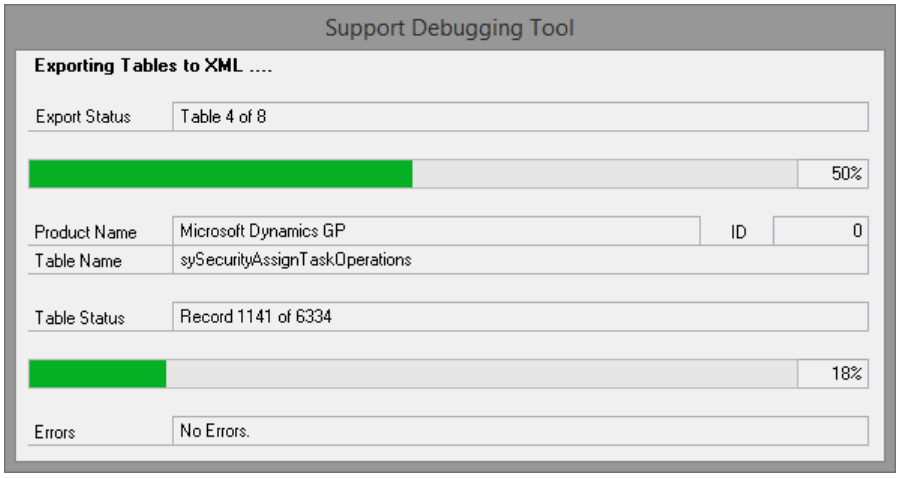
Export Button

Use this button to export the data to the file named in the Export Path field.

For each table specified in the scrolling window section of this window, you can specify an Optional SQL Where Clause Clause to restrict the records export for that table.

XML Table Export can be used to obtain a customer's data for specific tables without requiring a full SQL database backup. Just select the tables for which you need the data and click OK to save the selection. Then use the Configuration Export/Import window to export the setting file to send to the customer. The customer can then import the settings and use the XML Export window to export the desired tables.

During the export or import process, the following progress window will be displayed.



XML Table Export can be used to backup data before running test scenarios so the data can be restored afterwards to allow the scenarios to be run again with the same initial data.

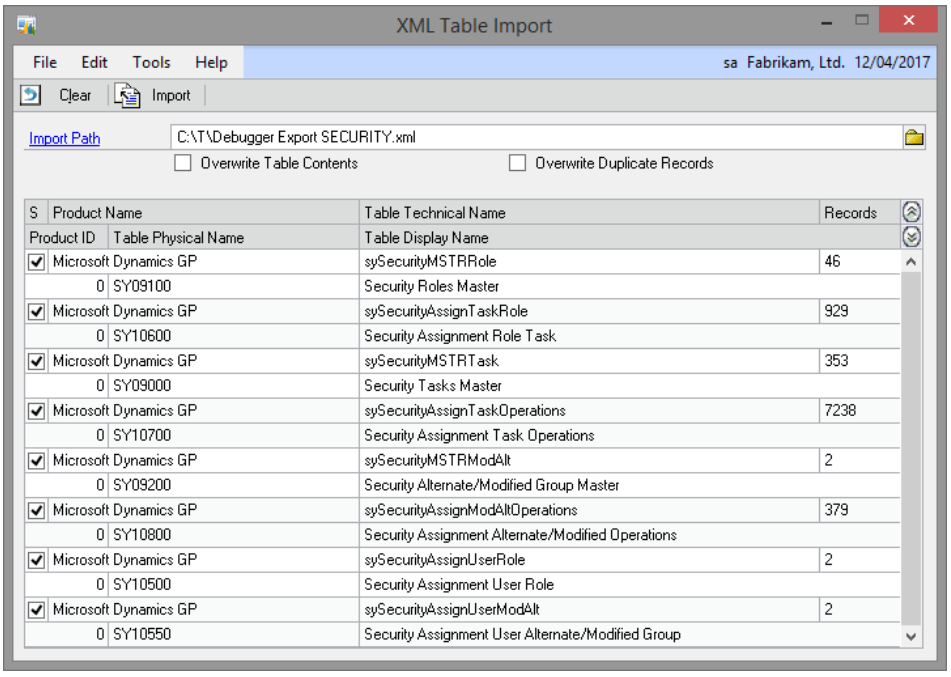
There is no data validation or business logic checking when data is imported using XML Table Import. This is similar to the Dexterity Table Import Utility. It is best to ensure that all related tables are exported by XML Table Export.

If both the XML Table Export window and XML Table Import window are open, the import path will default to the export path from the XML Table Export window.

XML Table Import

You can open the XML Table Import window by selecting XML Table Import from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The XML Table Import window can be used to import the contents of a number of tables from an XML file previously exported by the XML Table Export window.



Select the XML file as the Import Path. The tables contained in the file will be listed.

Select the tables you want to import and then click Import to start importing.

When importing data into tables it is possible that the tables already contain data and that duplicate records may occur. XML Table Import has overwrite options to handle this situation.

The following Overwrite options are available:

Overwrite Table Contents

Checking this option will cause the original contents of the table to be deleted prior to importing the XML file. None of the original data will be kept.

Overwrite Duplicate Records

Checking this option will allow XML Table Import to overwrite a duplicate record with the data from the XML file. If this option is not checked and a duplicate occurs, the data from the XML file will be ignored and a duplicate record error logged.

During the export or import process, the progress window will be displayed.

XML Table Import can be used to restore data from backups you made before running test scenarios. This allows the scenarios to be run again with the same initial data.

There is no data validation or business logic checking when data is imported using XML Table Import. This is similar to the Dexterity Table Import Utility. It is best to ensure that all related tables are exported by XML Table Export.

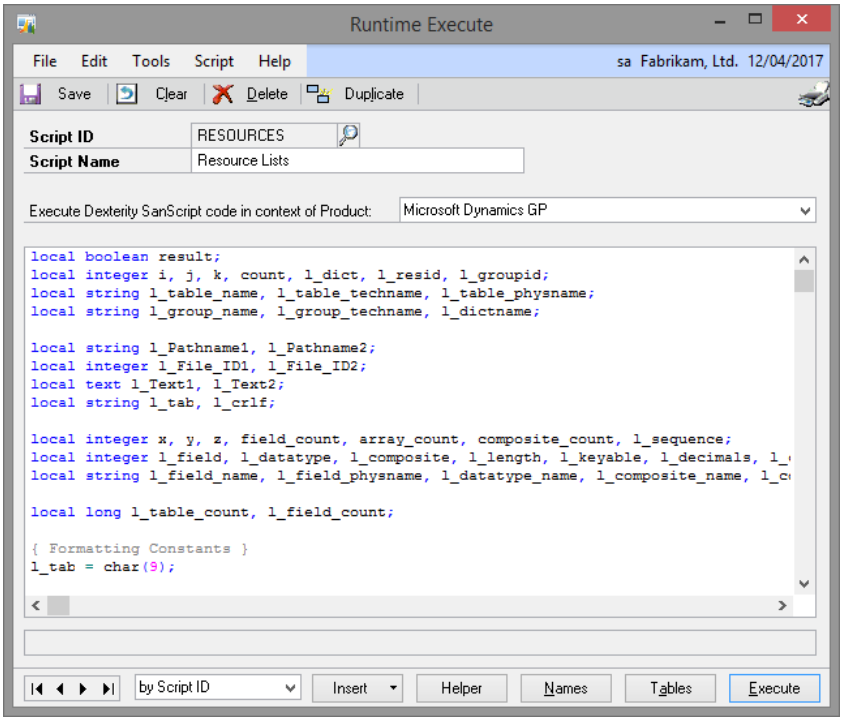
If both the XML Table Export window and XML Table Import window are open, the import path will default to the export path from the XML Table Export window.

Runtime Execute

You can open the Runtime Execute window by selecting Runtime Execute from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The Runtime Execute window can be used to run any Dexterity sanScript code without requiring the Dexterity development environment. Scripts written in this window can be used to manipulate tables using Dexterity commands or to call existing functions and procedures in any dictionary.

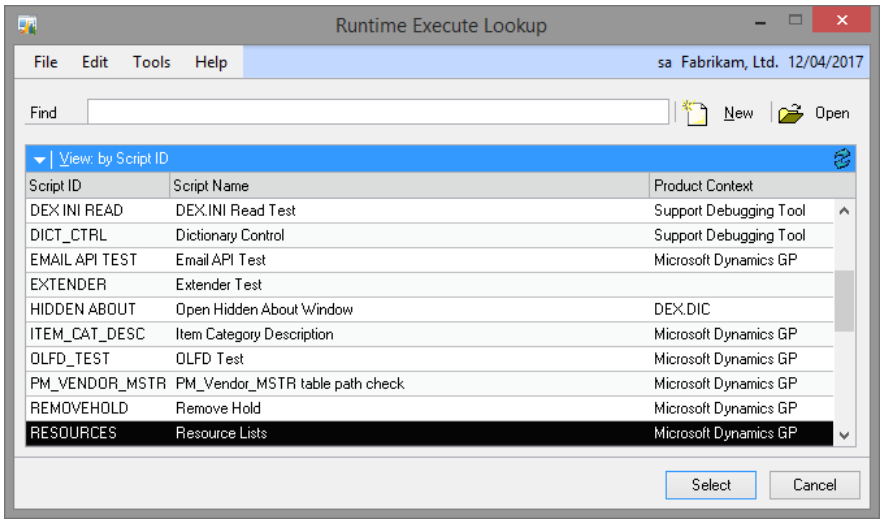
Script IDs created in this window can be loaded and executed from an Automatic Debugger Mode trigger or another Runtime Execute script. This allows code re-use in a similar fashion to having multiple procedure calls.



The following is a description of the individual fields on the window:

Script ID

This field contains a unique identifier for each Runtime Execute script in the system. The lookup button can be clicked to select from existing script IDs.



Note that the Script IDs starting with the prefix character of tilde (~) are reserved for use by Microsoft Support.

Script Name

This field contains a description of the script.

Execute Dexterity SanScript code in the context of Product

This drop-down list contains a list of products currently installed on the Microsoft Dynamics GP workstation.

Script

This text field contains the script to be executed. It cannot have any parameters. The script runs as though it is a global procedure in the context of the dictionary specified in the drop-down list. The script is checked for syntax errors when saved.

Runtime Execute can be used to manipulate data in tables when complex business logic is required. In this situation writing the equivalent code in Transact SQL can be extremely difficult. You could loop through a range of records in table and conditionally make different changes depending on the data in the records. For example, re-formatting phone numbers in the Customer Master table to different formats depending on whether they are domestic, international or mobile/cell numbers.

The following is a description of the additional buttons on the window:

Insert Button

Use this button to insert a Dexterity sanScript code construct. See the section under Automatic Debugger Mode for more information.

Helper Button

Use this button to open the Insert Helper Function window and insert a helper function into the script. See the section under Automatic Debugger Mode for more information.

The Helper Function Assistant can also create template scripts for use with Runtime Execute and Report Writer Functions as described in chapter 6.

Names Button

Use this button to insert a form, window or field resource into the script. Once clicked the Resource Explorer window will open. See the section under Automatic Debugger Mode for more information.

Tables Button

Use this button to insert a table or field resource into the script. Once clicked the Table Explorer window will open. See the section under Automatic Debugger Mode for more information.

Execute Button

Use this button to execute the script in the context of the dictionary specified. Any compile errors will be shown in the status pane below the script. Execution errors will cause an Exception Error Dialog to open.

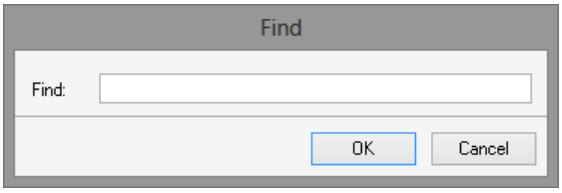
Duplicate Button

Use this button to duplicate the current script ID to a new script ID. This is useful when an existing script ID is very similar to the new one you want to create.

The following is a description of the Script menu available for the window:

Find …

Use this menu option to open the script editor Find window to search for text. Control-F can be used as a shortcut.

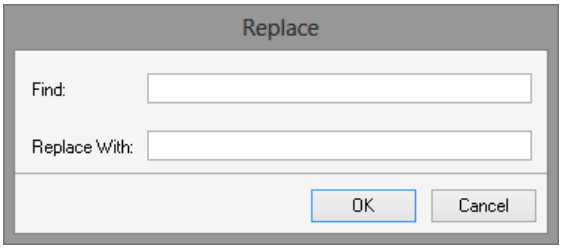


Find Next

Use this menu option to find the next occurrence. Control-G can be used as a shortcut.

Replace …

Use this menu option to open the script editor Replace window to search and replace text. Control-R can be used as a shortcut.



Replace and Find Next

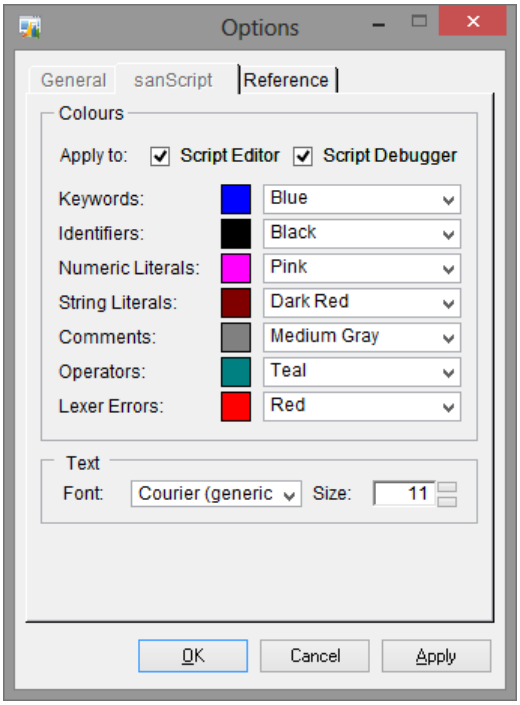
Use this menu option to replace and find the next occurrence. Control-B can be used as a shortcut.

Check Syntax

Use this menu option to check the syntax of the current script. Any errors will be displayed in a dialog window. Control-S can be used as a shortcut.

Options

Use this menu option to open Dexterity options window to allow the syntax highlighting colors, font style, and size to be changed. Control-O can be used as a shortcut.

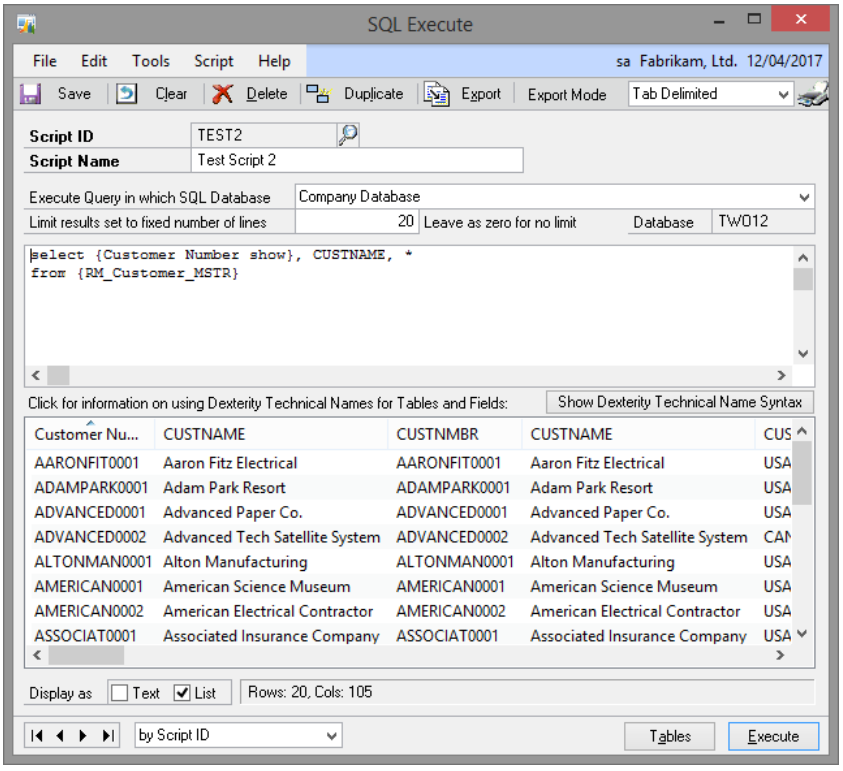


SQL Execute

You can open the SQL Execute window by selecting SQL Execute from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The SQL Execute window can be used to run any Transact SQL statements without requiring the SQL Administration Tools or MS Query. Commands written in this window can be used to view or manipulate data in any table. This window is similar to the Query Analyzer window that is installed with the SQL Server client tools.

Script IDs created in this window can be loaded and executed from an Automatic Debugger Mode trigger or Runtime Execute script. This allows a Transact SQL query to be used within Dexterity code.



Dexterity table and field names can be used in the Transact SQL when surrounded by braces { }. They will be converted to the equivalent physical names prior to the code being executed.

The **alias** keyword can be used to specify an alias other than the table's physical name. The **show** keyword can be used to display the field's Dexterity display name as the column name. The **field** keyword is used to limit the generated physical equivalents to be only the column name without the table name or alias prefix.

Following are some examples of using the resource name conversions and keywords:

```
select * from {table RM_Customer_MSTR}
```

is converted to

```
select * from RM00101
```

```
select {'Customer Number' of table RM_Customer_MSTR}
from {table RM_Customer_MSTR}
```

is converted to

```
select RM00101.CUSTNMBR
from RM00101
```

```
select {'Customer Number' of table RM_Customer_MSTR show}
from {table RM_Customer_MSTR}
```

is converted to

```
select RM00101.CUSTNMBR as [Customer Number]
from RM00101
```

```
select {'Customer Number' of table RM_Customer_MSTR field}
from {table RM_Customer_MSTR}
```

is converted to

```
select CUSTNMBR
from RM00101
```

```
select {'Customer Number' of table RM_Customer_MSTR show field}
from {table RM_Customer_MSTR}
```

is converted to

```
select CUSTNMBR as [Customer Number]
from RM00101
```

```
select {'Customer Number' of table RM_Customer_MSTR show alias a}
from {table RM_Customer_MSTR alias a}
```

is converted to

```
select a.CUSTNMBR as [Customer Number]
from RM00101 a
```

The "table" keyword, specifying the table for a field, and surrounding field names containing spaces with single quotes are now optional, so

```
select {Customer Number}
from {RM_Customer_MSTR}
```

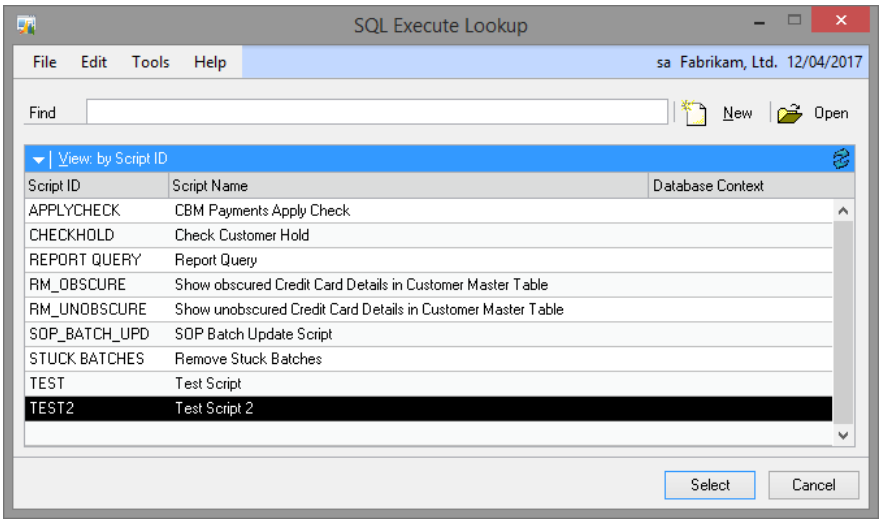
is converted to

```
select CUSTNMBR
from RM00101
```

The following is a description of the individual fields on the window:

Script ID

This field contains a unique identifier for each SQL Execute script in the system. The lookup button can be clicked to select from existing script IDs.



Note that the Script IDs starting with the prefix character of tilde (~) are reserved for use by Microsoft Support.

Script Name

This field contains a description of the script.

Execute Query in which SQL Database

This drop down list contains a list of SQL databases. The System database and each of the company databases appear in this list.

Limit results set to fixed number of lines

You can use this field to limit the amount of data returned in the results set. Set its value to zero (0) for no limit.

Setting the value of this field to zero (0) can cause SQL Execute to take a long time to display the results if the returned results set is very large.

Database

This non-editable field shows the name of the selected SQL Database.

Script

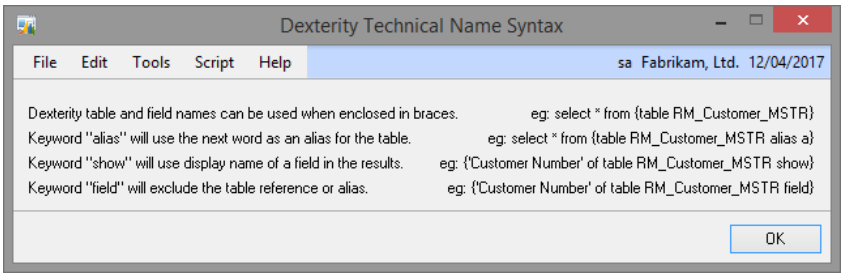
This text field contains the Transact SQL statements to be executed.

SQL Execute can be used to manipulate data in tables when large set-based changes are required. In this situation writing the equivalent Dexterity sanScript code may not be the most efficient method.

The following is a description of the additional buttons on the window:

Show Dexterity Technical Name Syntax Button

Use this button to display examples of how Dexterity Technical Names can be used in the script.



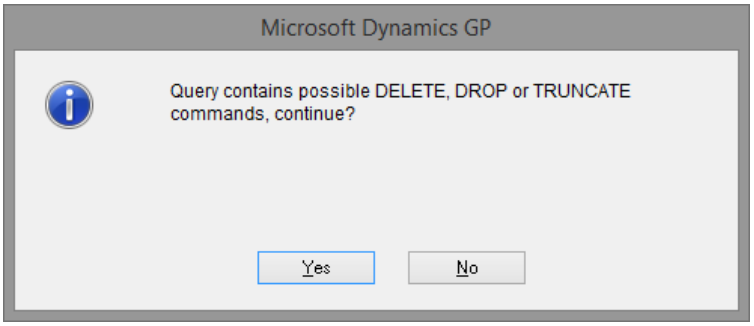
Tables Button

Use this button to insert a table or field resource into the script. Once clicked the Table Explorer window will open. See the section under Automatic Debugger Mode for more information.

Execute Button

Use this button to execute the script in the context of the SQL database specified. Any execution errors will cause an Exception Error Dialog to open. Results can be shown as unformatted text or as a list.

Before the SQL script is executed, it is checked for possible damaging commands and if they exist an additional confirmation is required.



Duplicate Button

Use this button to duplicate the current script ID to a new script ID. This is useful when an existing script ID is very similar to the new one you want to create.

Export Button

This button will allow the result set displayed in the list view to be exported to a file or directly to an email. The default email settings can be set up in the Administrator Settings window.

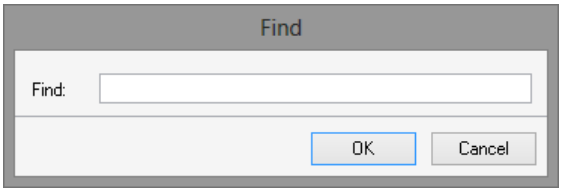
Export Mode

Use this drop down list to select the format for the exported file. The file can be exported as Tab Delimited, Comma Delimited or as a HTML Table.

The following is a description of the Script menu available for the window:

Find …

Use this menu option to open the script editor Find window to search for text. Control-F can be used as a shortcut.

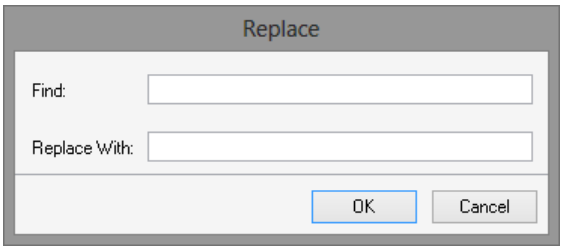


Find Next

Use this menu option to find the next occurrence. Control-G can be used as a shortcut.

Replace …

Use this menu option to open the script editor Replace window to search and replace text. Control-R can be used as a shortcut.



Replace and Find Next

Use this menu option to replace and find the next occurrence. Control-B can be used as a shortcut.

Check Syntax

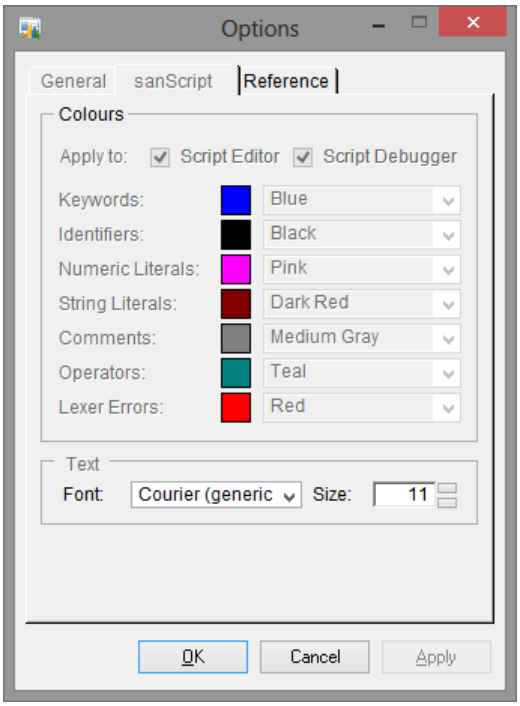
Use this menu option to check the syntax of Dexterity resource names contained in curly braces in the current script. Control-S can be used as a shortcut.

Convert References

Use this menu option to convert the Dexterity resource names contained in curly braces in the current script to their SQL equivalents.

Options

Use this menu option to open Dexterity options window to allow the font style and size to be changed. Control-O can be used as a shortcut.

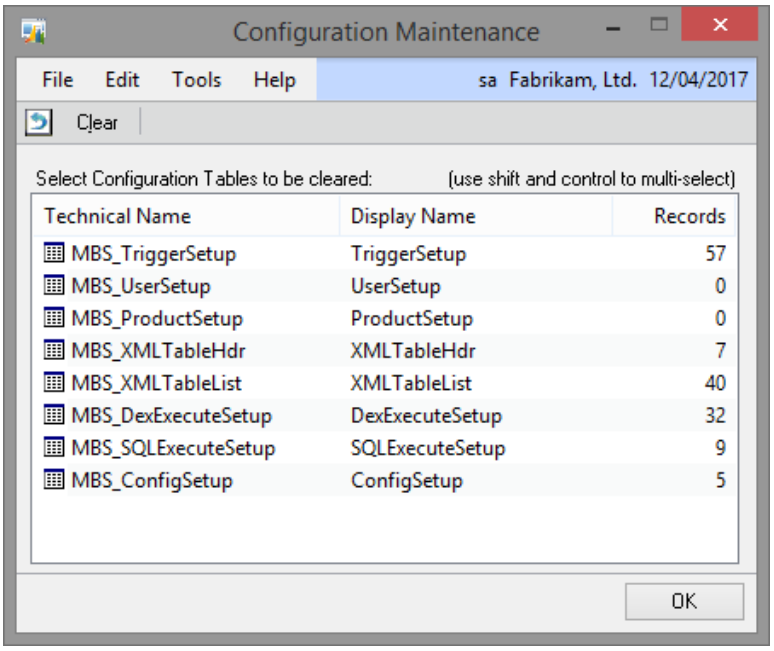


The Color options are disabled because the syntax highlighting is not available for SQL scripts.

Configuration Maintenance

You can open the Configuration Maintenance window by selecting Configuration Maintenance from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The Configuration Maintenance window can be used to clear the contents of the Support Debugging Tool settings tables.



The following is a description of the individual fields on the window:

Clear Button

This button will clear the contents of the selected tables. You can use the shift and control keys to select multiple tables.

The system will always have a trigger ID named DEFAULT. This trigger will be automatically added when the MBS_TriggerSetup table is cleared.

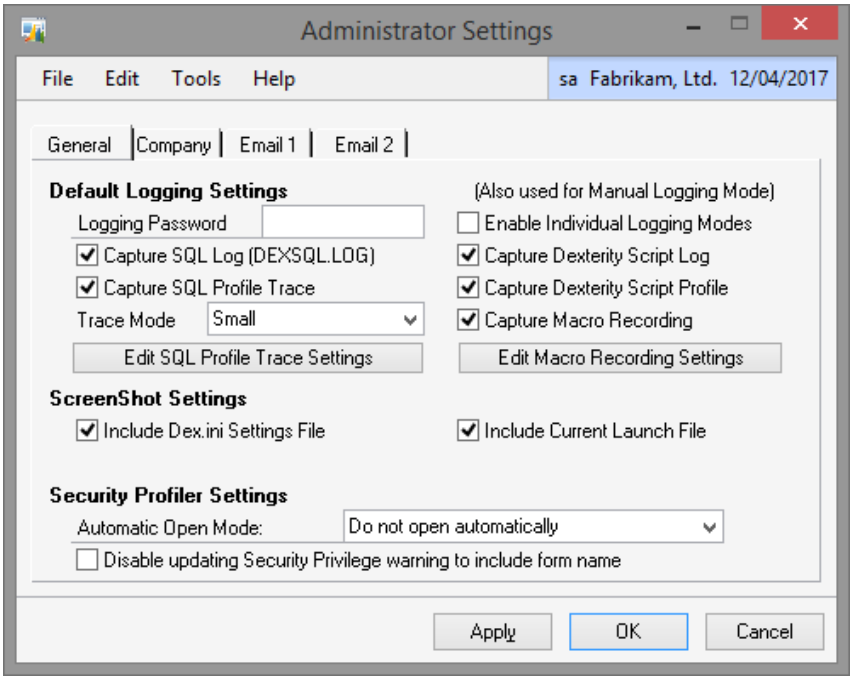
Administrator Settings

You can open the Administrator Settings window by selecting Administrator Settings from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The Administrator Settings window can change settings used within the Support Debugging Tool. It is divided into four tabbed sections.

General Tab

The General tab contains settings for controlling the behavior of some of the features of the Support Debugging Tool.



The following is a description of the individual fields on the window:

Logging Password

You can specify an optional password to be requested before Manual Logging Mode can be enabled.

Enable Individual Logging Modes

You use this option to enable Individual Logging Control. By default this option is disabled which hides the Logging Options button on the Support Debugging Tool main window

Capture SQL Log

You can select which of the logging modes to enable, this option enables the SQL Logging when Manual Logging Mode is used.

Capture SQL Profile Trace

You can select which of the logging modes to enable, this option enables the SQL Profile Tracing when Manual Logging Mode is used.

SQL Profile Tracing is not enabled until it has been setup using the SQL Profile Trace Settings window.

SQL Profile Trace Mode

When using SQL Profile Tracing, you can use this option to select the type of SQL Profile Trace created. You can select between Small, Medium, Large and Performance. The Other mode can be used in conjunction with a customized MBS_SQL_Tracing_API_5 stored procedure in the DYNAMICS database.

Capture SQL Log

You can select which of the logging modes to enable, this option enables the SQL Logging when Manual Logging Mode is used.

Capture Dexterity Script Log

You can select which of the logging modes to enable, this option enables the Dexterity Script Logging when Manual Logging Mode is used.

Capture Dexterity Script Profile

You can select which of the logging modes to enable, this option enables the Dexterity Script Profiling when Manual Logging Mode is used.

Capture Macro Recording

You can select which of the logging modes to enable, this option enables the Macro Recording when Manual Logging Mode is used.

For v10.00: Macro Recording is not enabled until it has been setup using the Macro Recording Settings window.

Macro Recording can only work when a single instance of Microsoft Dynamics GP is running on a workstation, or if multiple instances are running, Macro Recording will only work on the first instance launched.

Edit SQL Profile Trace Settings

This button will open the SQL Profile Trace Settings window (see section below).

Edit Macro Recording Settings

For v10.00: This button will open the Macro Recording Settings window (see section below).

Include Dex.ini Settings File

This checkbox specifies the default setting for ScreenShot.

Include User Dex.ini Settings File

This checkbox specifies the default setting for ScreenShot.

Include Current Launch File

This checkbox specifies the default setting for ScreenShot.

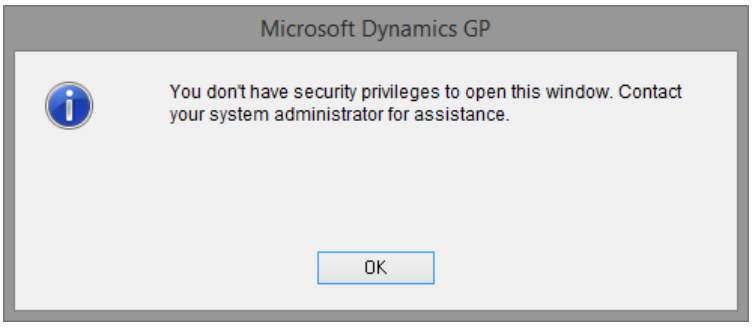
Automatic Open Mode

This option controls whether the Security Profiler window should automatically open when a security error occurs. The options are Do not open Automatically, Open on Errors only; and Open on Error & Warnings.

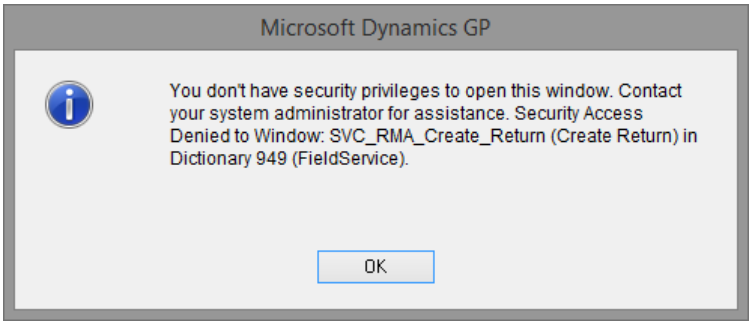
An Error is a situation that will cause the application to generate a dictionary not loaded or permission denied error dialog. A Warning is a situation where no error dialog will be generated, but the resources defined in the settings will not be opened as expected.

Disable updating Security Privilege warning to include form name

This option controls whether the Security Privilege warning dialog (screenshot below) includes additional information about the resource for which security access has been denied.



By default, once the Support Debugging Tool is installed, additional information will be included on the dialog (screenshot below). This will help administrators identify the issues even if the Security Profiler window is not in use.



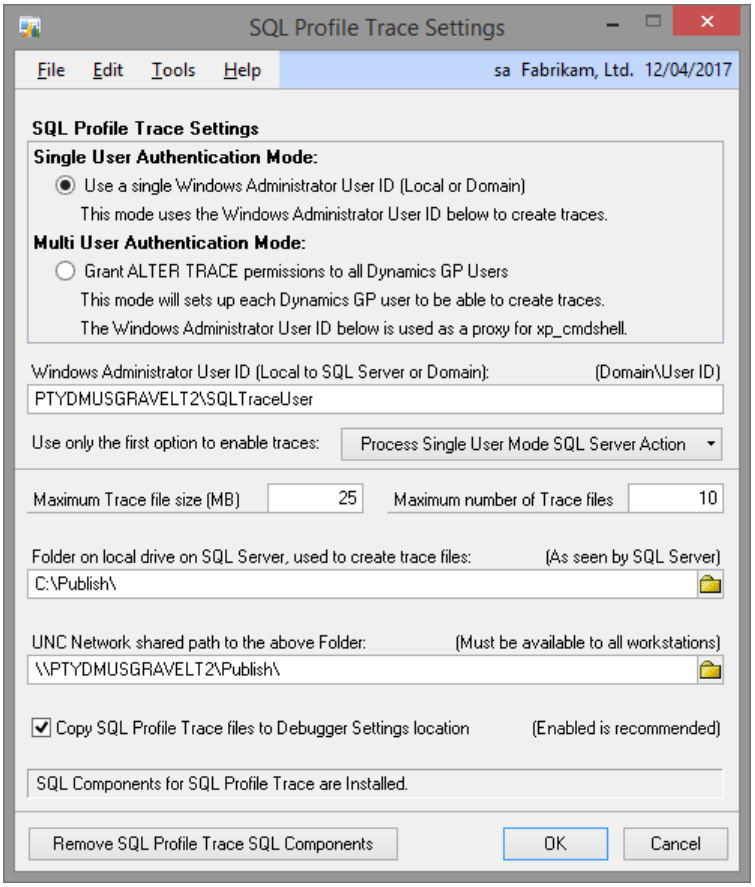
Enabling this option will disable the additional information and revert back to the standard dialog.

SQL Profile Trace Settings

The SQL Profile Trace Settings window contains all the options to enable SQL Profile Tracing and create the database objects needed.

The SQL Profile Tracing functionality of the Support Debugging Tool creates a series of SQL Stored Procedures in the DYNAMICS system database:

- MBS_SQL_Tracing_API
- MBS_SQL_Tracing_API_1 (Small)
- MBS_SQL_Tracing_API_2 (Medium)
- MBS_SQL_Tracing_API_3 (Large)
- MBS_SQL_Tracing_API_4 (Performance)
- MBS_SQL_Tracing_API_5 (Other)
- MBS_SQL_Tracing_Read
- MBS_SQL_Tracing_Version

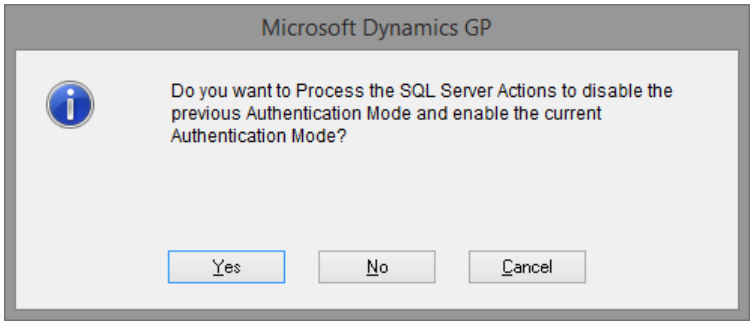


The following is a description of the individual fields on the window:

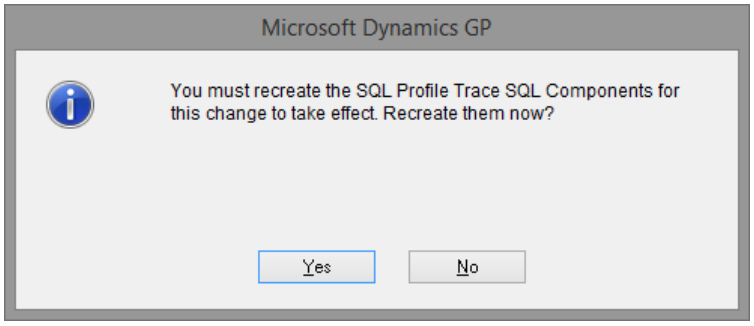
Single User Authentication Mode

Single User Authentication Mode uses a single Windows user to create the SQL Profile Traces. This is the preferred Authentication Mode as it does not require individual users to have their privileges elevated.

If the Authentication Mode is already enabled and you change the setting, you will receive a dialog to process the necessary changes at the SQL Server. It is recommended that you allow the system to make the changes.



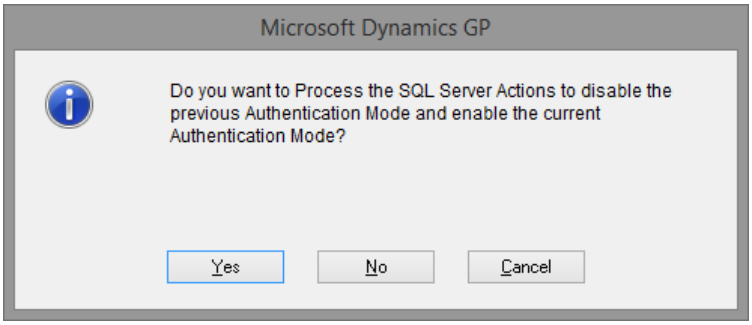
If the SQL Profile Trace SQL Components are already created and you change the setting, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.



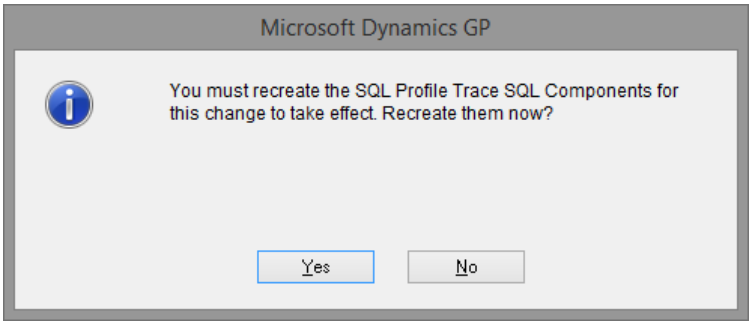
Multi User Authentication Mode

Multi User Authentication Mode uses the individual Dynamics GP users to create the SQL Profile Traces and only uses the Windows user as a proxy for the xp_cmdshell command. Using this mode will elevate individual users' rights to allow them to create traces.

If the Authentication Mode is already enabled and you change the setting, you will receive a dialog to process the necessary changes at the SQL Server. It is recommended that you allow the system to make the changes.



If the SQL Profile Trace SQL Components are already created and you change the setting, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.



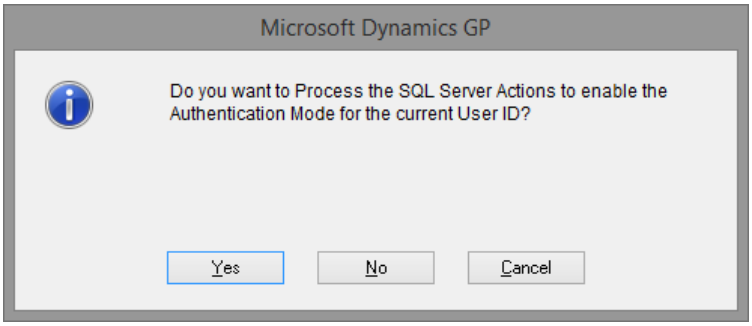
Windows Administrator User ID

Depending on Authentication Mode, this Windows User ID is used to create traces and/or as a proxy for the xp_cmdshell command.

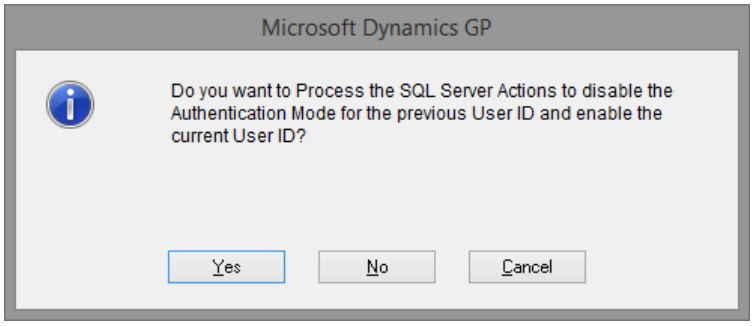
The user can be either a local user on the SQL Server machine or a domain user. The user must be added to the local administrator group on the SQL Server machine. It is recommended that the password for the user is set to not expire.

The user does not need to be manually added to SQL Server, the Support Debugging Tool will perform that step.

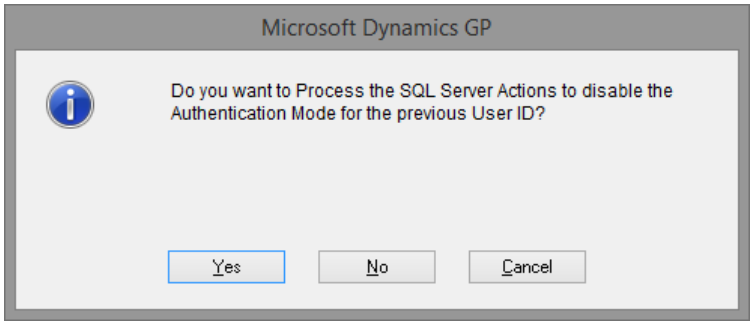
When you enter the User ID, you will receive a dialog asking to process the steps to enable the Authentication mode. It is recommended that you allow the system to make the changes.



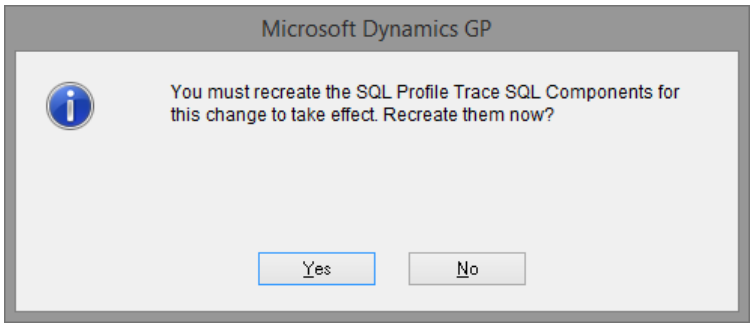
If the Authentication Mode is already enabled and you change the User ID, you will receive a dialog to process the necessary changes at the SQL Server. It is recommended that you allow the system to make the changes.



If you remove the User ID, you will receive a dialog asking to process the steps to disable the Authentication mode. It is recommended that you allow the system to make the changes.



If the SQL Profile Trace SQL Components are already created and you change the User ID, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.



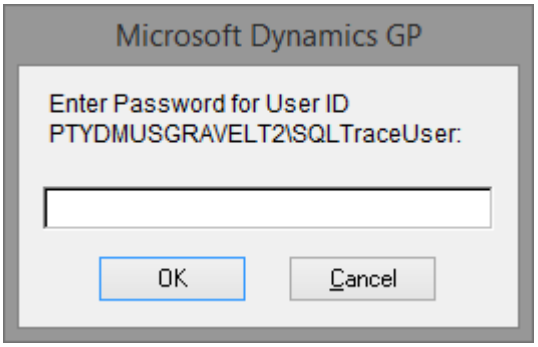
Process Single User Mode SQL Server Action

There are seven setting changes required on SQL Server to allow Single User Authentication Mode to work. This button allows the steps to enable and disable the settings to be run individually or as one action.

As the system already prompts for these actions to be executed automatically, you would not normally need to manually run the actions using this button.

If a new user is added to Microsoft Dynamics GP, you will need to run the Grant IMPERSONATE permission to all users option again to allow the new user to be able to create traces.

The Enable xp_cmdshell proxy account with User ID option will ask for the password for the Windows Administrator User ID.



The password is not validated at this time. If it is not entered correctly, it will prevent the SQL Profile Trace File being copied to the Debugger Settings folder when the trace is stopped. The error will show in the Debugger_<User>_<Company>.log file.

As each step is processed a Desktop Alert is displayed to show that the action completed.

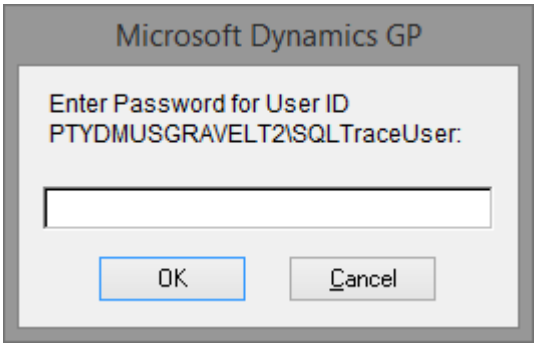
Process Multi User Mode SQL Server Action

There are four setting changes required on SQL Server to allow Multi User Authentication Mode to work. This button allows the steps to enable and disable the settings to be run individually or as one action.

As the system already prompts for these actions to be executed automatically, you would not normally need to manually run the actions using this button.

If a new user is added to Microsoft Dynamics GP, you will need to run the Grant IMPERSONATE permission to all users option again to allow the new user to be able to create traces.

The Enable xp_cmdshell proxy account with User ID option will ask for the password for the Windows Administrator User ID.



The password is not validated at this time. If it is not entered correctly, it will prevent the SQL Profile Trace File being copied to the Debugger Settings folder when the trace is stopped. The error will show in the Debugger_<User>_<Company>.log file.

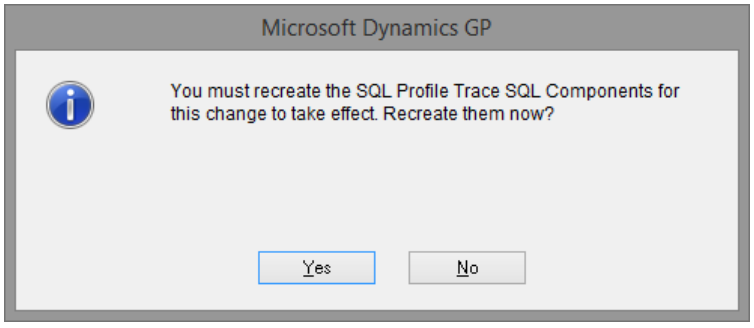
As each step is processed a Desktop Alert is displayed to show that the action completed.

Maximum Trace file size

Use this setting to control the maximum size a SQL Profile Trace file can get to before a new file is created. The default value for this field is 25 MB.

If you set the field back to zero, it will restore the default values for Maximum Trace file size and Maximum number of Trace files.

If the SQL Profile Trace SQL Components are already created and you change this setting, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.

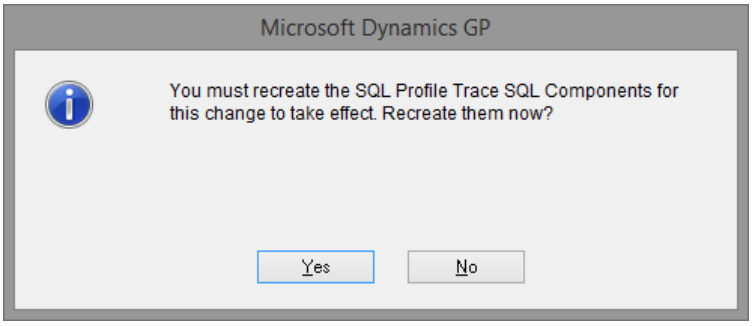


Maximum number of Trace files

Use this setting to control the number of trace files created by the SQL Profile Trace. As the trace file reaches the Maximum Trace file size a new trace file will be created with a numbered suffix added to the filename. This setting controls how many of the individual trace files are kept and will delete the oldest trace files as new ones are created. The default value for this field is 10.

If you set the field back to zero, the trace will only create a single file which will grow in size until the trace is stopped.

If the SQL Profile Trace SQL Components are already created and you change this setting, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.

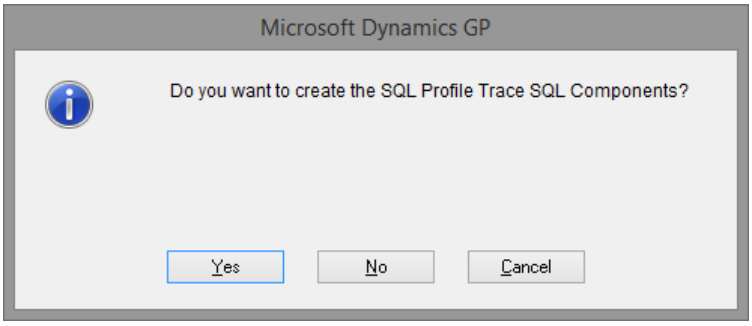


Folder on local drive on SQL Server

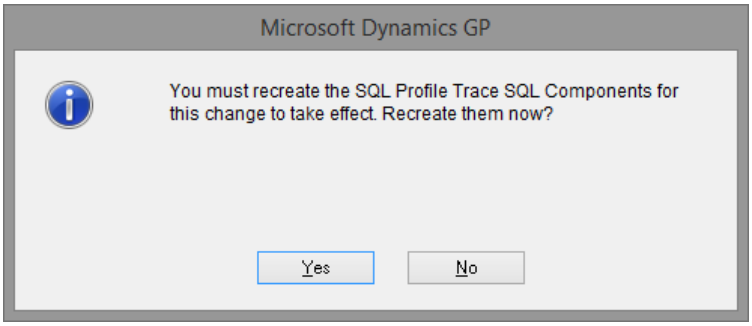
This is the path to a folder that is local to SQL Server that is to be used as a temporary location for SQL Profile Trace files while they are being created.

The folder must use a path that is valid as seen from the SQL Server machine. All Microsoft Dynamics GP Users as well as the Windows Administrator User ID must have Full Control rights to this folder.

When you enter the path, you will receive a dialog asking to create the SQL Profile Trace SQL Components (stored procedures). It is recommended that you allow the system to make the changes.



If the SQL Profile Trace SQL Components are already created and you change the path, you will receive a dialog to recreate them. It is recommended that you allow the system to make the changes.



UNC Network shared path to above Folder

This is the path to the local folder on the SQL Server from the previous field once it has been shared on the network.

The folder must be shared so that all Microsoft Dynamics GP Users as well as the Windows Administrator User ID have Full Control rights to this folder.

This path is used after the SQL Profile Trace is created to copy the trace files from the temporary location on the SQL Server to the Debugger Settings location.

Copy SQL Profile Trace files to Debugger Settings location

This checkbox can be used to control where the SQL Profile Trace files are copied from the temporary location on the SQL Server to the Debugger Settings location.

It is recommended that this setting is enabled.

Create SQL Profile Trace SQL Components

This button can be used to manually create the SQL Profile Trace SQL Components (stored procedures) on the SQL Server.

Remove SQL Profile Trace SQL Components

This button can be used to manually remove the SQL Profile Trace SQL Components (stored procedures) on the SQL Server.

Macro Recording Settings

The Macro Recording Settings window contains the option to enable Macro Recording for v10.00.

The following is a description of the individual fields on the window:

Disable Mouse Wheel

This option will update the MouseWheel Dex.ini setting to disable Mouse Wheel Scrolling in the application. This helps with an issue with Macro Recording in v10.00.

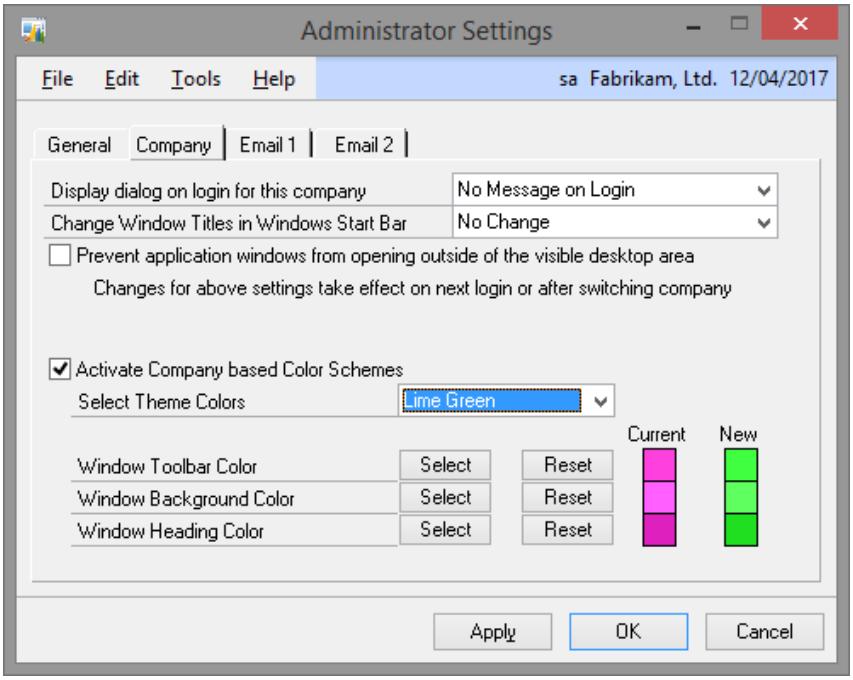
A restart of the application is required for this setting to take effect.

Macro Recording can only work when a single instance of Microsoft Dynamics GP is running on a workstation, or if multiple instances are running, Macro Recording will only work on the first instance launched.

The Disable Mouse Wheel option is also available from the Other tab of the Dex.ini Settings window.

Company Tab

The Company tab contains settings for controlling visual indicators to help users identify the company they are currently accessing. These visual indicators are designed to prevent the accidental entry of data into an incorrect company.



The following is a description of the individual fields on the window:

Display dialog on login for this company

This setting can be used to display a dialog after a user has logged into a company. The settings are **No Message on Login** or one of the two choices below:

Test Company Message: "This company is set up for testing only. Do not use this company when processing live data."

Historical Company Message: "This company is used for storing historical information only. Do not use this company when processing current-year data."

This feature is providing a user interface to the existing dialog functionality as described in Knowledge Base (KB) article 885542:

http://support.microsoft.com/kb/885542

This feature takes effect on next login or after switching company.

Change Window Titles in Windows Start Bar

This setting can be used to prefix the window titles as seen in the Windows Start Bar with the User ID and/or Company ID.

This feature is useful when running multiple instances of Microsoft Dynamics GP on a single workstation. It allows users to easily identify which window belongs to each instance of the application by displaying the User ID and/or Company ID in the Windows Start Bar.

This feature takes effect on next login or after switching company.

Prevent application windows from opening outside of the visible desktop area

This setting checks the location of all windows as they open and if they will not be in the visible desktop, their position will be adjusted to make sure they are fully visible.

This feature takes effect on next login or after switching company.

Activate Company based Color Schemes

This option can be used to change the background colors for each company to allow an effective visual cue as to the company currently being used.

Once the option is activated, the colors for the Window Toolbar Color, Window Background Color, and Window Heading Color can be selected using the Select Theme drop down list which offers 16 preset themes. The colors can also be manually selected with the Select Buttons. The Reset Buttons can be used to restore the default colors.

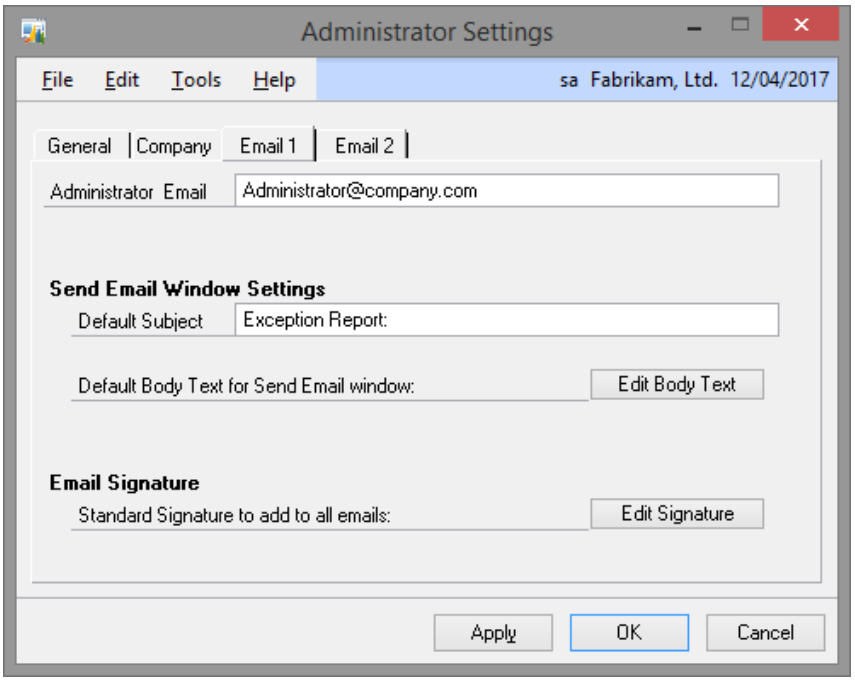
The current color scheme and the new color scheme are displayed. When the Apply Button or OK Button is clicked, the new color scheme will be applied.

The Color Scheme data is stored at both the Company and System level. This design allows for live company databases to be copied into a test company database while maintaining the correct color schemes.

When running on the Web Client, the Change Window Titles in Windows Start Bar and Activate Company based Color Schemes options are disabled as they are not supported.

Email 1 Tab

The Email 1 tab contains settings for defining the default values when sending emails from the Support Debugging Tool.



The following is a description of the individual fields on the window:

Administrator Email

This field can be used to specify the default To email address(es) when sending emails.

Email addresses can be in the following formats and multiple addresses should be separated by a semi-colon (;):

- name@domain.com
- Full Name<name@domain.com>
- Full Name (when in Microsoft Outlook mode only)

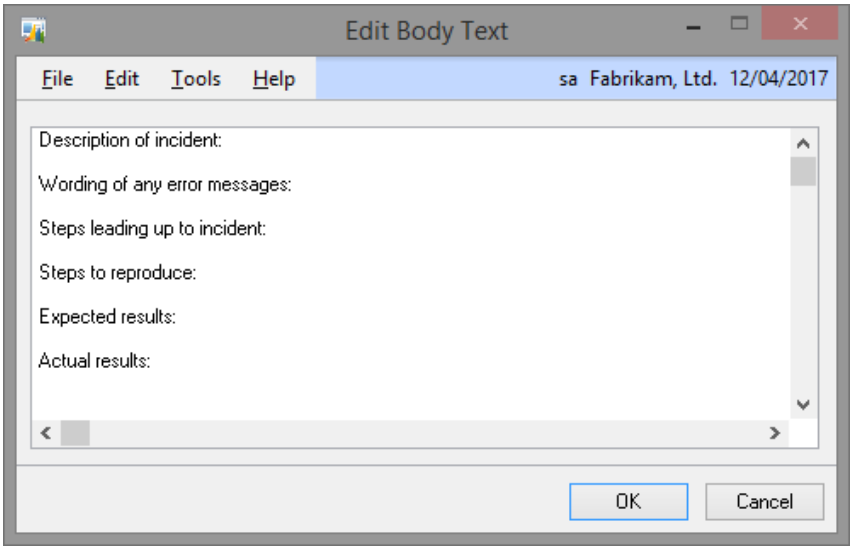
Default Subject

This field can be used to specify the default Subject line for the Send Email window.

Default Body Text for Send Email window

This button can be used to specify the default Body Text line for the Send Email window. This can be used to create a standard template for reporting issues to the system administrator.

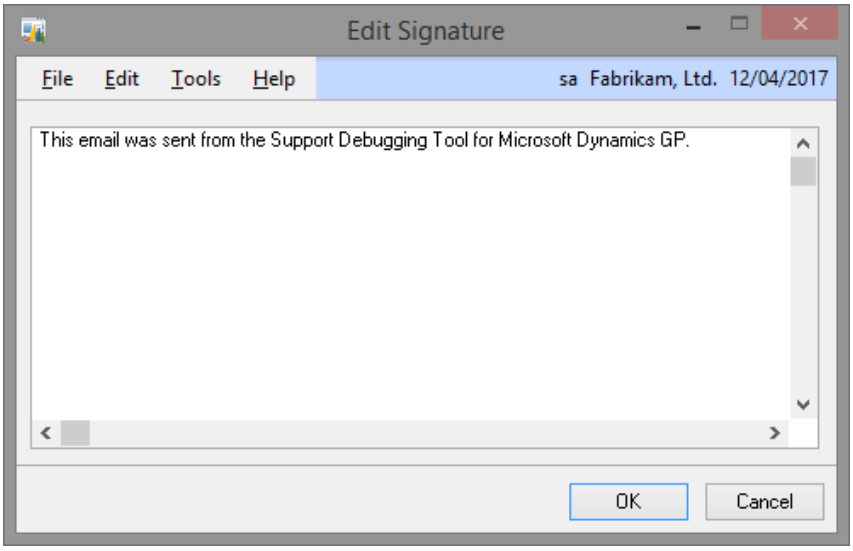
Click on the Edit Body Text Button to open the Edit Body Text window.



Standard Signature to add to all emails

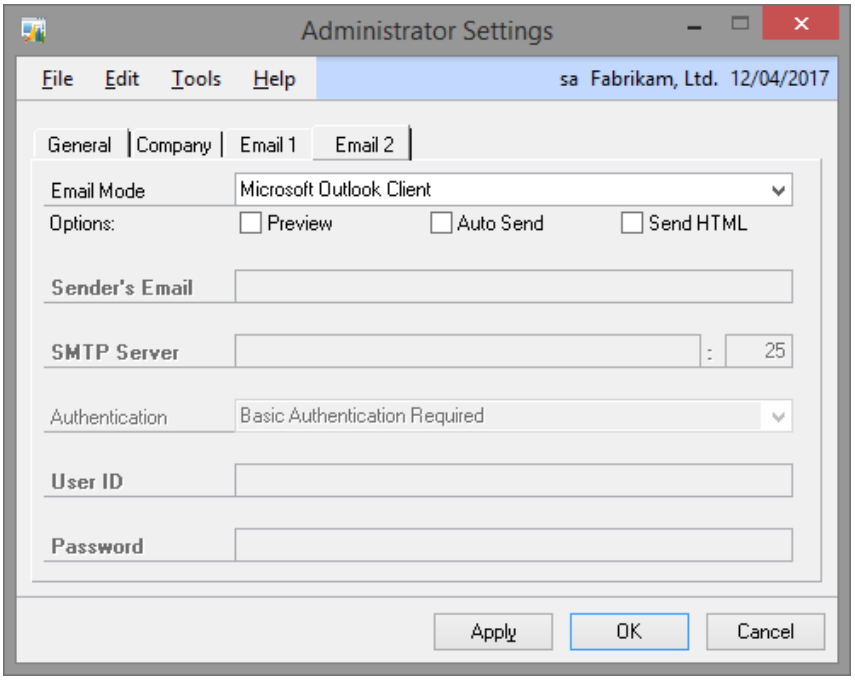
This button can be used to create a standard signature to add to the bottom of all emails sent from the Support Debugging Tool. If no signature is defined, the text in the screenshot below will be used.

Click on the Edit Signature Button to open the Edit Signature window.



Email 2 Tab

The Email 2 tab contains settings for email engine of the Support Debugging Tool. Use this tab the transport protocols and security settings for sending emails can be configured.



The following is a description of the individual fields on the window:

Email Mode

This field can be used to select whether the email engine is using a Microsoft Outlook Client (default setting) or a SMTP Server via CDO (Collaboration Data Objects) to send emails. Using SMTP instead of Outlook is useful for a Terminal Server environment where it is unlikely that an Outlook client is installed and set up on the Terminal Server.

For version 11.0 (for Microsoft Dynamics GP 2010) and later, you can also select to use any MAPI Compliant Client for sending emails. This will work for email clients other than Outlook if they are MAPI compliant. As Outlook is MAPI compliant, this mode also works for Outlook.

For version 12.0 (for Microsoft Dynamics GP 2013) and later, you can also select to use Exchange Web Services for sending emails. This will work directly with the Exchange Server and so does not require an email clients to be installed.

When running on the Web Client, the Microsoft Outlook Client email mode is not supported. It is recommended to use the SMTP or Exchange modes which do not require an email client.

Preview

This option controls if the Send Email window is displayed whenever an email is sent.

Auto Send

This option controls if the email is automatically sent when an email is submitted. If Preview is unchecked, the email is submitted immediately, or if Preview is selected the email is submitted when the Send Button is clicked.

When using SMTP mode, Auto Send is always enabled. When using Outlook mode, this option controls whether the email is shown in the Outlook client before it is sent, without Auto Send the user will need to click the Send button in Outlook.

Send HTML

This option controls whether emails generated in the Support Debugging Tool are sent as plain text or as HTML.

Sender's Email

This field must contain a single valid email address for use as the sender's email address when in SMTP mode. It is recommended to create a new email address for emails sent from Microsoft Dynamics GP.

The email address can be in the following formats:

- name@domain.com
- Full Name<name@domain.com>

SMTP Server

This field defines the SMTP Server's address. It can be specified as a name or as an IP address.

SMTP Server Port

This field defines the SMTP Server Port to use, the default value is 25.

Authentication

This drop down list specifies what level of authentication is required to send emails via the SMTP Server. The options are:

- No Authentication Required
- Basic Authentication Required
- NTLM Authentication Required
- Basic Authentication & SSL Required
- NTLM Authentication & SSL Required

You can specify whether Basic or NTLM (Windows NT LAN Manager) Authentication is to be used and whether SSL (Secure Sockets Layer) should be used.

User ID

This field contains the user ID to login into the SMTP Server with. This would normally be the user ID associated with the Sender's Email defined above.

Password

This field contains the password to login into the SMTP Server with. This would normally be the password associated with the Sender's Email defined above.

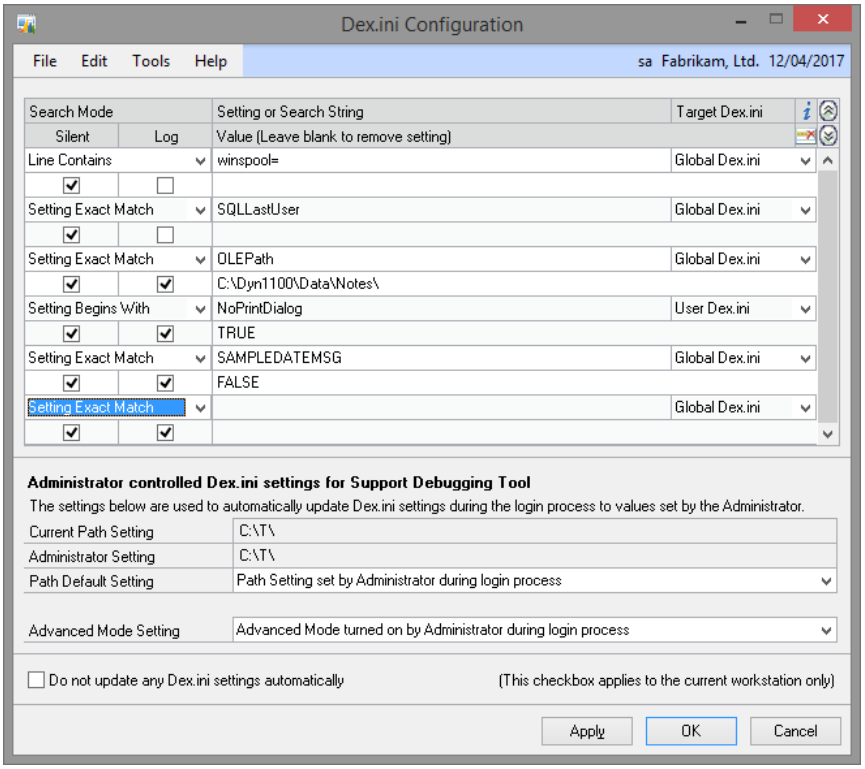
Dex.ini Configuration

You can open the Dex.ini Configuration window by selecting Dex.ini Configuration from the Options button drop list on the main window. This is an Advanced Mode Feature; please see the beginning of the chapter for instructions on enabling Advanced Mode.

The Dex.ini Configuration window can be used to automate changes to Dex.ini settings for all workstations in the system.

It can also be used by an Administrator to control the location for the Support Debugging Tool's setup file (Debugger.xml) and if Advanced Mode should be active.

The Administrator controlled Dex.ini settings store their configuration in the syUserDefaults (SY01402) table in the DYNAMICS SQL Database. On login, the configuration is checked and the Dex.ini settings on the current workstation are updated if necessary.



The following is a description of the individual fields on the window:

Settings List

This list contains the Dex.ini settings to be checked on login. The setting can be specified with an exact value (this is needed to add a new setting), or can be specified using a "contains" or "begins with" search. The search can be applied against the Dex.ini settings listed in the Dex.ini file (i.e. left of the = sign) or against the entire line in the file. Using a search can find and update multiple lines in the Dex.ini file if more than one setting or line meets the Search Mode and Search String criteria.

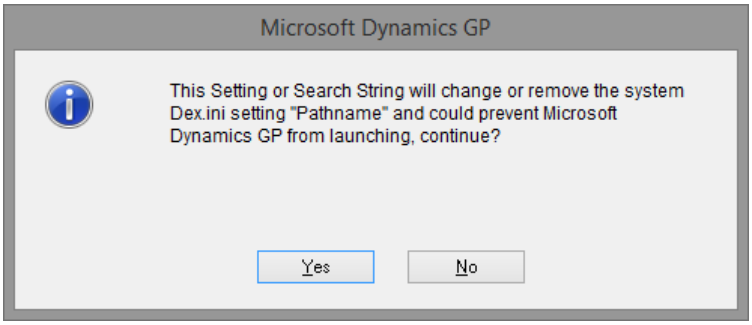
The Silent checkbox should be checked (default) if the Dex.ini setting should be updated without asking the user.

The Log checkbox should be checked (default) if the Dex.ini setting changes should be recorded in the Debugger log files.

The Value field contains the value to change the Dex.ini setting to. Leaving this field blank will remove the Dex.ini setting from the Dex.ini file.

The Target Dex.ini field allows the selection of whether this setting should be applied to the Global Level Dex.ini file (default), to the User level Dex.ini file, or to Both Dex.ini files.

Before the line in the Setting List is saved, it is checked for possible damaging settings and if they exist an additional confirmation is required. Possible changes to the following Dex.ini settings are detected: Pathname, Initial, Synchronize, Workstation, Workstation2, FileHandler, DatabaseType.



Current Path Setting

This field displays the current location for the Debugger.xml file as defined by the Pathname location for Debugger Setup files, exports and logs setting in the Dex.ini Settings window.

Administrator Setting

This field displays the location for the Debugger.xml file as currently set by the Administrator. This value comes from the syUserDefaults (SY01402) table in the DYNAMICS SQL Database.

Path Default Setting

This field can be used by the Administrator to force all workstations with the Support Debugging Tool installed to use a Debugger.xml setup file from a specified shared location. This avoids needing to visit individual workstations to change the Pathname location for Debugger Setup files, exports and logs setting in the Dex.ini Settings window manually. It will update the MBS_Debug_Path Dex.ini setting on login.

Many features of the Support Debugging Tool work better when a single Debugger.xml setup file stored in a shared location is used. This feature automates the configuration to ensure all workstations are pointing to the correct location.

Advanced Mode Setting

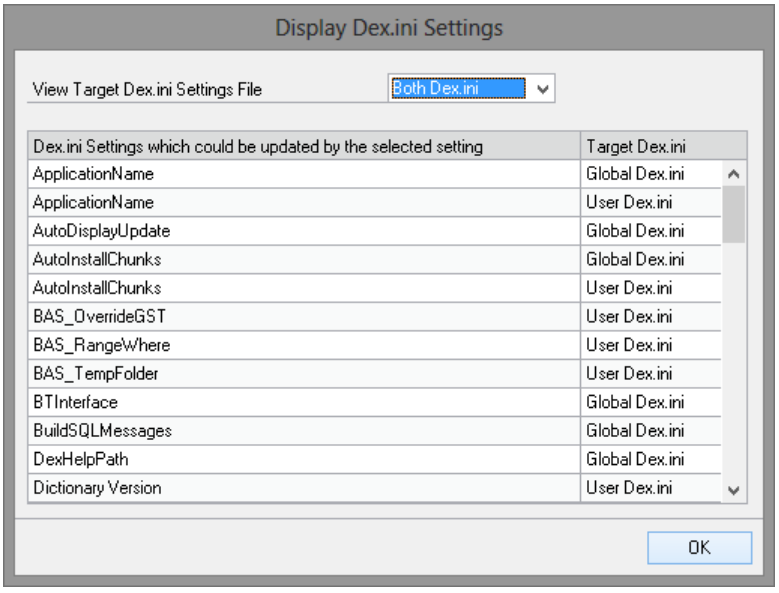
This field can be used by the Administrator to control whether the Advanced Mode features will be available. It can be set to leave the setting controlled by the Enable Debugger Advanced Mode Features setting in the Dex.ini Settings window, or to override this setting to turn it on or off during login. . It will update the MBS_Debug_Mode Dex.ini setting on login.

Do not update any Dex.ini settings automatically

This field can be used on individual workstations to prevent the Support Debugging Tool from automatically changing any Dex.ini Settings. It will update the MBS_Debug_ConfigurationOverride Dex.ini setting. This can be useful on test or administration workstations which might not want their Dex.ini settings changed.

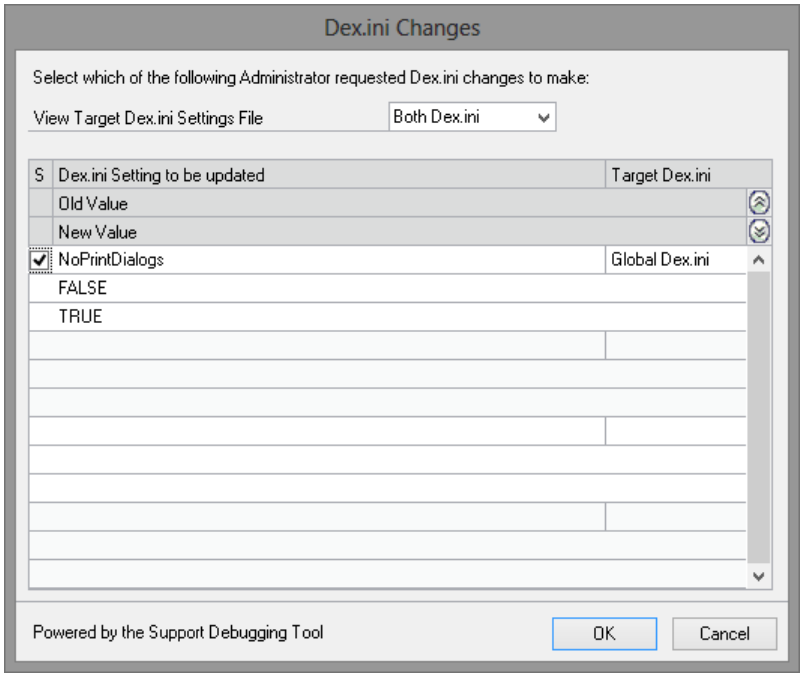
The Apply Button can be used to save the changes to the setup files without closing the window.

Click the Info button to display a list of Dex.ini settings that can be changed by the current Search Mode and Search String criteria. The Display Dex.ini Settings window will open.



Setting changes specified in the Dex.ini Configuration window are checked against the Dex.ini file when a user logs in. The system looks for settings which differ from the specified values. If the change is marked as Silent, the setting will be automatically updated.

If any changes need to be made where the Silent checkbox is not selected, then the user will be presented with a dialog asking them to confirm which changes should be applied.



If a user opts to deselect a Dex.ini change that setting will be displayed again when the user logs in again or changes company.

Chapter 5: Dex.ini Settings

Support Debugging Tool Settings

The Support Debugging Tool uses the Dex.ini file to store a number of settings. The default location for the Dex.ini file is in the data subfolder beneath the Microsoft Dynamics GP application folder. These settings are explained below:

MBS_Debug_Mode

This setting can be TRUE or missing, and denotes whether Advanced Mode is active. The default for this setting is missing, which means that Standard Mode is being used.

MBS_Debug_Path

This setting can point to a location for the Debugger.xml setup file. The default for this setting is missing, which means that the Debugger.xml file will be stored in the data subfolder beneath the Microsoft Dynamics GP application folder.

MBS_Debug_SetupMode

This setting can be TRUE or missing, and denotes whether Setup Mode is enabled. The default for this setting is missing, which means that Setup Mode is not enabled.

MBS_Debug_AutoOpen

This setting can be TRUE or missing, and denotes whether the Support Debugging Tool window should open automatically after logging into a company.

MBS_Debug_Version

This setting tracks the last used version of the Support Debugging Tool on the current workstation.

MBS_Debug_LogOnStartup

This setting can be TRUE or missing, and denotes whether to automatically start logging when Microsoft Dynamics GP is next started. The default for this setting is missing, which means that the feature is disabled.

MBS_Debug_RuntimeCheck

This setting can be FALSE or missing, and denotes whether the Runtime Engine version and build information is checked for compatibility. The default for this setting is missing, which means that the version and build will be checked.

MBS_Debug_ShowRuntime

This setting can be TRUE or missing, and denotes whether the Runtime Engine is shown when creating Dexterity sanScript scripts in either the Automatic Debugger Mode Setup window or the Runtime Execute window. If this setting is enabled, the Resource Explorer window and Table Explorer window will also display resources from the Runtime Engine dictionary DEX.DIC. The default for this setting is missing, which means the runtime engine is not displayed.

MBS_Debug_ConfigurationOverride

This setting can be TRUE or missing, and denotes whether the Support Debugging Tool is allowed to automatically update Dex.ini Settings for this workstation as defined in the Dex.ini Configuration window. Set to TRUE to prevent any updates.

MBS_Debug_LogAppDetails

This setting can be TRUE or missing, and denotes whether the Support Debugging Tool should log an entry into the Debugger_<User>_<Company>.log file each time a user logs into a company.

SQLLogRename

This setting can be used to automatically rename the DEXSQL.LOG file each day. The value will be the date of the last rename in the form YYYYMMDD.

SQLLastCompany

This setting is used to automatically store the last Company ID selected for the current workstation. This allows the company selection drop down list to be defaulted to the last company used.

DefaultLastCompany

This setting can be FALSE or missing and is used to disable the automatic defaulting of the last company used when logging into Microsoft Dynamics GP or switching companies.

MBS_Debug_WinDebugger

This setting is used to store the last window size, position, and state for main the Support Debugging Tool window.

MBS_Debug_WinDebuggerSetup

This setting is used to store the last window size, position, and state for the Support Debugging Tool Setup window.

MBS_Debug_WinDebuggerStatus

This setting is used to store the last window size, position, and state for the Automatic Debugger Mode Status window.

MBS_Debug_WinResourceInformation

This setting is used to store the last window size, position, and state for the Resource Information window.

MBS_Debug_WinSecurityProfiler
This setting is used to store the last window size, position, and state for the Security Profiler window.

MBS_Debug_WinSecurityInfo
This setting is used to store the last window size, position, and state for the Security Information window.

MBS_Debug_WinSecurityInfoResource
This setting is used to store the last window size, position, and state for the Security Information Resources window.

MBS_Debug_WinDictionaryControl
This setting is used to store the last window size, position, and state for the Dictionary Control window.

MBS_Debug_WinXMLTableExport
This setting is used to store the last window size, position, and state for the XML Table Export window.

MBS_Debug_WinXMLTableImport
This setting is used to store the last window size, position, and state for the XML Table Import window.

MBS_Debug_WinRuntimeExecute
This setting is used to store the last window size, position, and state for the Runtime Execute window.

MBS_Debug_WinSQLExecute
This setting is used to store the last window size, position, and state for the SQL Execute window.

MBS_Debug_WinConfigurationExportImport
This setting is used to store the last window size, position, and state for the Configuration Export/Import window.

MBS_Debug_WinConfigurationMaintenance
This setting is used to store the last window size, position, and state for the Configuration Maintenance window.

MBS_Debug_WinScreenShot
This setting is used to store the last window size, position and state for the ScreenShot window.

MBS_Debug_WinAdminSettings
This setting is used to store the last window size, position and state for the Administrator Settings window.

MBS_Debug_WinConfigSettings
This setting is used to store the last window size, position and state for the Dex.ini Configuration window.

MBS_Debug_WinSendEmail
This setting is used to store the last window size, position and state for the Send Email window.

MBS_Debug_WinResourceExplorer
This setting is used to store the last window size, position and state for the Resource Explorer window.

MBS_Debug_WinMenuExplorer
This setting is used to store the last window size, position and state for the Menu Explorer window.

MBS_Debug_WinTableExplorer
This setting is used to store the last window size, position and state for the Table Explorer window.

MBS_Debug_WinReportExplorer
This setting is used to store the last window size, position and state for the Report Explorer window.

MBS_Debug_WinObjectExplorer
This setting is used to store the last window size, position and state for the Security Object Explorer window.

MBS_Debug_WinTableLookup
This setting is used to store the last window size, position and state for the Table Lookup window.

MBS_Debug_WinFieldLookup
This setting is used to store the last window size, position and state for the Field Lookup window.

MBS_Debug_WinKeyLookup
This setting is used to store the last window size, position and state for the Table Keys Lookup window.

MBS_Debug_Automate_File
This setting is used by Microsoft Support to provide the full path or filename to a Diagnostics configuration settings file to be loaded after logging into Microsoft Dynamics GP. If the full path is not provided, the file can be located in the Debugger logs folder, the application's Data folder, or the folders where the DYNAMICS.EXE or DYNAMICS.SET are located. Trigger IDs, Script IDs and Profile IDs loaded with this option should be prefixed with a tilde (~) character. By default, this setting is removed after use.

MBS_Debug_Automate_Script
This setting is used by Microsoft Support to provide the Script ID for a Runtime Execute Diagnostics script to be executed after logging into Microsoft Dynamics GP. The Script ID executed with this option should be prefixed with a tilde (~) character. By default, this setting is removed after use.

MBS_Debug_Automate_Status

This setting is used by Microsoft Support to control to the behavior of the Diagnostics automation features of the Support Debugging Tool. By default, this setting is removed after use. The valid flags (which can be added together) are as follows:

1 – Do not delete settings loaded from configuration settings file.
2 – Do not delete Diagnostics Automation Dex.ini settings.
4 – Do not delete configuration settings XML file.
8 – Do not display "Please Wait" dialogs while loading settings file.

MBS_Debug_DisableSplitters

This setting can be used disable the splitter functionality on the Security Information and Resource Explorer windows. Set it to TRUE to disable the splitters.

MBS_Debug_VBADisableReset

This setting is used by the Support Debugging Tool to signify that Visual Basic for Applications (VBA) should be re-enabled after one login.

MBS_Debug_VSTDisable

This setting is used by the Support Debugging Tool to disable Visual Studio Tools Addins on login.

MBS_Debug_VSTDisableReset

This setting is used by the Support Debugging Tool to signify that Visual Studio Tools Addins should be re-enabled after one login.

MBS_Debug_SkipVersionChecks

This setting is used to allow the Support Debugging Tool to run on a different version of Dexterity than the one it was built for. It is to be used when testing the Support Debugging Tool on upcoming versions of Microsoft Dynamics GP.

System Settings

The Support Debugging Tool can also manipulate the values of certain system settings stored in the Dex.ini settings file:

SQLLogSQLStmt
This setting can be TRUE or FALSE and controls whether statements Microsoft Dynamics GP sends to the SQL Server are logged to the DEXSQL.LOG file by default.

SQLLogODBCMessages
This setting can be TRUE or FALSE and controls whether ODBC messages returned from the SQL Server back to the Microsoft Dynamics GP client are logged to the DEXSQL.LOG file by default.

SQLLogAllODBCMessages
This setting can be TRUE or FALSE and controls whether all ODBC messages returned from the SQL Server back to the Microsoft Dynamics GP client are logged to the DEXSQL.LOG file by default.

SQLLogPath
This setting can be used to change the default location of the DEXSQL.LOG file.

ScriptDebugger
This setting can be TRUE or FALSE and controls whether the Dexterity Debug menu is available in runtime mode.

ScriptDebuggerProduct
This setting contains the Dexterity Product ID that will be used to set the initial context of the Debug menu. The default value is 0 for Dynamics.

ShowDebugMessages
This setting can be TRUE or FALSE and controls whether internal debug message dialogs are displayed when the Debug Menu is enabled. It is recommended that this should be set to FALSE for production systems.

ScriptLogEnhanced
This setting can be TRUE or FALSE and controls whether the enhanced Dexterity Script Log features added in v10.00 service pack 4 and later are enabled. Enabling this option adds time stamps and flagging of background processes to the script log. The default value is set to TRUE by the Support Debugging Tool.

ApplicationName
This setting contains the name to be shown on the title bar when the application first launches. If this value is not defined, the name in the title bar will default to “Dexterity Runtime”.

AutoInstallChunks
This setting allows chunks to be included without prompting when Microsoft Dynamics GP is launched.

AllowWrongDex
This setting allows a mismatched Dex.dic and Dexterity Runtime version to be used. It is not recommended to use this option.

SkipVersionChecks
This setting allows Microsoft Dynamics GP to launch without errors even when the dictionary version numbers do not match the version information in the database. It is not recommended to use this option.

SAMPLEDATEMSG
This setting prevents the Fabrikam sample company date warning dialog from opening when logging in.

SQLLoginCompatibilityMode
This setting controls if Microsoft Dynamics GP continues to use SQL Login Compatibility Mode after v10.00 service pack 2 and later is installed.

ExportOneLineBody
This setting controls whether text report body sections are exported as a single line in the export file.

ExportLinesPerPage
This setting controls the number of lines to include on a report page when it is exported to a file.

ExportPDFLinesPerPage
This setting controls the number of lines to include on a report page when it is exported to a PDF file.

DebugRW
This setting is used to configure the Report Writer to create a debugging log file named DebugRW.txt that will appear in the data subfolder beneath the Microsoft Dynamics GP application folder.

SuppressChangeDateDialog
This setting prevents the Change Date dialog from being displayed at midnight. If used, the User Date will not change at midnight.

ShowAdvancedMacroMenu
This setting will enable the Advanced Macro Menu from the Tools >> Macro menu.

ShowAllMenuItems
This setting will leave all menu items showing even if the module is not installed or if access is denied.

SuppressSound
This setting disables all sound from the Microsoft Dynamics GP application.

QueueMoreInfo
This setting can be used to enable the More Info button on the Process Monitor window.

MouseWheel
This setting can be used to disable Mouse Wheel scrolling in the application. This helps avoid an issue with Macro Recording in v10.00.

DebugFonts
This setting can be used to enable logging of Report Writer selections to the DebugLog.txt file.

TPELogging
This setting can be used to enable logging of the internals of the Template Processing Engine (TPE) for word templates.

KeepTemplateTempFiles
This setting can be used to disable the automatic removal of the temporary files used when the Template Processing Engine (TPE) runs.

VBADisable
This setting can be used to disable Visual Basic for Applications when restarting Microsoft Dynamics GP.

EnableServerDropDown
This setting can be used to disable the Data Source Server selection when logging into Microsoft Dynamics GP.

DefaultLastUser
This setting can be used to disable the defaulting of the last user used when logging into Microsoft Dynamics GP.

EnableWCRibbons
This setting can be used to disable the GP 2013 R2 or later Web Client style ribbons in the desktop client for the current workstation.

Script Editor Settings

The Support Debugging Tool uses some of the Dexterity Script Editor Dex.ini settings:

ScriptEditorSyntaxColoring
This setting stores whether Syntax Highlighting is enabled.

ScriptKeywordColor
This setting stores the color selection for keywords.

ScriptIdentifierColor
This setting stores the color selection for identifiers.

ScriptNumberColor
This setting stores the color selection for numbers.

ScriptStringColor
This setting stores the color selection for strings.

ScriptCommentColor
This setting stores the color selection for comments.

ScriptOperatorColor
This setting stores the color selection for operators.

ScriptErrorColor
This setting stores the color selection to display Scripting Highlighting errors.

ScriptEditorFontName
This setting stores the font style section.

ScriptEditorFontSize
This setting stores the font size section.

Chapter 6: Helper Functions

The Support Debugging Tool has a number of helper functions which can be used to make cross-dictionary Dexterity sanScript simpler to write. The Helper Function Assistant window will automatically insert the code required to use these functions.

Below are the details of the helpers available:

- *MBS_Get_Window_Value*
- *MBS_Set_Window_Value*
- *MBS_Get_Table_Value1*
- *MBS_Set_Table_Value1*
- *MBS_Get_Table_Value2*
- *MBS_Set_Table_Value2*
- *MBS_Get_Table_Value3*
- *MBS_Set_Table_Value3*
- *MBS_Get_Table_Value4*
- *MBS_Set_Table_Value4*
- *MBS_Runtime_Execute*
- *MBS_SQL_Check_Exists*
- *MBS_Export_SQL_Query_To_File*
- *MBS_Script_Load_Dex*
- *MBS_Script_Load_SQL*
- *MBS_Param_Set*
- *MBS_Param_Get*
- *MBS_Param_Del*
- *MBS_Param_DelAll*
- *MBS_Auto_Log*
- *MBS_Trigger_Start*
- *MBS_Trigger_Stop*
- *MBS_Email_API*

The Helper Function Assistant can also create template scripts for use with Runtime Execute and Report Writer Functions as described in the next chapter.

MBS_Get_Window_Value

This call is used to obtain the value of a window field from any open form in any dictionary.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Form_Name;
in string IN_Window_Name;
in string IN_Field_Name;
out anonymous field OUT_Field_Value;
out integer OUT_Status;
```

An example script is:

```
local integer l_status;
local string l_field;
call with name "MBS_Get_Window_Value" in dictionary 5261,
Dictionary, "Form", "Window", "Field", l_field, l_status;
if l_status = OKAY then
    warning str(l_field);
end if;
```

MBS_Set_Window_Value

This call is used to set the value of a window field from any open form in any dictionary. You have the option to also run the target field's change script.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Form_Name;
in string IN_Window_Name;
in string IN_Field_Name;
in anonymous field IN_Field_Value;
in boolean IN_Run_Flag;
out integer OUT_Status;
```

An example script is:

```
local integer l_status;
local string l_field;
l_field = "Value";
call with name "MBS_Set_Window_Value" in dictionary 5261,
Dictionary, "Form", "Window", "Field", l_field, true {run script},
l_status;
if l_status <> OKAY then
    warning str(l_status);
end if;
```

MBS_Get_Table_Value1

This call is used to obtain the value of a field located in any table in any dictionary using an index containing one field.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
out anonymous field OUT_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1;
l_key1 = "Value1";
call with name "MBS_Get_Table_Value4" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index},
                  "Key1", l_key1;
if l_status = OKAY then
    warning str(l_field);
end if;
```

MBS_Set_Table_Value1

This call is used to update the value of a field located in any table in any dictionary using an index containing one field. You can specify whether the creation of a new table record is allowed.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
in anonymous field IN_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in boolean IN_Allow_Add;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1;
l_key1 = "Value1";
l_field = "Value"
call with name "MBS_Set_Table_Value1" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index}, true
{allow add},
                  "Key1", l_key1;
if l_status <> OKAY then
   warning str(l_status);
end if;
```

MBS_Get_Table_Value2

This call is used to obtain the value of a field located in any table in any dictionary using an index containing two fields.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
out anonymous field OUT_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2;
l_key1 = "Value1";
l_key2 = "Value2";
call with name "MBS_Get_Table_Value4" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index},
                  "Key1", l_key1,
                  "Key2", l_key2;
if l_status = OKAY then
   warning str(l_field);
end if;
```

MBS_Set_Table_Value2

This call is used to update the value of a field located in any table in any dictionary using an index containing two fields. You can specify whether the creation of a new table record is allowed.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
in anonymous field IN_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in boolean IN_Allow_Add;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2;
l_key1 = "Value1";
l_key2 = "Value2";
l_field = "Value"
call with name "MBS_Set_Table_Value2" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index}, true
{allow add},
                  "Key1", l_key1,
                  "Key2", l_key2;
if l_status <> OKAY then
   warning str(l_status);
end if;
```

MBS_Get_Table_Value3

This call is used to obtain the value of a field located in any table in any dictionary using an index containing three fields.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
out anonymous field OUT_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
in string IN_Key_Name3;
in anonymous field IN_Key_Value3;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2, l_key3;
l_key1 = "Value1";
l_key2 = "Value2";
l_key3 = "Value3";
call with name "MBS_Get_Table_Value3" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index},
                  "Key1", l_key1,
                  "Key2", l_key2,
                  "Key3", l_key3;
if l_status = OKAY then
   warning str(l_field);
end if;
```

MBS_Set_Table_Value3

This call is used to update the value of a field located in any table in any dictionary using an index containing three fields. You can specify whether the creation of a new table record is allowed.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
in anonymous field IN_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in boolean IN_Allow_Add;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
in string IN_Key_Name3;
in anonymous field IN_Key_Value3;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2, l_key3;
l_key1 = "Value1";
l_key2 = "Value2";
l_key3 = "Value3";
l_field = "Value"
call with name "MBS_Set_Table_Value3" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index}, true
{allow add},
                  "Key1", l_key1,
                  "Key2", l_key2,
                  "Key3", l_key3;
if l_status <> OKAY then
   warning str(l_status);
end if;
```

MBS_Get_Table_Value4

This call is used to obtain the value of a field located in any table in any dictionary using an index containing four fields.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
out anonymous field OUT_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
in string IN_Key_Name3;
in anonymous field IN_Key_Value3;
in string IN_Key_Name4;
in anonymous field IN_Key_Value4;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2, l_key3, l_key4;
l_key1 = "Value1";
l_key2 = "Value2";
l_key3 = "Value3";
l_key4 = "Value4";
call with name "MBS_Get_Table_Value4" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index},
                  "Key1", l_key1,
                  "Key2", l_key2,
                  "Key3", l_key3,
                  "Key4", l_key4;
if l_status = OKAY then
   warning str(l_field);
end if;
```

MBS_Set_Table_Value4

This call is used to update the value of a field located in any table in any dictionary using an index containing four fields. You can specify whether the creation of a new table record is allowed.

All table and field names need to be the technical names and surrounded by single quotes if they contain a space. The status returned will contain the number of errors that occurred, a value of OKAY (zero) means the call was successful. The Key Name fields need to contain the technical names of the segment fields of the index being used.

The parameter list for this call is:

```
in integer IN_Prod_ID;
in string IN_Table_Name;
in string IN_Field_Name;
in anonymous field IN_Field_Value;
out integer OUT_Status;
in integer IN_Index;
in boolean IN_Allow_Add;
in string IN_Key_Name1;
in anonymous field IN_Key_Value1;
in string IN_Key_Name2;
in anonymous field IN_Key_Value2;
in string IN_Key_Name3;
in anonymous field IN_Key_Value3;
in string IN_Key_Name4;
in anonymous field IN_Key_Value4;
```

An example script is:

```
local integer l_status;
local string l_field;
local string l_key1, l_key2, l_key3, l_key4;
l_key1 = "Value1";
l_key2 = "Value2";
l_key3 = "Value3";
l_key4 = "Value4";
l_field = "Value"
call with name "MBS_Set_Table_Value4" in dictionary 5261,
Dictionary, "Table", "Field", l_field, l_status, 1 {Index}, true
{allow add},
                  "Key1", l_key1,
                  "Key2", l_key2,
                  "Key3", l_key3,
                  "Key4", l_key4;
if l_status <> OKAY then
    warning str(l_status);
end if;
```

MBS_Runtime_Execute

This call is used to execute Dexterity sanScript in the context of the specified dictionary.

The parameter list for this call is:

```
inout text INOUT_Text;
in integer IN_Prod_ID;
out integer OUT_Status;
```

An example script is:

```
local integer l_status;
local text l_text;
local integer l_dict;

clear l_text;
l_text = l_text + "warning ""Hello World"";" + char(13);
l_dict = 0; {Dictionary}
call with name "MBS_Runtime_Execute" in dictionary 5261, l_text,
l_dict, l_status;
if l_status <> OKAY then
    warning l_text;
end if;
```

MBS_SQL_Check_Exists

This call is used to execute a SQL Select statement in the context of the current company database and indicate whether any data records were returned.

The text field returned will contain the error message, or the number of records returned with or without data depending on the options passed in.

The parameter list for this call is:

```
inout text INOUT_TSQL;
in boolean IN_Return_Data;
in boolean IN_Return_Columns;
out integer OUT_Status;
```

An example script is:

```
local integer l_status;
local text l_text;
clear l_text;
l_text = l_text + "select * from table" + char(13);
call with name "MBS_SQL_Check_Exists" in dictionary 5261, l_text,
true, true, l_status;
case l_status
    in [OKAY]
        warning l_text;
    in [MISSING]
        warning l_text;
    else
        warning l_text;
end case;
```

MBS_Export_SQL_Query_To_File

This call is used to execute a SQL Select statement in the context of the current company database and export the result set as a text file.

The parameter list for this call is:

```
inout text INOUT_Code;
inout string INOUT_Pathname;
in boolean IN_Header;
in boolean IN_Quotes;
in integer IN_Mode; { 0 - CSV, 1 - Tab, 2 - User Defined }
in string IN_Delimiter;
in boolean IN_Append;
out long OUT_Rows;
out integer OUT_Status;
```

An example script is:

```
local integer l_status;
local text l_text;
local string l_path;
local long l_rows;

clear l_text;
l_text = l_text + "select * from table" + char(13);
call with name "MBS_Export_SQL_Query_To_File" in dictionary 5261,
l_text, l_path, true {Header}, true {Quotes}, 0 {CSV}, ""
{Delimiter}, false, {Append}, l_rows, l_status;
if l_status = OKAY then
     l_text = str(l_rows) + " rows exported to " + l_path + ".";
     warning l_text;
end if;
```

MBS_Script_Load_Dex

This call is used to load a Dexterity sanScript script from a Runtime Execute Script ID. It is designed to be used with the MBS_Runtime_Execute Helper Function.

The parameter list for this call is:

```
in string IN_ScriptID;
inout text INOUT_Text;
inout integer INOUT_Dict;
```

An example script is:

```
local text l_text;
local integer l_dict;

call with name "MBS_Script_Load_Dex" in dictionary 5261, "XXXX",
l_text, l_dict;
```

MBS_Script_Load_SQL

This call is used to load a SQL script from a SQL Execute Script ID. It is designed to be used with the MBS_SQL_Check_Exists Helper Function.

The parameter list for this call is:

```
in string IN_ScriptID;
inout text INOUT_Text;
```

An example script is:

```
local text l_text;

call with name "MBS_Script_Load_SQL" in dictionary 5261, "XXXX",
l_text;
```

MBS_Param_Set

This call is used to store a value in the DUOS SY_User_Object_Store (SY90000) table which can then be read by another script. It is designed to be used with the MBS_Runtime_Execute and MBS_Param_Get Helper Functions as a method of passing parameters.

The parameter list for this call is:

```
in string IN_Parameter;
in string IN_Value;
```

An example script is:

```
local string l_string;

l_string = "Value";
call with name "MBS_Param_Set" in dictionary 5261, "Variable",
l_string;
```

MBS_Param_Get

This call is used to read a previously set value from the DUOS SY_User_Object_Store (SY90000) table. It is designed to be used with the MBS_Runtime_Execute and MBS_Param_Set Helper Functions as a method of passing parameters.

The parameter list for this call is:

```
in string IN_Parameter;
out string OUT_Value;
```

An example script is:

```
local string l_string;

call with name "MBS_Param_Get" in dictionary 5261, "Variable",
l_string;
```

MBS_Param_Del

This call is used to remove a previously set value from the DUOS SY_User_Object_Store (SY90000) table. It is designed to be used with the MBS_Runtime_Execute and MBS_Param_Set Helper Functions as a method of passing parameters.

The parameter list for this call is:

```
in string IN_Parameter;
```

An example script is:

```
call with name "MBS_Param_Del" in dictionary 5261, "Variable";
```

MBS_Param_DelAll

This call is used to remove all previously stored parameter values for the current user from the DUOS SY_User_Object_Store (SY90000) table. It is designed to be used with the MBS_Runtime_Execute and MBS_Param_Set Helper Functions.

There is no parameter list for this call.

An example script is:

```
call with name "MBS_Param_DelAll" in dictionary 5261;
```

MBS_Auto_Log

This call is used to add a message into the Support Debugging Tool log file. It is designed to be used with the Automatic Debugger Mode to record additional information when a trigger fires.

The parameter list for this call is:

```
in string IN_Message;
```

An example script is:

```
call with name "MBS_Auto_Log" in dictionary 5261, "Message";
```

MBS_Trigger_Start

This call is used to activate an Automatic Debugger Mode Trigger and is designed to be used with Non-logging triggers.in the Automatic Debugger Mode.

The parameter list for this call is:

```
in string IN_TriggerID;
```

An example script is:

```
call with name "MBS_Trigger_Start" in dictionary 5261, "XXXX";
```

MBS_Trigger_Stop

This call is used to deactivate an Automatic Debugger Mode Trigger and is designed to be used with Non-logging triggers.in the Automatic Debugger Mode.

The parameter list for this call is:

```
in string IN_TriggerID;
```

An example script is:

```
call with name "MBS_Trigger_Stop" in dictionary 5261, "XXXX";
```

MBS_Logging_Start

This call is used to programmatically start Manual Logging Mode and is designed to be used with Non-logging triggers.in the Automatic Debugger Mode.

There are no parameters for this call.

An example script is:

```
call with name "MBS_Logging_Start" in dictionary 5261;
```

MBS_Logging_Stop

This call is used to programmatically stop Manual Logging Mode and is designed to be used with Non-logging triggers.in the Automatic Debugger Mode.

There are no parameters for this call.

An example script is:

```
call with name "MBS_Logging_Stop" in dictionary 5261;
```

MBS_Email_API

This call is can be used to call the Email Engine in the Support Debugging Tool from external Dexterity applications.

The parameter list for this call is:

```
in string IN_EmailFrom;
in string IN_EmailTo;
in string IN_EmailCC;
in string IN_EmailBCC;
in string IN_EmailSubject;
in text IN_EmailBody;
in text IN_EmailSignature;
in boolean IN_EmailSignatureDefault;
in text IN_EmailAttachments;
in boolean IN_EmailPreview;
in boolean IN_EmailAutoSend;
```

An example script is:

```
local string l_EmailFrom;
local string l_EmailTo;
local string l_EmailCC;
local string l_EmailBCC;
local string l_EmailSubject;
local text l_EmailBody;
local text l_EmailSignature;
local boolean l_EmailSignatureDefault;
local text l_EmailAttachments;
local boolean l_EmailPreview;
local boolean l_EmailAutoSend;

l_EmailTo = "email@domain.com";
l_EmailSubject = "Email API Test";
l_EmailBody = "This is a test of the Email API"+char(13);
l_EmailSignatureDefault = true;
l_EmailAttachments          =          l_EmailAttachments          +
"C:\Dex1000\Data\Dex.ini"+char(13);
l_EmailAttachments          =          l_EmailAttachments          +
"C:\Dex1100\Data\Dex.ini"+char(13);
l_EmailPreview = false;
l_EmailAutoSend = false;

call with name "MBS_Email_API" in dictionary 5261, l_EmailFrom,
l_EmailTo, l_EmailCC, l_EmailBCC, l_EmailSubject, l_EmailBody,
l_EmailSignature, l_EmailSignatureDefault, l_EmailAttachments,
l_EmailPreview, l_EmailAutoSend;
```

Chapter 7: RW Functions

The Support Debugging Tool has support for six generic Report Writer functions in the core Dynamics.dic dictionary which can be used in the Report Writer with any report as a user defined function in a calculated field.

To use the functions with the Support Debugging Tool, the first two parameters passed in for each of the functions will be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of a Runtime Execute script.

The Dexterity sanScript code contained in the Script ID will then be executed allowing for the development of custom Report Writer functions. The Helper Function Assistant window can automatically add the template code to handle the parameter passing into and out of the script using the MBS_Param_Get and MBS_Param_Set helper functions.

Below are the details of the RW Functions available from the system series:

- *rw_ReportStart*
- *rw_ReportEnd*
- *rw_TableHeaderString*
- *rw_TableHeaderCurrency*
- *rw_TableLineString*
- *rw_TableLineCurrency*

You can use these functions to capture information off a report and store it in the log using the MBS_Auto_Log helper function. For example: you can capture the values of legends, calculated fields, report fields or values from any of the attached tables.

While the Report Writer Functions were designed to work with the report start and end events and with a header and line type document such as seen in Sales Order Processing, you can use the functions and parameters as desired to achieve the results required.

More detail on these functions is available from Knowledge Base (KB) article 888884:

http://support.microsoft.com/kb/888884

rw_ReportStart

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `string` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
```

An example of how it would be called from the Report Writer for a calculated field with a Result Type of string is:

```
FUNCTION_SCRIPT(  rw_ReportStart  5261  "Script ID"  )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local string MBS_Status;

MBS_Status = "";

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261, "ReportStart",
MBS_Status;
```

rw_ReportEnd

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `string` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
```

An example of how it would be called from the Report Writer for a calculated field with a Result Type of string is:

```
FUNCTION_SCRIPT(  rw_ReportEnd  5261  "Script ID"  )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local string MBS_Status;

MBS_Status = "";

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261, "ReportEnd",
MBS_Status;
```

rw_TableHeaderString

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `string` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
in string sNumber; {control field 1}
in integer sType; {control field 2}
in integer iControl; {which piece of data to return}
```

An example of how it would be called from the Report Writer for a calculated field with a Result Type of string is:

```
FUNCTION_SCRIPT(  rw_TableHeaderString  5261  "Script ID"
SOP_HDR_WORK.SOP Number  SOP_HDR_WORK.SOP Type  1  )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local string MBS_TableHeaderString;
local string MBS_Number;
local integer MBS_Type;
local integer MBS_Control;
local string MBS_String;

call with name "MBS_Param_Get" in dictionary 5261, "Number",
MBS_Number;
call with name "MBS_Param_Get" in dictionary 5261, "Type",
MBS_String;
MBS_Type = integer(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "Control",
MBS_String;
MBS_Control = integer(value(MBS_String));
MBS_TableHeaderString = "";

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261,
"TableHeaderString", MBS_TableHeaderString;
```

rw_TableHeaderCurrency

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `currency` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
in string sNumber; {control field 1}
in integer sType; {control field 2}
in integer iControl; {which piece of data to return}
```

An example of how it would be called from the Report Writer for a calculated field with a Result Type of currency is:

```
FUNCTION_SCRIPT(  rw_TableHeaderCurrency  5261  "Script ID"
SOP_HDR_WORK.SOP Number  SOP_HDR_WORK.SOP Type  1  )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local currency MBS_TableHeaderCurrency;
local string MBS_Number;
local integer MBS_Type;
local integer MBS_Control;
local string MBS_String;

call with name "MBS_Param_Get" in dictionary 5261, "Number",
MBS_Number;
call with name "MBS_Param_Get" in dictionary 5261, "Type",
MBS_String;
MBS_Type = integer(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "Control",
MBS_String;
MBS_Control = integer(value(MBS_String));
MBS_TableHeaderCurrency = 0.0000;

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261,
"TableHeaderCurrency", str(MBS_TableHeaderCurrency);
```

rw_TableLineString

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `string` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
in string sNumber; {control field 1}
in integer sType; {control field 2}
in currency cSequenceOne; {control field 3}
in currency cSequenceTwo; {control field 4}
in integer iControl; {which piece of data to return}
```

To use the rw_TableLineString report writer function we need to be able to pass the two sequence fields as currency data type. So to use the Sales Order Processing fields SOP_LINE_WORK.Line Item Sequence and SOP_LINE_WORK.Component Sequence, we will need to create two calculated fields to convert them from a long integer to a currency data type.

Calculated Field **(C) Line Item Sequence** is defined as result type currency with the expression of *SOP_LINE_WORK.Line Item Sequence * 1.00000*.

Calculated Field **(C) Component Sequence** is defined as result type currency with the expression of *SOP_LINE_WORK.Component Sequence * 1.00000*.

An example of how it would be called from the Report Writer for a calculated field with a Result Type of string is:

```
FUNCTION_SCRIPT( rw_TableLineString 5261 "Script ID"
SOP_LINE_WORK.SOP Number SOP_LINE_WORK.SOP Type (C) Line Item
Sequence (C) Component Sequence 1 )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local string MBS_TableLineString;
local string MBS_Number;
local integer MBS_Type;
local currency MBS_SequenceOne;
local currency MBS_SequenceTwo;
local integer MBS_Control;
local string MBS_String;

call with name "MBS_Param_Get" in dictionary 5261, "Number",
MBS_Number;
call with name "MBS_Param_Get" in dictionary 5261, "Type",
MBS_String;
MBS_Type = integer(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "SequenceOne",
MBS_String;
MBS_SequenceOne = currency(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "SequenceTwo",
MBS_String;
MBS_SequenceTwo = currency(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "Control",
MBS_String;
MBS_Control = integer(value(MBS_String));
MBS_TableLineString = "";

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261,
"TableLineString", MBS_TableLineString;
```

rw_TableLineCurrency

This report writer function can be used in the Report Writer as a user defined function in a calculated field. The first two parameters passed in need to be the Dictionary ID for the Support Debugging Tool (5261) and the Script ID of the Runtime Execute script to be executed.

The returned value for this report writer function is a `currency` and the input parameter list for this report writer function is:

```
in integer dict_id; {Dictionary ID}
in string script_id; {Script ID}
in string sNumber; {control field 1}
in integer sType; {control field 2}
in currency cSequenceOne; {control field 3}
in currency cSequenceTwo; {control field 4}
in integer iControl; {which piece of data to return}
```

To use the rw_TableLineCurrency report writer function we need to be able to pass the two sequence fields as currency data type. So to use the Sales Order Processing fields SOP_LINE_WORK.Line Item Sequence and SOP_LINE_WORK.Component Sequence, we will need to create two calculated fields to convert them from a long integer to a currency data type.

Calculated Field **(C) Line Item Sequence** is defined as result type currency with the expression of *SOP_LINE_WORK.Line Item Sequence * 1.00000*.

Calculated Field **(C) Component Sequence** is defined as result type currency with the expression of *SOP_LINE_WORK.Component Sequence * 1.00000*.

An example of how it would be called from the Report Writer for a calculated field with a Result Type of currency is:

```
FUNCTION_SCRIPT( rw_TableLineCurrency 5261 "Script ID"
SOP_LINE_WORK.SOP Number SOP_LINE_WORK.SOP Type (C) Line Item
Sequence (C) Component Sequence 1 )
```

The template Runtime Execute script added by the Helper Function Assistant window is:

```
local currency MBS_TableLineCurrency;
local string MBS_Number;
local integer MBS_Type;
local currency MBS_SequenceOne;
local currency MBS_SequenceTwo;
local integer MBS_Control;
local string MBS_String;

call with name "MBS_Param_Get" in dictionary 5261, "Number",
MBS_Number;
call with name "MBS_Param_Get" in dictionary 5261, "Type",
MBS_String;
MBS_Type = integer(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "SequenceOne",
MBS_String;
MBS_SequenceOne = currency(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "SequenceTwo",
MBS_String;
MBS_SequenceTwo = currency(value(MBS_String));
call with name "MBS_Param_Get" in dictionary 5261, "Control",
MBS_String;
MBS_Control = integer(value(MBS_String));
MBS_TableLineCurrency = 0.00000;

{ Add your code below here }

{ Add your code above here }

call with name "MBS_Param_Set" in dictionary 5261,
"TableLineCurrency", str(MBS_TableLineCurrency);
```

Support Debugging Tool Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

** End of document - Debugger.doc - DM - 18 September 2014 **