Microsoft Dynamics™

Support Debugging Tool for
Microsoft Dynamics™ GP

User's Guide

Build 12

Copyright

Copyright © 2010 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation. Notwithstanding the foregoing, the licensee of the software with which this document was provided may make a reasonable number of copies of this document solely for internal use.

Trademarks

Microsoft, Excel, FRx, MapPoint, Microsoft Dynamics, Outlook, SharePoint SQL Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation or FRx Software Corporation or their affiliates in the United States and/or other countries. FRx Software Corporation is a subsidiary of Microsoft Corporation. FairCom and c-tree Plus are trademarks of FairCom Corporation and are registered in the United States and other countries.

The names of actual companies and products mentioned herein may be trademarks or registered marks - in the United States and/or other countries - of their respective owners.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Intellectual property

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Warranty disclaimer

Microsoft Corporation disclaims any warranty regarding the sample code contained in this documentation, including the warranties of merchantability and fitness for a particular purpose.

Limitation of Liability

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Microsoft Corporation. Microsoft Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this manual. Neither Microsoft Corporation nor anyone else who has been involved in the creation, production or delivery of this documentation shall be liable for any indirect, incidental, special, exemplary or consequential damages, including but not limited to any loss of anticipated profit or benefits, resulting from the use of this documentation or sample code.

License Agreement

Use of this product is covered by a license agreement provided with the software product.

Application & documentation designed and developed by

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

Contents

Chapter 1: Introduction **1**

Examples of use 2

Support 3

Chapter 2: Standard Mode Features **4**

Installation, Navigation and Security 5

Manual Logging Mode 9

Individual Logging Control 12

About Support Debugging Tool 13

Dex.ini Settings 14

Resource Information 22

Security Profiler 28

Security Information 31

Configuration Export/Import 35

ScreenShot 37

Send Email 41

Chapter 3: Advanced Mode Features **43**

Advanced Mode Access 44

Automatic Debugger Mode 45

Dictionary Control 62

XML Table Export 65

XML Table Import 68

Runtime Execute 70

SQL Execute 74

Configuration Maintenance 80

Administrator Settings 81

Dex.ini Configuration 88

Chapter 4: Dex.ini Settings **91**

Support Debugging Tool Settings 91

System Settings 94

Script Editor Settings 96

Chapter 5: Helper Functions **97**

MBS_Get_Window_Value 98

MBS_Set_Window_Value 99

MBS_Get_Table_Value1 100

MBS_Set_Table_Value1 101

MBS_Get_Table_Value2 102

MBS_Set_Table_Value2 103

MBS_Get_Table_Value3 104

MBS_Set_Table_Value3 105

MBS_Get_Table_Value4 106

MBS_Set_Table_Value4 107

MBS_Runtime_Execute 108
MBS_SQL_Check_Exists 109
MBS_Script_Load_Dex 110
MBS_Script_Load_SQL 111
MBS_Param_Set 112
MBS_Param_Get 113
MBS_Param_Del 114
MBS_Param_DelAll 115
MBS_Auto_Log 116

Chapter 6: RW Functions 117

rw_ReportStart 118
rw_ReportEnd 119
rw_TableHeaderString 120
rw_TableHeaderCurrency 121
rw_TableLineString 122
rw_TableLineCurrency 124

Support Debugging Tool Index 126

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.
- 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 troubleshooting tool developed and released by the Microsoft Dynamics GP product support team. If you require assistance to use the tool, or have feedback or suggestions for improvements or new features, please post your question or comments on the public Dynamics GP Developer newsgroup (microsoft.public.dynamics.gp.developer). You can use the link below to access a web-based interface to the newsgroup.

http://www.microsoft.com/Businesssolutions/Community/Newsgroups/dgbrowser/en-us/default.mspx?dg=microsoft.public.dynamics.gp.developer

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

Note: *Technical support for this tool is limited to break/fix and how to questions on the tool's specific functionality. If you need assistance in understanding the generated log files, or how to use the tool to resolve a specific issues, this should be handled as part of a standard technical support incident related to the specific issue you are attempting to troubleshoot or resolve.*

Chapter 2: Standard Mode Features

This chapter includes the following sections:

- *Installation, Navigation and Security*
- *Manual Logging Mode*
- *Individual Logging Control*
- *About Support Debugging Tool*
- *Dex.ini Settings*
- *Resource Information*
- *Security Profiler*
- *Security Information*
- *Configuration Export/Import*
- *ScreenShot*
- *Send Email*

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 or Windows Server 2008 system and User Access Control (UAC) is active, please launch Microsoft Dynamics GP with the Run as Administrator option to complete the installation.

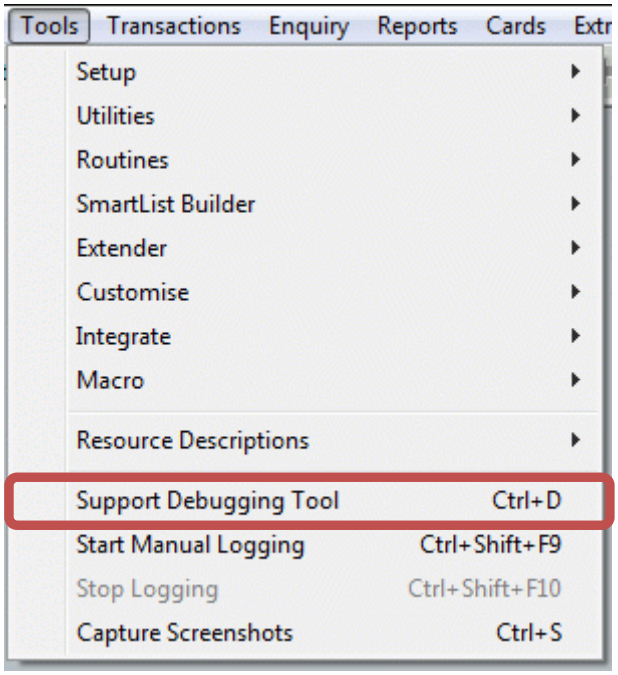
The Support Debugging Tool does not use any SQL tables to store its data, instead it uses an XML file called Debugger.xml. By default this file is stored in the application f older (v8.00 and v9.00) or in the data subfolder beneath the application folder (v10.00 or later) and will be created if required 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.

The Support Debugging Tool requires the version 8.00 Dexterity Runtime update 8.00m88 or version 9.00 Service Pack 3 or later (see links below). The Support Debugging Tool works with all builds of version 10.00 and later versions.

Dexterity Runtime Update Page for Great Plains 8.0
https://mbs.microsoft.com/customersource/support/downloads/servicepacks/dexruntime800.htm?printpage=false

Service Packs for Microsoft Dynamics GP v9.0
https://mbs.microsoft.com/customersource/support/downloads/servicepacks/microsoftdynamicsgp90servicepacks.htm?printpage=false

For versions 8.00 and 9.00: Once logged into Microsoft Dynamics GP, the Support Debugging Tool can be found under the Tools menu (highlighted below). It also has the keyboard shortcut Ctrl+D assigned to it.



As security for these versions is granted by default, no additional action is required to use the tool. Standard Security (Tools >> Setup >> System >> Security) or Advanced Security (Tools >> Setup >> System >> Advanced Security) can be used to deny access to forms of the tool if desired. 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.

For version 10.00 and later: 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.

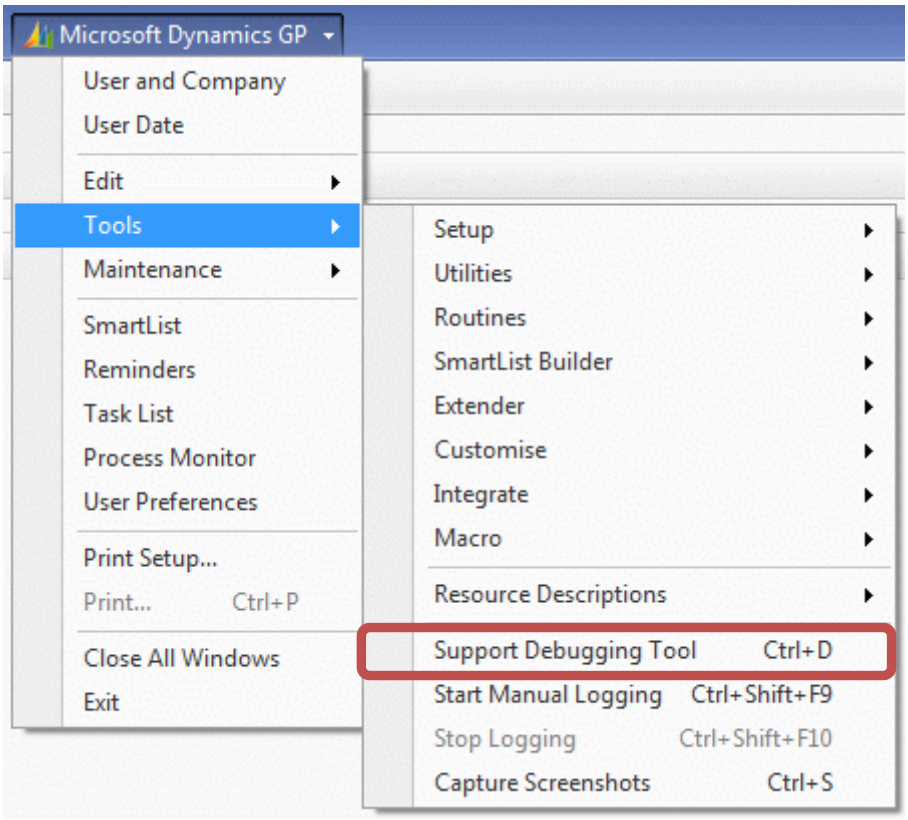
To grant security to the forms of the Support Debugging Tool use the User Security Setup window (Microsoft Dynamics GP >> Tools >> Setup >> System >> User Security). After selecting the user and company, select one of the two security roles below:

MBS DEBUGGER ADMIN (Debugging Tool Administrator)

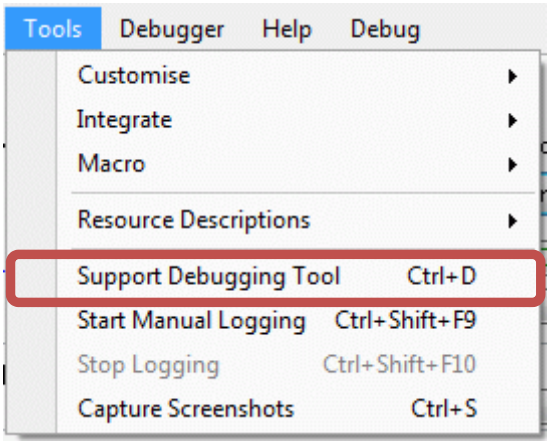
MBS DEBUGGER USER (Debugging Tool User)

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.

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.



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 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.

Manual Logging Mode

The Microsoft Dynamics GP core application runs on the Dexterity runtime engine. Dexterity provides the following logging facilities:

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.

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.

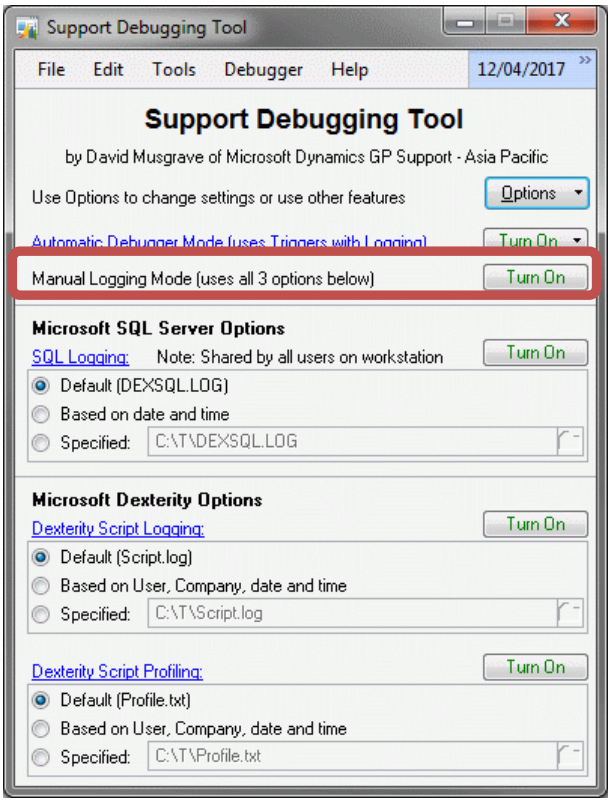
Using any logging facility will create additional processing overhead for the application. Logging should only be used when actually looking to resolve an issue with the system.

Manual Logging Mode

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.

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 Microsoft Dynamics GP application folder (v8.00 and v9.00), or the data subfolder beneath the Microsoft Dynamics GP application folder (v10.00 or later). 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.

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

  These files will contain the Dexterity Script Logging results.

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

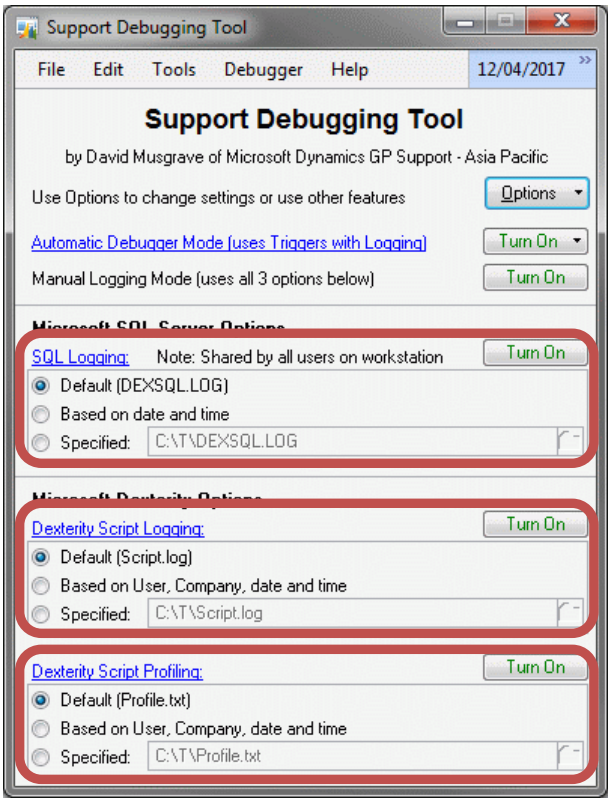
  These files will contain the Dexterity Script Profiling results.

<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.

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.

Individual Logging Control

Individual Logging Control allows each of the logging options to be controlled independently. 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.



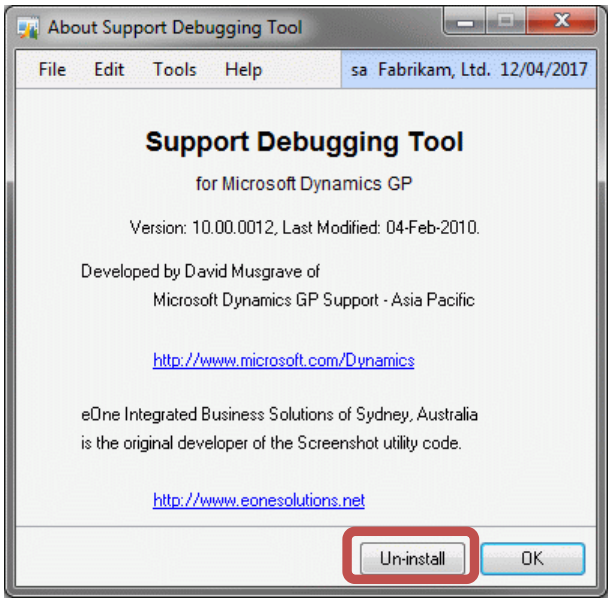
Turning all logging methods on using the Based on date and time and Based on User, Company, date and time is the same as using Manual Logging Mode.

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.

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. 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.

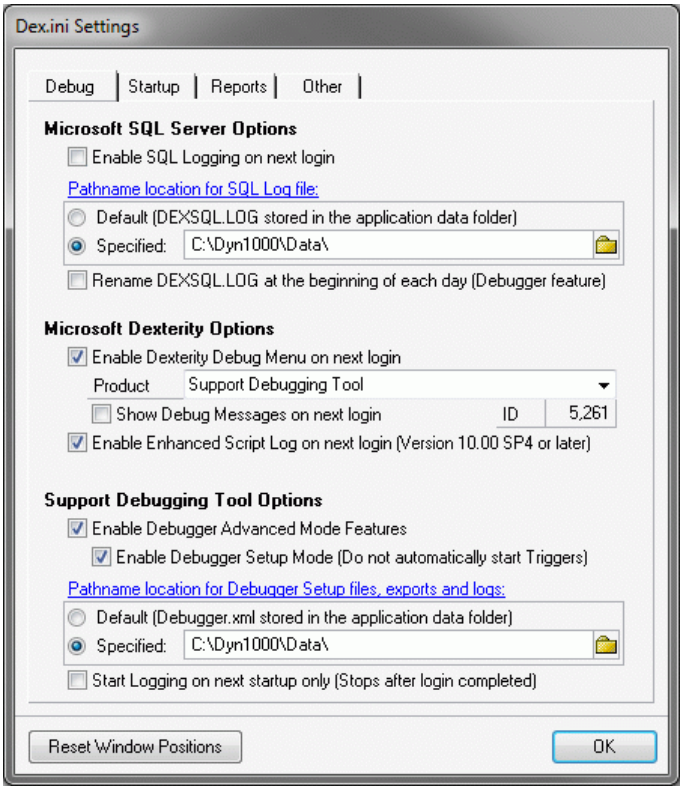
Dex.ini Settings

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.

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.

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. It will also open the main Support Debugging Tool window after login. 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.

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.

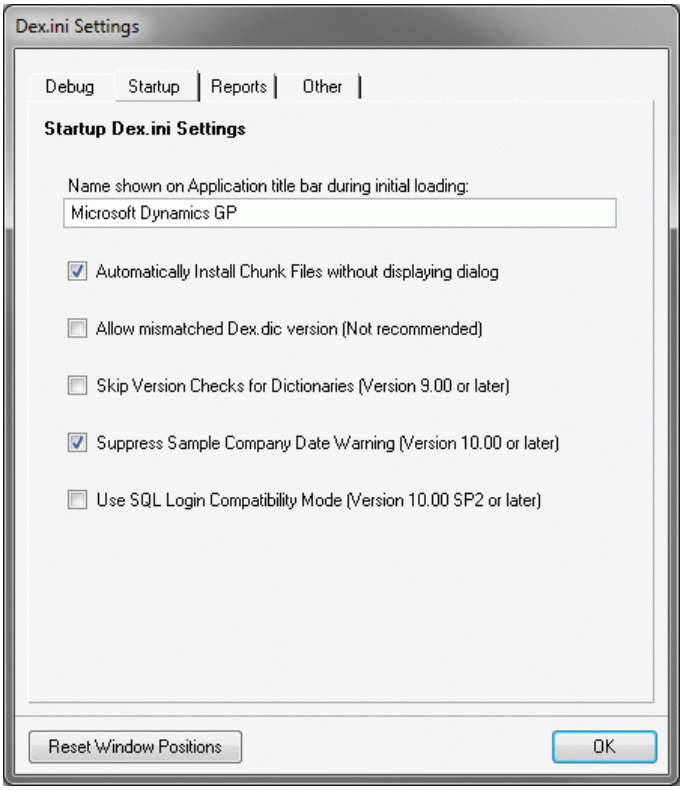
Start Logging on next startup only

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.

Allow mismatched Dex.dic version

This option will update the AllowWrongDex Dex.ini setting to allow Microsoft Dynamics GP to launch with a mismatched Dex.dic version. Use of this option is not recommended.

Skip Version Checks for Dictionaries

This option will update the SkipVersionChecks Dex.ini setting to allow Microsoft Dynamics GP to launch even when the loaded dictionary versions do not match the version information in the database. This option should not be used in a production environment. This option is valid from v9.00 onwards.

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. This option requires v10.00 or later.

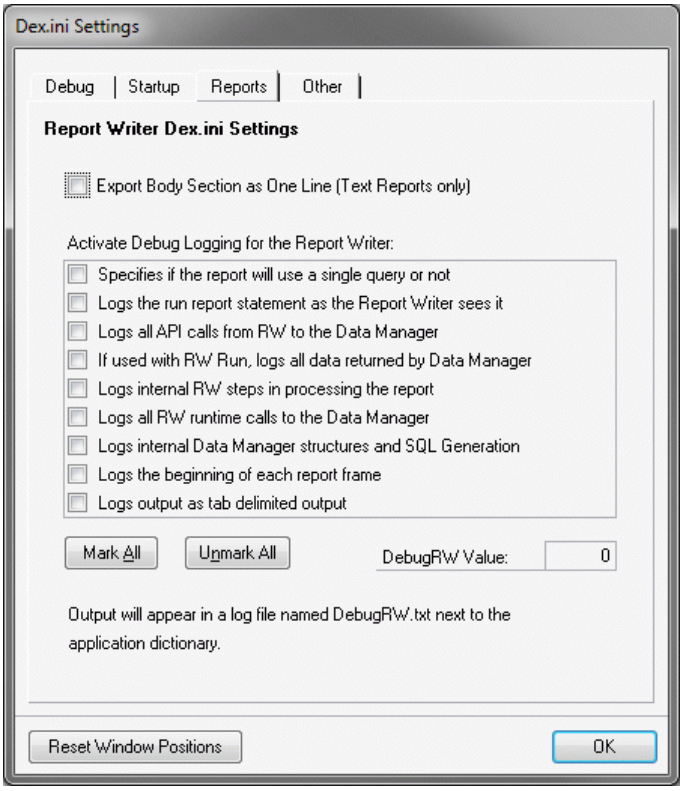
Use SQL Login Compatibility Mode

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).

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.

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

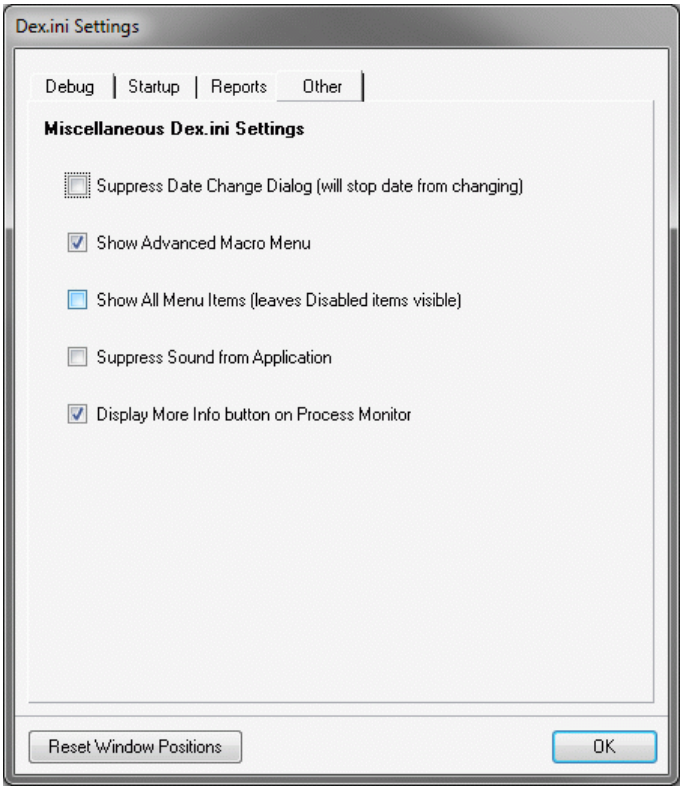
Use this button to activate all the Report Writer debug logging.

Unmark All

Use this button to turn off Report Writer debug logging.

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.

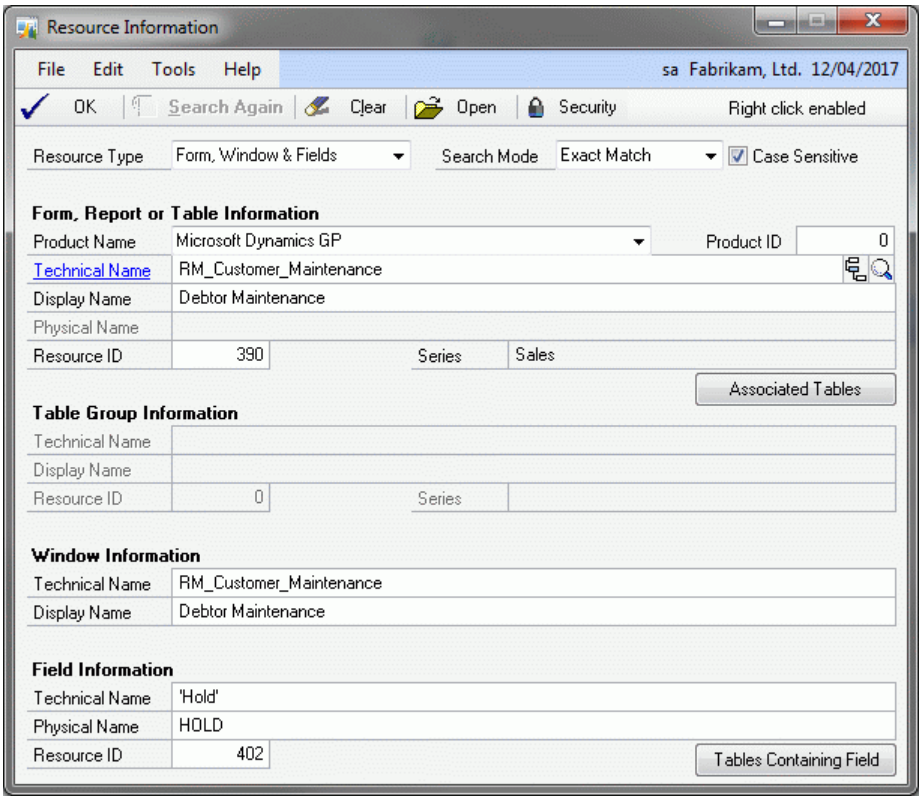
Display More Info button on Process Monitor

This option will update the QueueMoreInfo Dex.ini setting to display the More Info button on the Process Monitor window (File >> Process Monitor for v8.00 and v9.00 or Microsoft Dynamics GP >> Process Monitor for v10.00 or later).

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 or report resource in the any dictionary currently installed in the Microsoft Dynamics GP application.



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.

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; or Reports.

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.

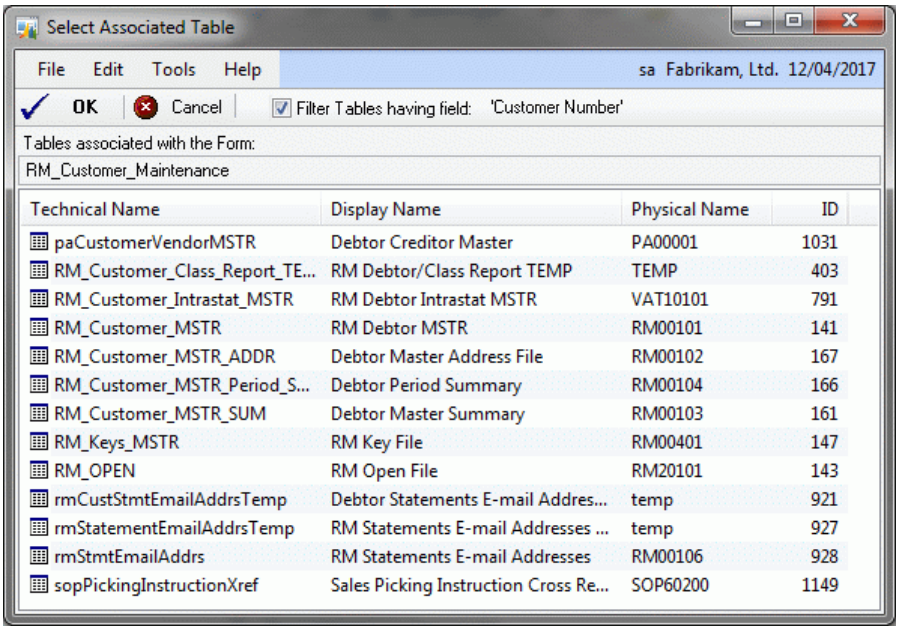
Case Sensitive

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.

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.

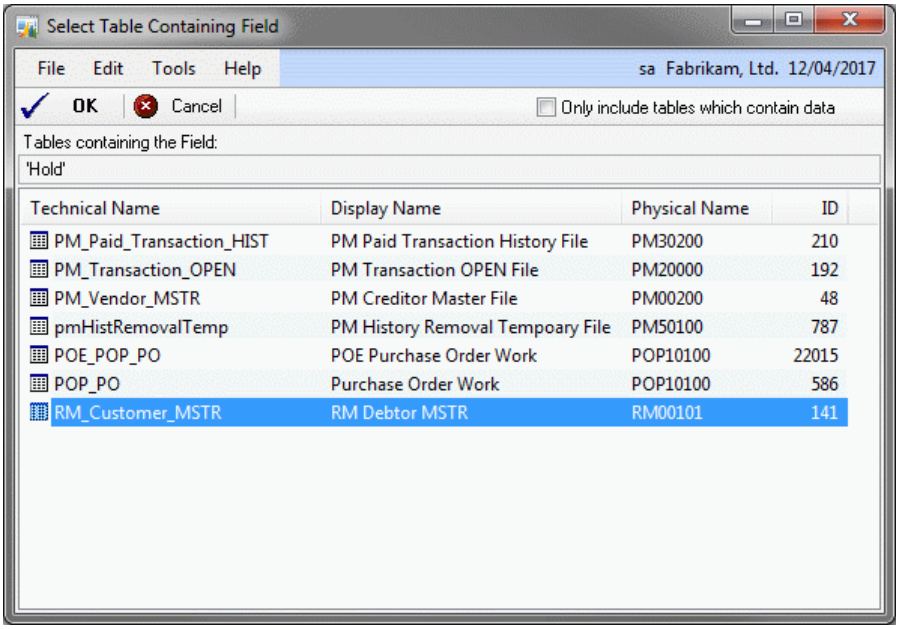
If a field is selected on the Resource Information window, you will have the option to filter the list of tables to only tables having the specified field. If the field is not available in any tables, this option will be disabled.



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.

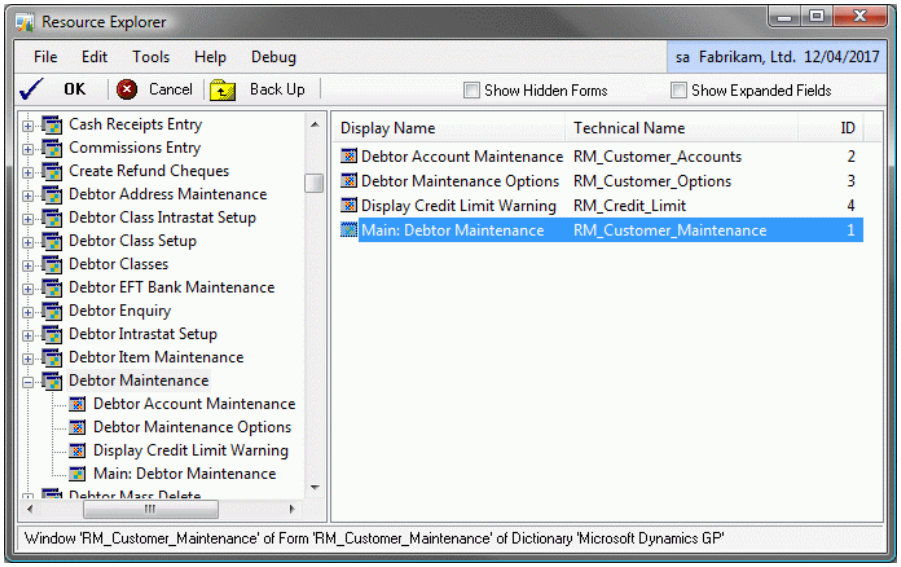
Tables Containing Field Button

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.



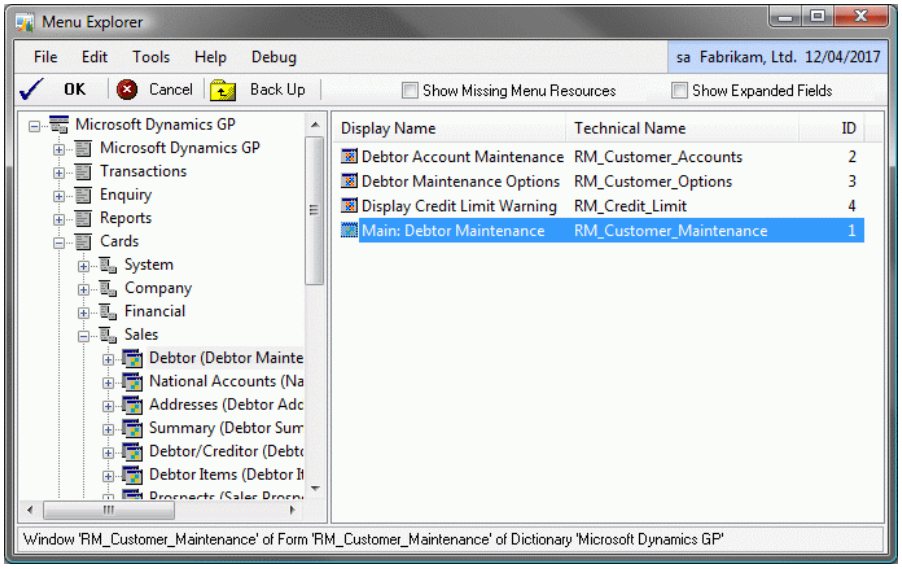
Selecting a table from this window will change the Resource Information window into Table & Field mode and display the details of the selected table and field.

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.



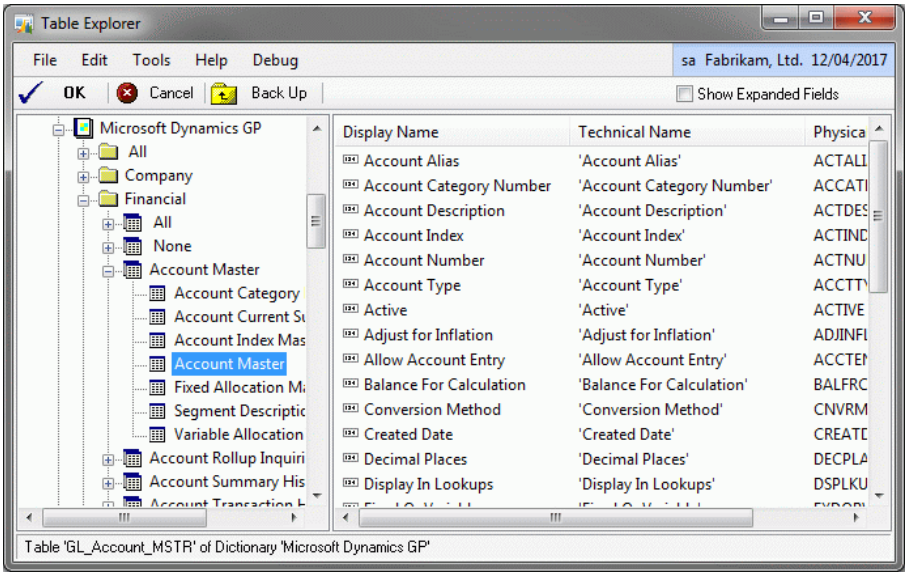
To insert a form name or window name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

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.



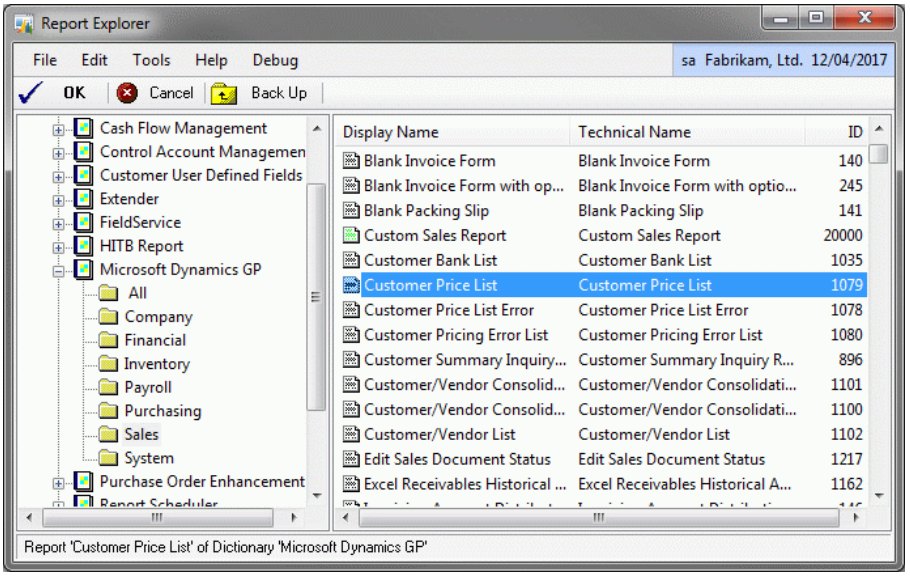
To insert a form name or window name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

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.



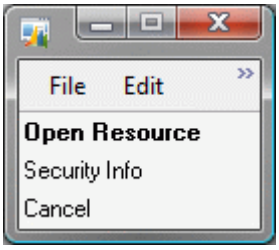
To insert a table name, select the resource in the right hand pane and click OK. If no resources are selected on the right hand pane, the currently selected resource in the left hand pane will be used when OK is clicked.

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

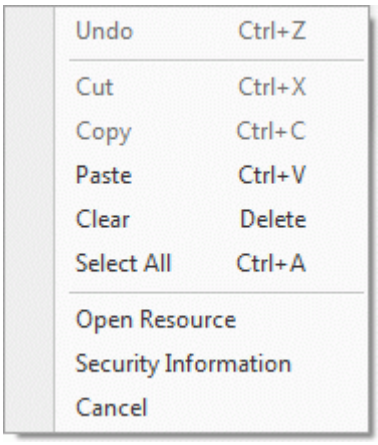


To insert a report name, select the resource in the right hand pane and click OK. Custom Reports are shown with a different icon in the right hand pane.

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.



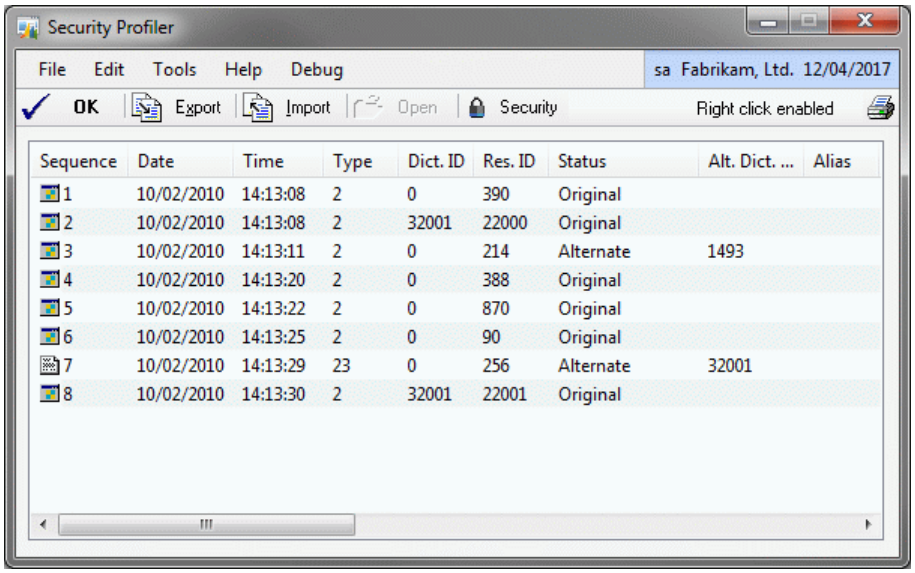
or



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.



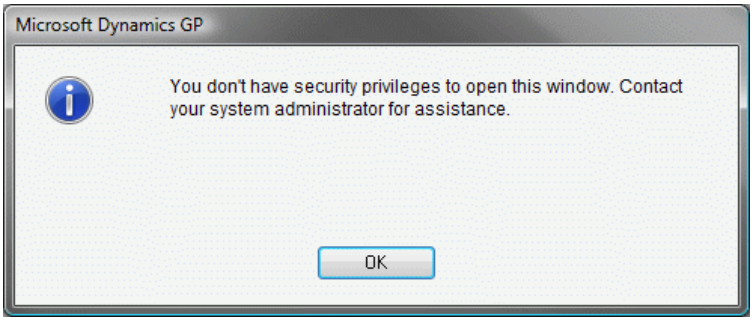
Sequence	Date	Time	Type	Dict. ID	Res. ID	Status	Alt. Dict. ...	Alias
1	10/02/2010	14:13:08	2	0	390	Original		
2	10/02/2010	14:13:08	2	32001	22000	Original		
3	10/02/2010	14:13:11	2	0	214	Alternate	1493	
4	10/02/2010	14:13:20	2	0	388	Original		
5	10/02/2010	14:13:22	2	0	870	Original		
6	10/02/2010	14:13:25	2	0	90	Original		
7	10/02/2010	14:13:29	23	0	256	Alternate	32001	
8	10/02/2010	14:13:30	2	32001	22001	Original		

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 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.



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.

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.

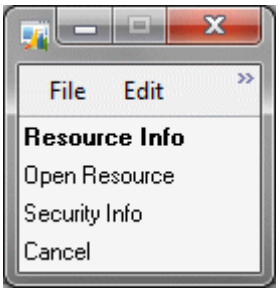
Security Button

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.

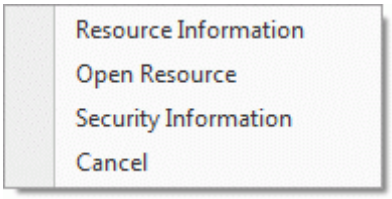
Print Button

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.



or



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.

Security Information

You can 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.

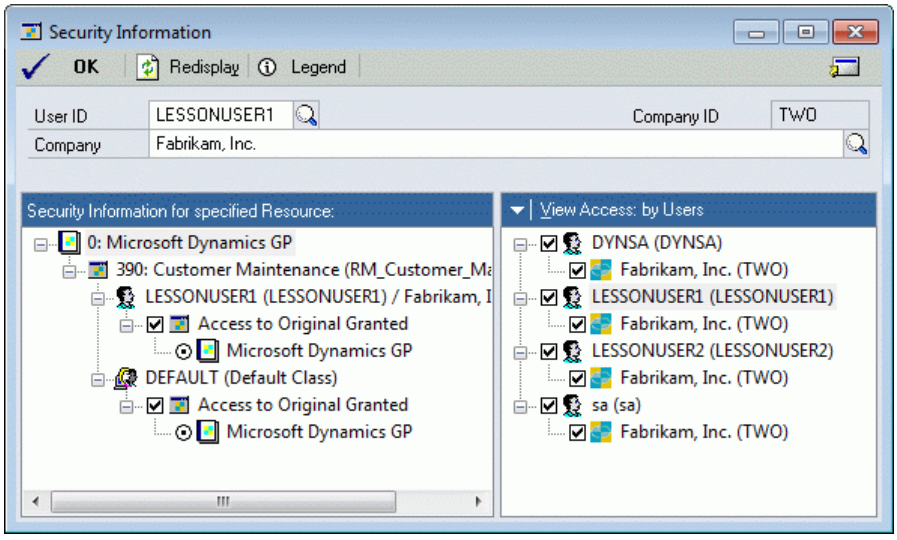
For a user to have access to the Security Information window they must have access to the security windows under Tools >> Setup >> System.

For versions 8.00 and 9.00: The user must have access to Standard Security and User Class Setup or to Advanced Security.

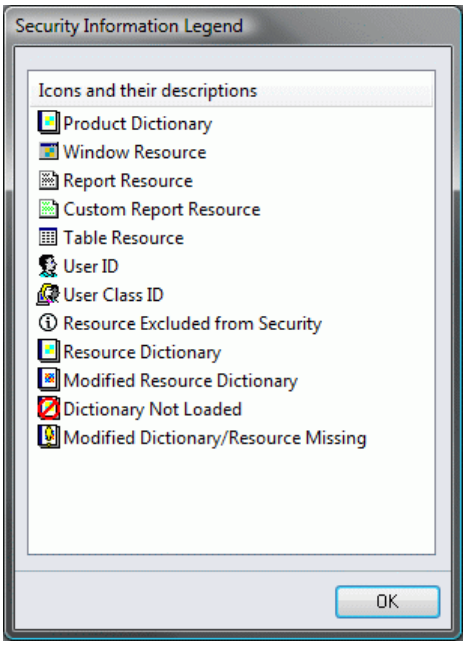
For version 10.00 and later: The user must have access to User Security Setup, Security Task Setup, Security Role Setup and Alternate/Modified Forms and Reports.

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.

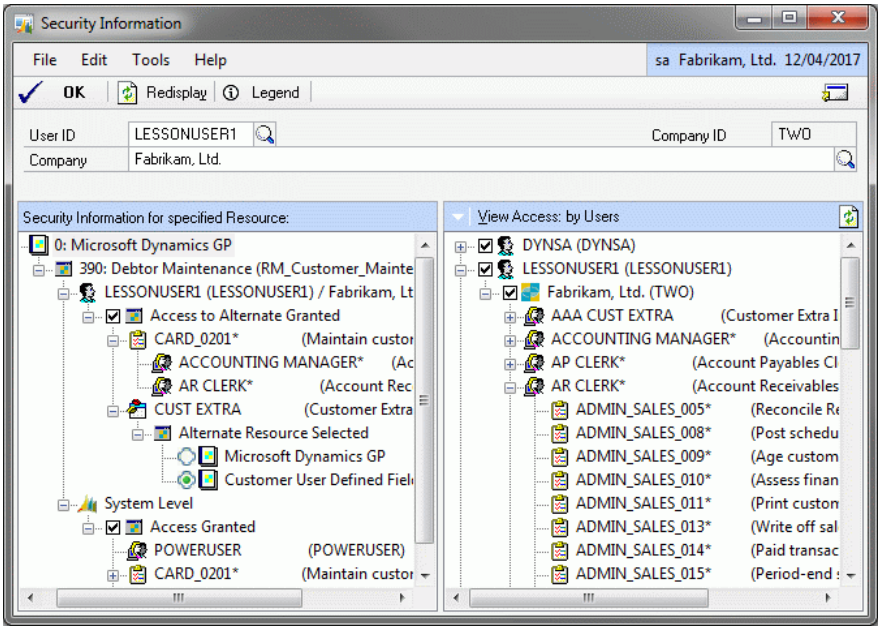
Below is an example of the Security Information window for versions 8.00 and 9.00. It shows the security settings for the user and if the user belongs to a user class it also shows the settings for the class.



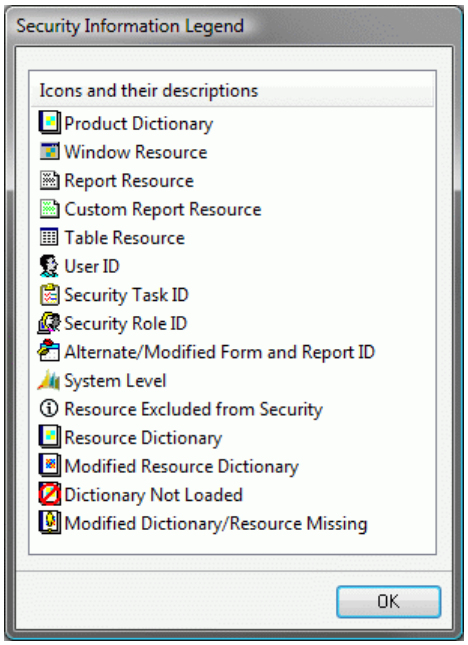
If you click the Legend Button the following window will be displayed to show the meanings of the different icons used.



Below is an example of the Security Information window for version 10.00 and later. 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.



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.

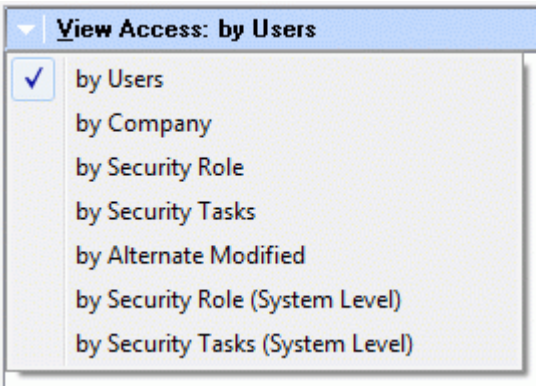
Legend Button
This button will open the Security Information Legend window.

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

***For version 10.00 and later**: 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.*

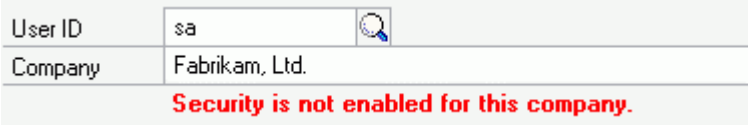
The right hand pane on the Security Information window displays a number of different views into the company access and security information. Use the View Access button drop down list to change view. When changing views the currently selected object will remain selected if possible. This pane can be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

Below are the views available for v10.00 or later.



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

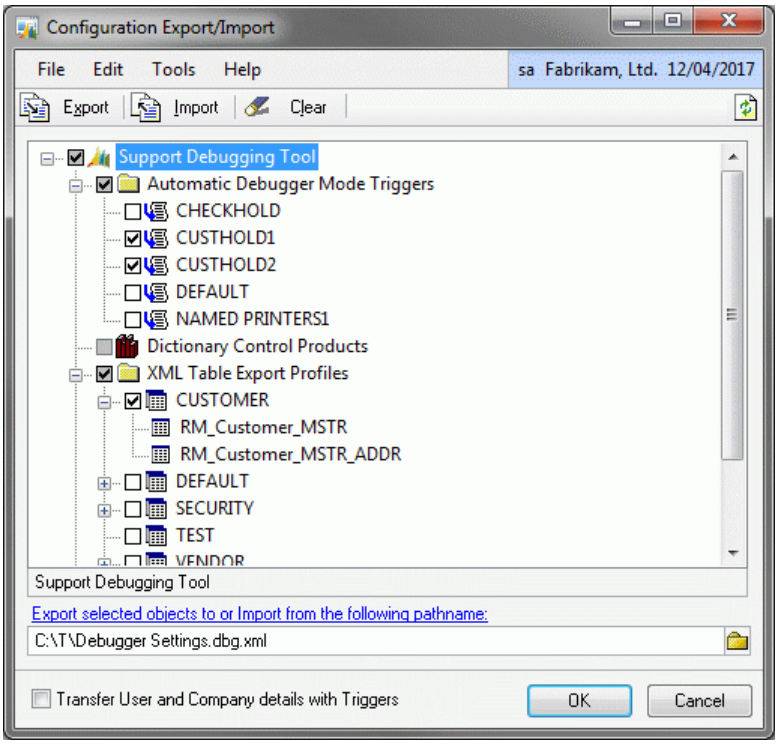
The Security Information window will highlight when security is not activated for the selected company. This can be enabled from Company Setup window (Microsoft Dynamics GP >> Tools >> Setup >> Company >> Company).



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.



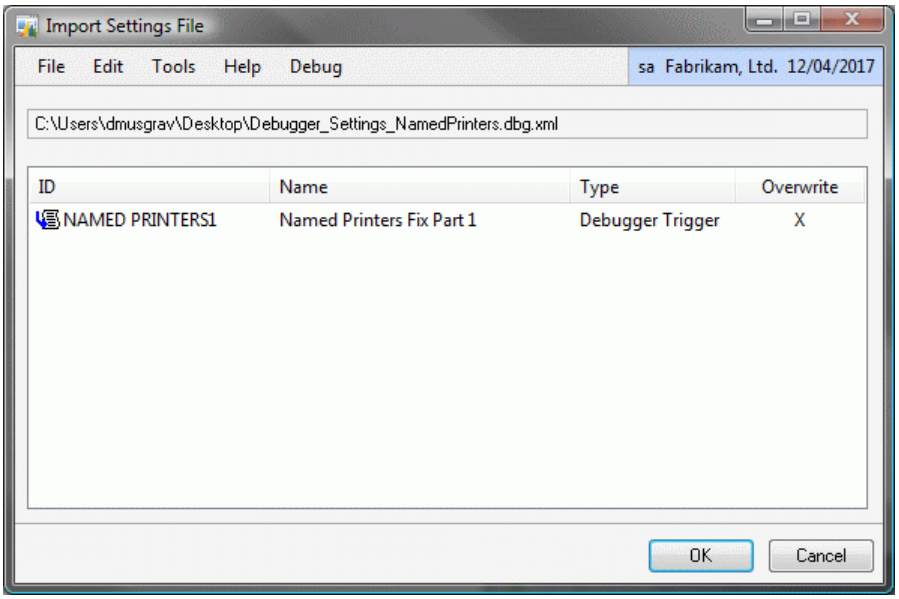
Below is a description of the individual fields on the window:

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 showing objects that will be overwritten before importing.



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

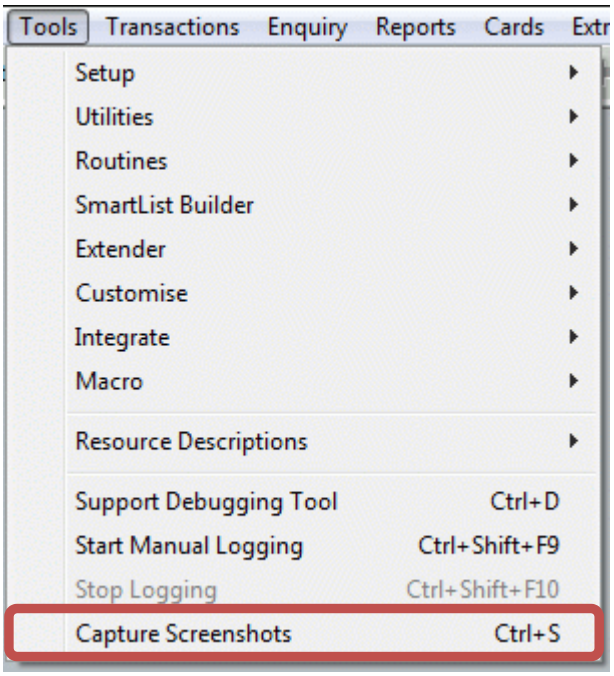
ScreenShot is a tool which can capture screenshots of all the open windows in the application as well as a System Status report and then either email or save the files. ScreenShot is based on code originally developed by eOne Integrated Business Solutions.

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.

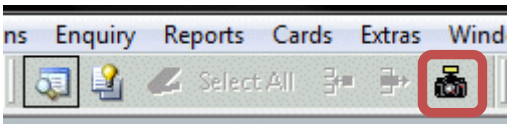
The System Status report contains information about the system including registration information, current login information, environmental information (such as operating system, database and ODBC versions), product information (including all version and build numbers) and a list of the attached screenshots.

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

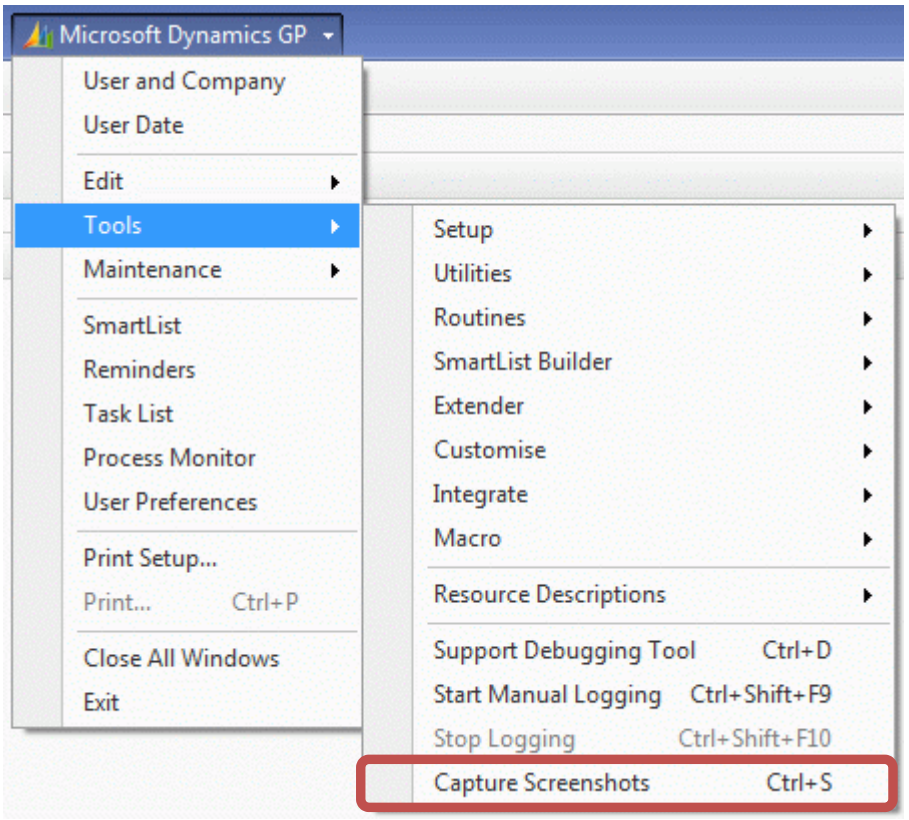
For versions 8.00 and 9.00: You can open it directly from the Tools menu (highlighted below). It also has the keyboard shortcut Ctrl+S assigned to it.



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

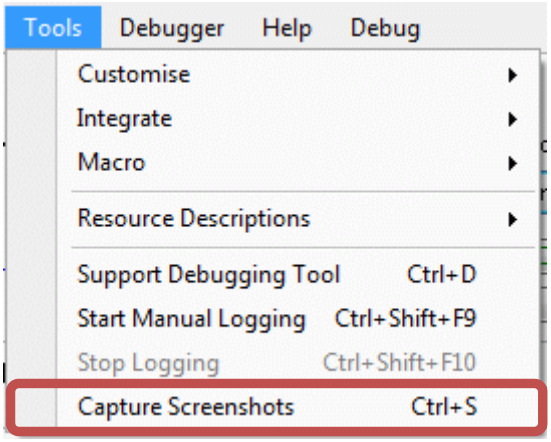


For versions 10.00 and later: You can open it directly from the Tools menu underneath the Microsoft Dynamics GP menu (highlighted below). It also has the keyboard shortcut Ctrl+S assigned to it.



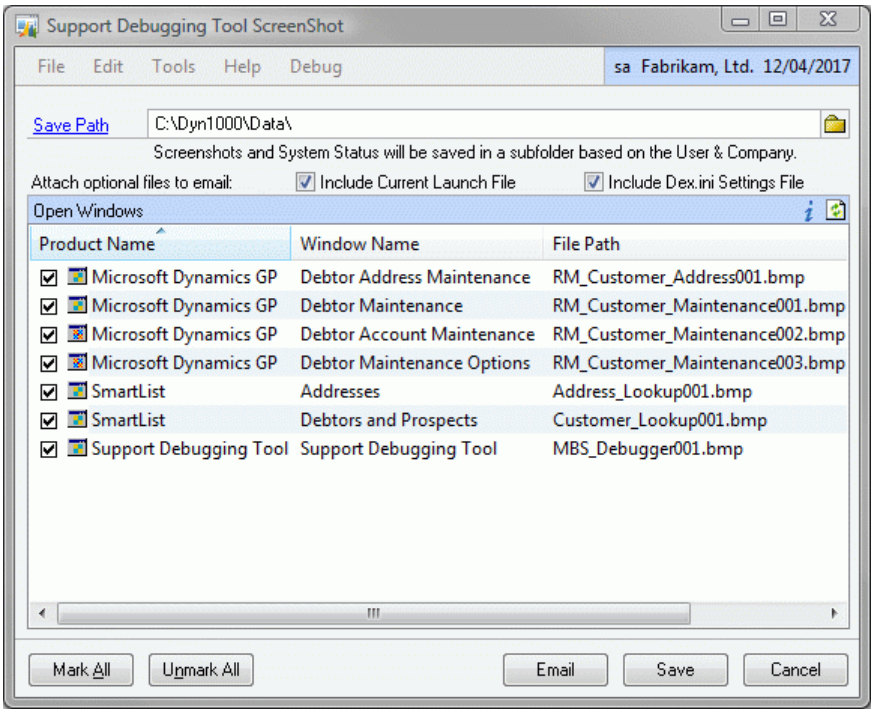
If the Standard Toolbar is displayed, you can launch ScreenShot from the Capture Screenshots button (highlighted below).

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 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.

Include Dex.ini Settings File

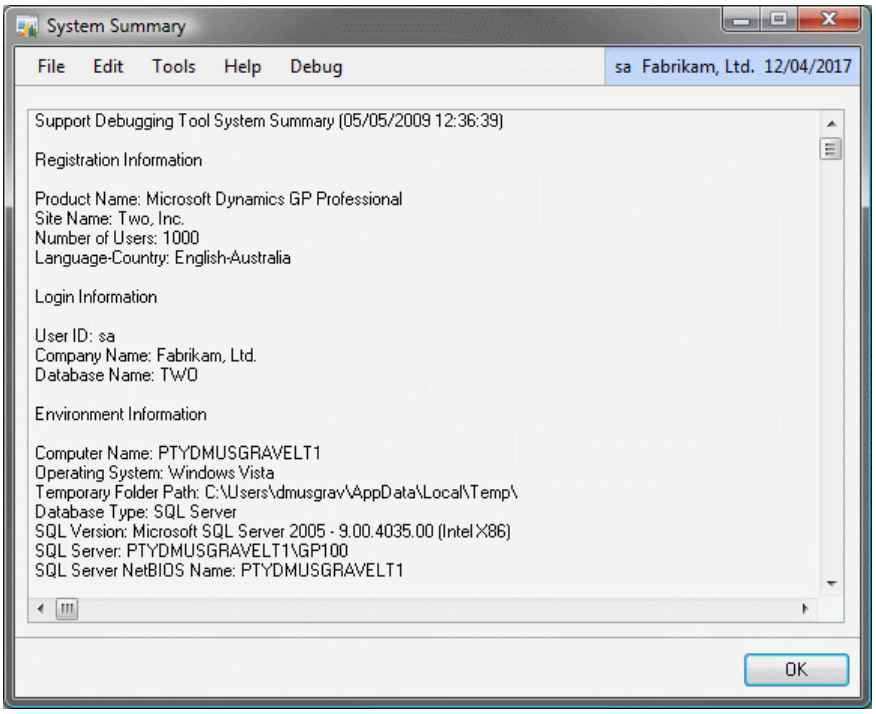
This checkbox tells ScreenShot whether to include the Dex.ini settings file as an attachment for the email. The default setting for this checkbox can be set up in the Administrator Settings window.

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.

Unmark All Button

This button will de-select all windows so that individual windows can be selected.

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

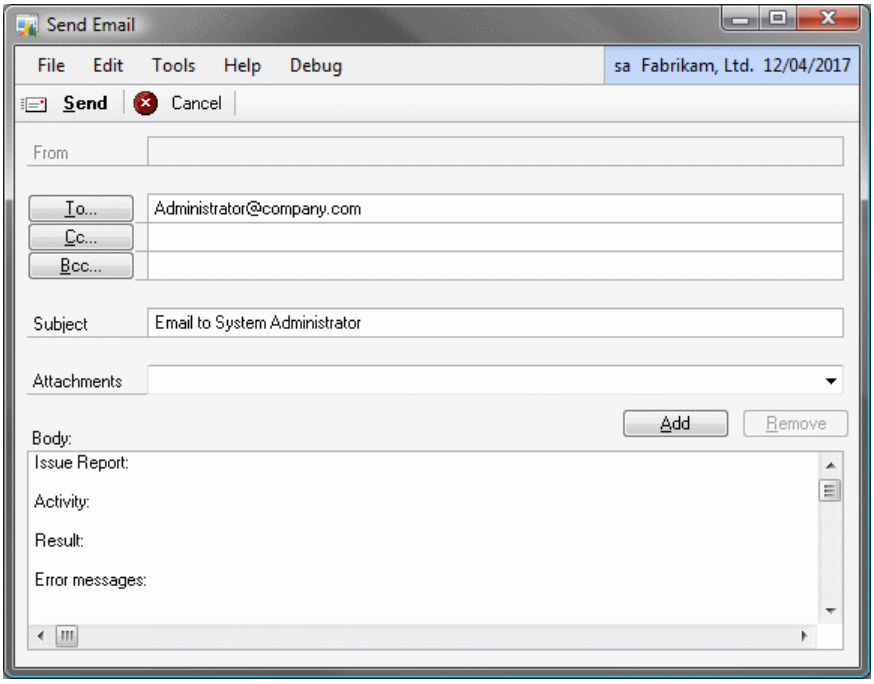
This button will close ScreenShot.

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 3: Advanced Mode Features

This chapter includes the following sections:

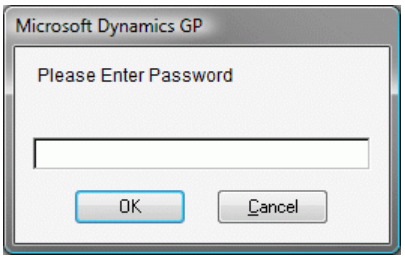
- *Advanced Mode Access*
- *Automatic Debugger Mode*
- *Dictionary Control*
- *XML Table Export*
- *XML Table Import*
- *Runtime Execute*
- *SQL Execute*
- *Configuration Maintenance*
- *Administrator Settings*
- *Dex.ini Configuration*

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

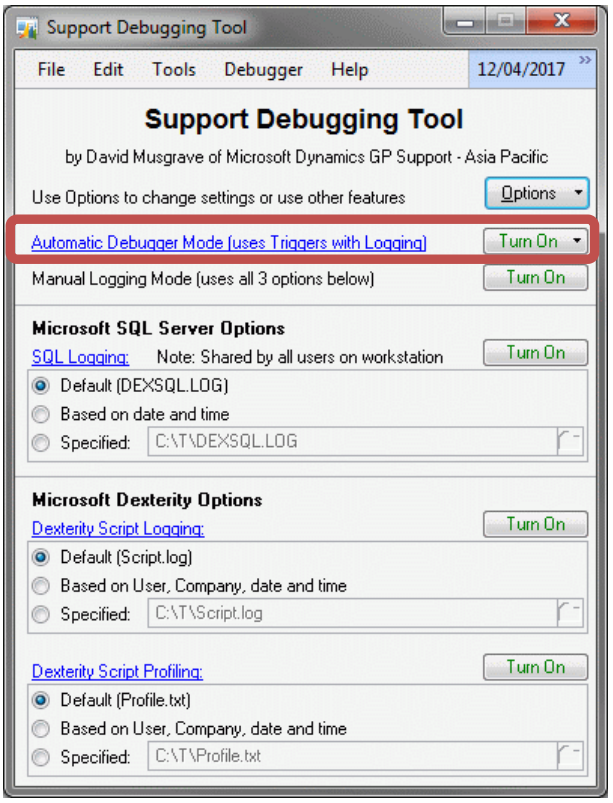
To use Automatic Debugger Mode you must create a trigger ID for each issue or exception condition being monitored. For each trigger ID, an event must be identified which can be used to look for the exception condition. For example, if the exception condition involves data in a table, the trigger event 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.

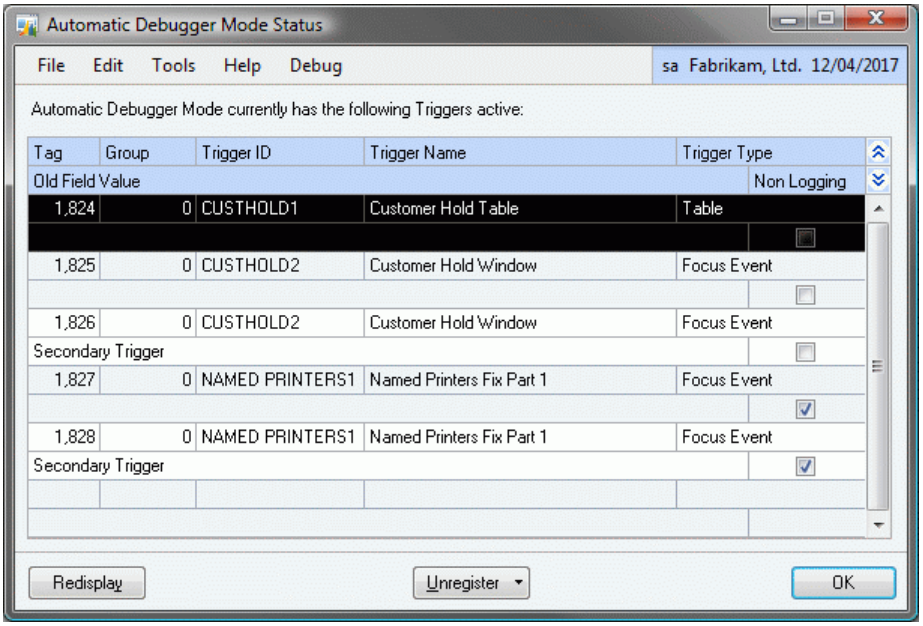


When manually activating the Automatic Debugger Mode, you can select whether to activate:

- 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

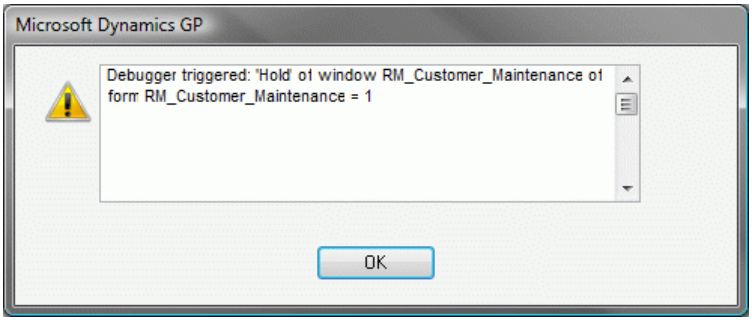
  This file will contain the exported table record.

- 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 Microsoft Dynamics GP application folder (v8.00 and v9.00), or the data subfolder beneath the Microsoft Dynamics GP application folder (v10.00 or later). 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).

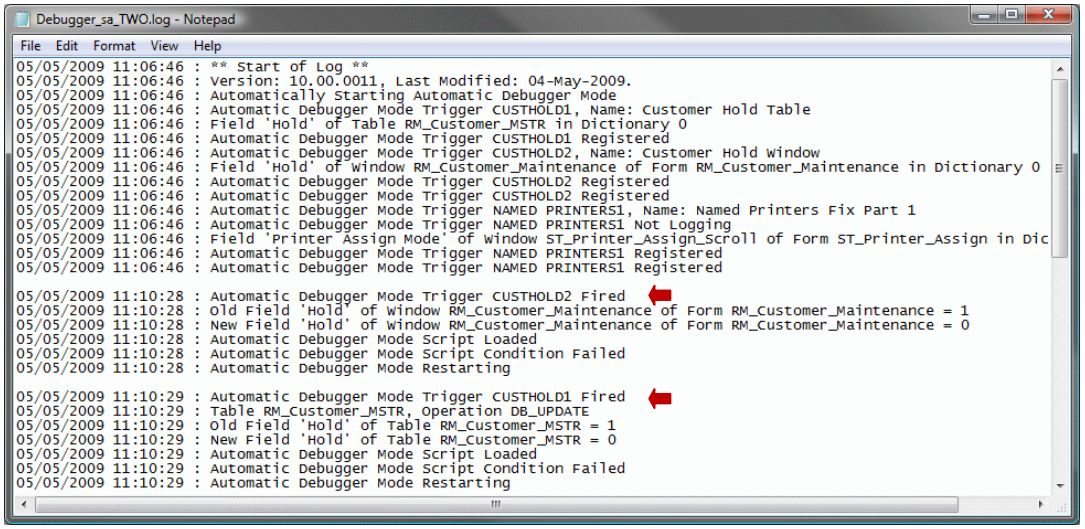
If the action to display a message was selected, a dialog with the display message specified will be shown.



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

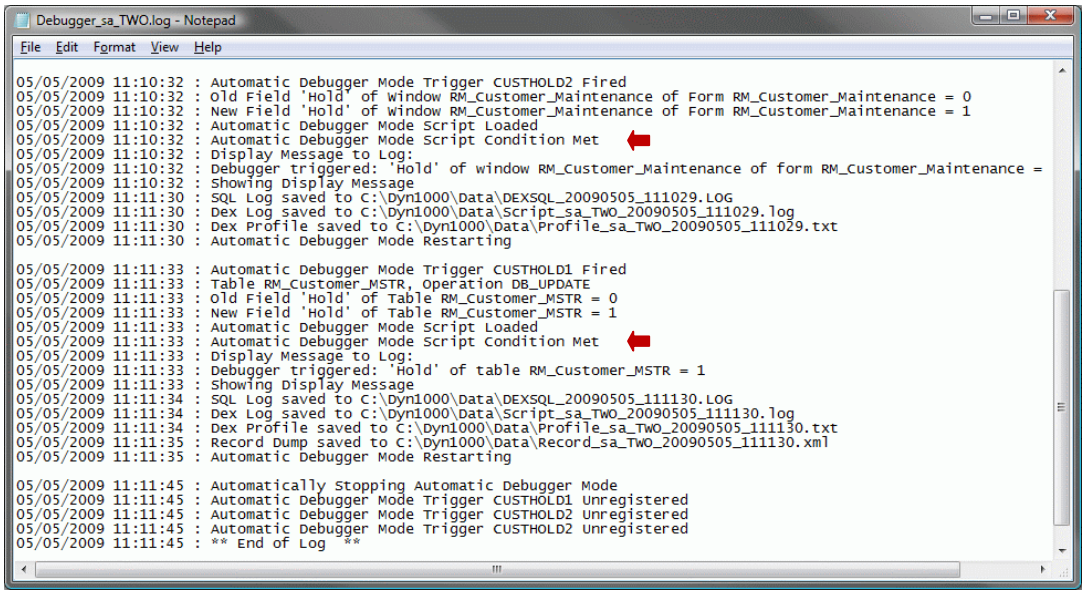
Log File

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.



```
Debugger_sa_TWO.log - Notepad
File Edit Format View Help
05/05/2009 11:06:46 : ** Start of Log **
05/05/2009 11:06:46 : Version: 10.00.0011, Last Modified: 04-May-2009.
05/05/2009 11:06:46 : Automatically Starting Automatic Debugger Mode
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger CUSTHOLD1, Name: Customer Hold Table
05/05/2009 11:06:46 : Field 'Hold' of Table RM_Customer_MSTR in Dictionary 0
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger CUSTHOLD1 Registered
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger CUSTHOLD2, Name: Customer Hold Window
05/05/2009 11:06:46 : Field 'Hold' of Window RM_Customer_Maintenance of Form RM_Customer_Maintenance in Dictionary 0
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger CUSTHOLD2 Registered
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger CUSTHOLD2 Registered
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger NAMED PRINTERS1, Name: Named Printers Fix Part 1
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger NAMED PRINTERS1 Not Logging
05/05/2009 11:06:46 : Field 'Printer Assign Mode' of Window ST_Printer_Assign_Scroll of Form ST_Printer_Assign in Dic
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger NAMED PRINTERS1 Registered
05/05/2009 11:06:46 : Automatic Debugger Mode Trigger NAMED PRINTERS1 Registered

05/05/2009 11:10:28 : Automatic Debugger Mode Trigger CUSTHOLD2 Fired
05/05/2009 11:10:28 : old Field 'Hold' of Window RM_Customer_Maintenance of Form RM_Customer_Maintenance = 1
05/05/2009 11:10:28 : New Field 'Hold' of Window RM_Customer_Maintenance of Form RM_Customer_Maintenance = 0
05/05/2009 11:10:28 : Automatic Debugger Mode Script Loaded
05/05/2009 11:10:28 : Automatic Debugger Mode Script Condition Failed
05/05/2009 11:10:28 : Automatic Debugger Mode Restarting

05/05/2009 11:10:29 : Automatic Debugger Mode Trigger CUSTHOLD1 Fired
05/05/2009 11:10:29 : Table RM_Customer_MSTR, Operation DB_UPDATE
05/05/2009 11:10:29 : old Field 'Hold' of Table RM_Customer_MSTR = 1
05/05/2009 11:10:29 : New Field 'Hold' of Table RM_Customer_MSTR = 0
05/05/2009 11:10:29 : Automatic Debugger Mode Script Loaded
05/05/2009 11:10:29 : Automatic Debugger Mode Script Condition Failed
05/05/2009 11:10:29 : Automatic Debugger Mode Restarting
```

Below is an example log when the conditions were met.

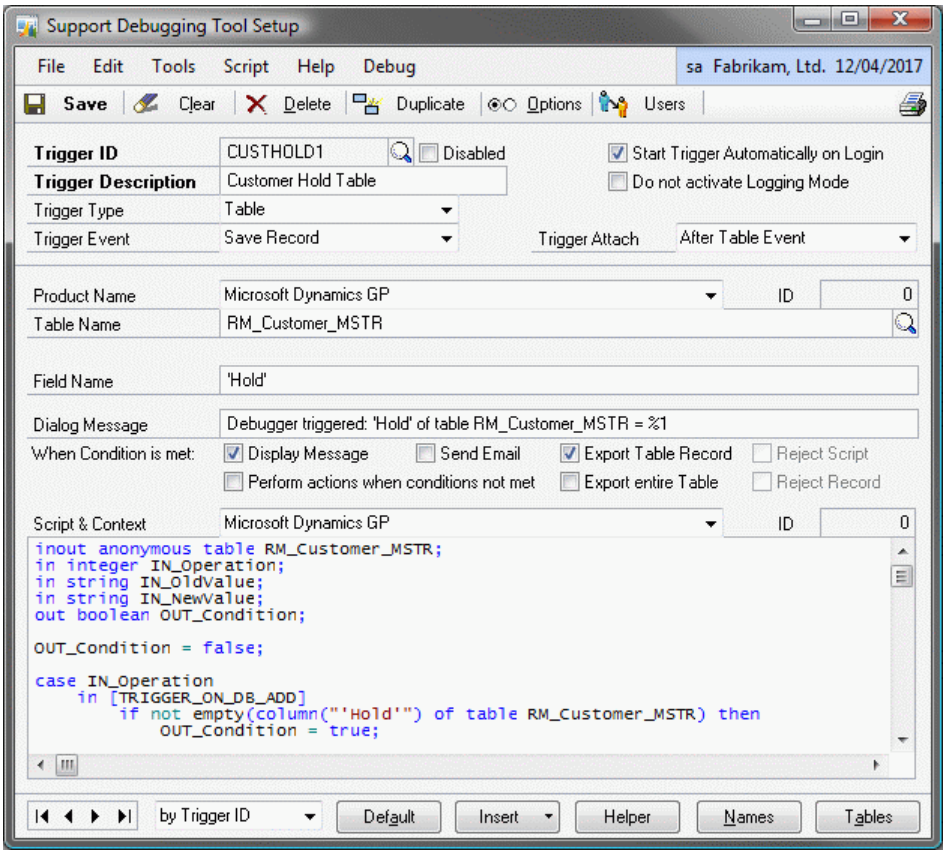


```
Debugger_sa_TWO.log - Notepad
File Edit Format View Help

05/05/2009 11:10:32 : Automatic Debugger Mode Trigger CUSTHOLD2 Fired
05/05/2009 11:10:32 : old Field 'Hold' of Window RM_Customer_Maintenance of Form RM_Customer_Maintenance = 0
05/05/2009 11:10:32 : New Field 'Hold' of Window RM_Customer_Maintenance of Form RM_Customer_Maintenance = 1
05/05/2009 11:10:32 : Automatic Debugger Mode Script Loaded
05/05/2009 11:10:32 : Automatic Debugger Mode Script Condition Met
05/05/2009 11:10:32 : Display Message to Log:
05/05/2009 11:10:32 : Debugger triggered: 'Hold' of window RM_Customer_Maintenance of form RM_Customer_Maintenance =
05/05/2009 11:10:32 : Showing Display Message
05/05/2009 11:11:30 : SQL Log saved to C:\Dyn1000\Data\DEXSQL_20090505_111029.LOG
05/05/2009 11:11:30 : Dex Log saved to C:\Dyn1000\Data\Script_sa_TWO_20090505_111029.log
05/05/2009 11:11:30 : Dex Profile saved to C:\Dyn1000\Data\Profile_sa_TWO_20090505_111029.txt
05/05/2009 11:11:30 : Automatic Debugger Mode Restarting

05/05/2009 11:11:33 : Automatic Debugger Mode Trigger CUSTHOLD1 Fired
05/05/2009 11:11:33 : Table RM_Customer_MSTR, Operation DB_UPDATE
05/05/2009 11:11:33 : Old Field 'Hold' of Table RM_Customer_MSTR = 0
05/05/2009 11:11:33 : New Field 'Hold' of Table RM_Customer_MSTR = 1
05/05/2009 11:11:33 : Automatic Debugger Mode Script Loaded
05/05/2009 11:11:33 : Automatic Debugger Mode Script Condition Met
05/05/2009 11:11:33 : Display Message to Log:
05/05/2009 11:11:33 : Debugger triggered: 'Hold' of table RM_Customer_MSTR = 1
05/05/2009 11:11:33 : Showing Display Message
05/05/2009 11:11:34 : SQL Log saved to C:\Dyn1000\Data\DEXSQL_20090505_111130.LOG
05/05/2009 11:11:34 : Dex Log saved to C:\Dyn1000\Data\Script_sa_TWO_20090505_111130.log
05/05/2009 11:11:34 : Dex Profile saved to C:\Dyn1000\Data\Profile_sa_TWO_20090505_111130.txt
05/05/2009 11:11:35 : Record Dump saved to C:\Dyn1000\Data\Record_sa_TWO_20090505_111130.xml
05/05/2009 11:11:35 : Automatic Debugger Mode Restarting

05/05/2009 11:11:45 : Automatically Stopping Automatic Debugger Mode
05/05/2009 11:11:45 : Automatic Debugger Mode Trigger CUSTHOLD1 Unregistered
05/05/2009 11:11:45 : Automatic Debugger Mode Trigger CUSTHOLD2 Unregistered
05/05/2009 11:11:45 : Automatic Debugger Mode Trigger CUSTHOLD2 Unregistered
05/05/2009 11:11:45 : ** End of Log  **
```

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.



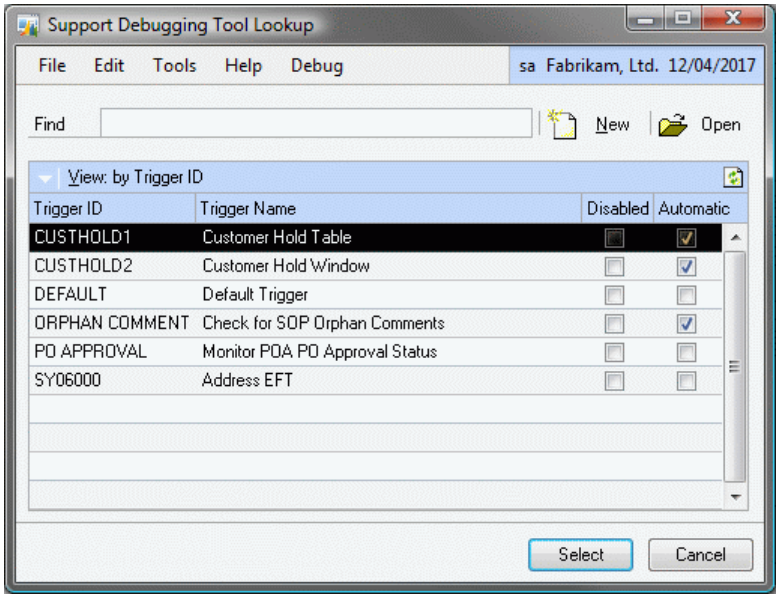
The window is divided into four sections; the trigger header section, the resource selection section, the action section and the scripting section.

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.



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
- Procedure
- Function
- Focus Event
- Warning Dialog
- Timed Event

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
- 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
- Warning Dialog
  - Warning Dialog
- Timed Event
  - Every 1 Minute
  - Every 5 Minutes
  - Every 10 Minutes
  - Every 15 Minutes
  - Every 30 Minutes
  - Every 60 Minutes

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
- Procedure
  - Before Original
  - After Original
- Function
  - Before Original
  - After Original
- Focus Event
  - Before Original
  - After Original
- Warning Dialog
  - Before Original
  - After Original
- Timed Event
  - After Timed Event

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.

The following is a description of the individual resource selection fields on the window. 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.

Table Name
This field contains the technical name for the table selected.

Form Name
This field contains the technical name for the form selected.

Window Name
This field contains the technical name for the window 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.

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

Below is a description of the individual action fields on the window. These actions will be processed when the trigger fires and the conditional script returns true or if the Perform actions when conditions not met checkbox is selected.

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.

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

Send Email
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.

Export Table Record
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.

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.

Perform actions when conditions 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.

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

Reject Record

When this checkbox is selected, a reject record command will be issue 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.

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.

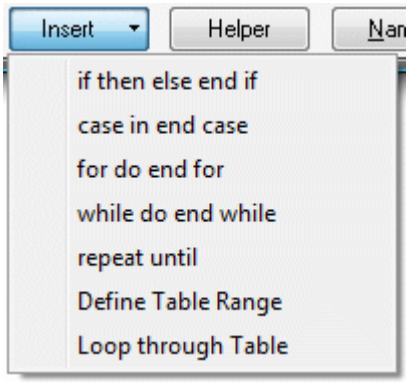
The following is a description of the additional buttons on the window:

Default Button

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

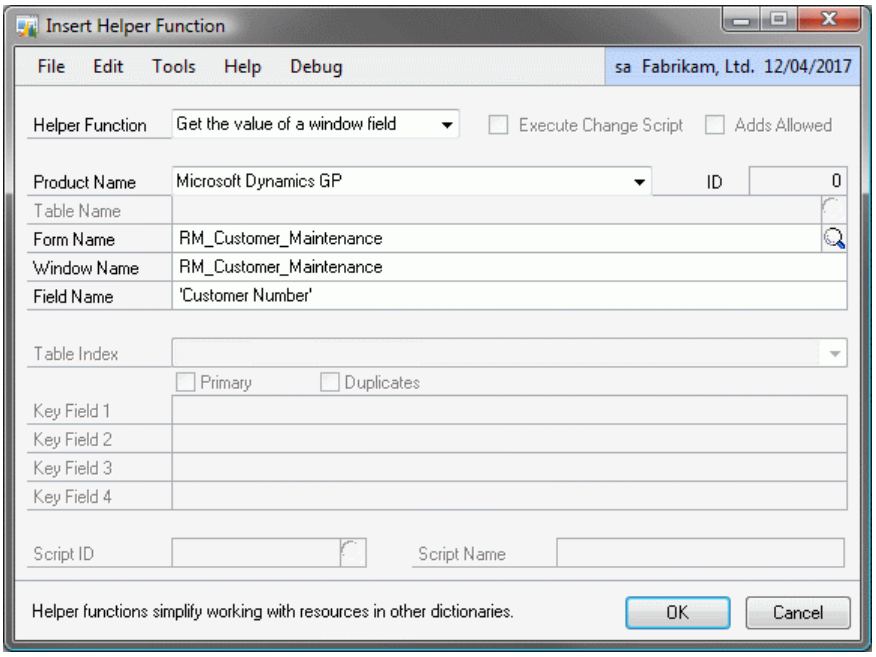
Insert Button

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.

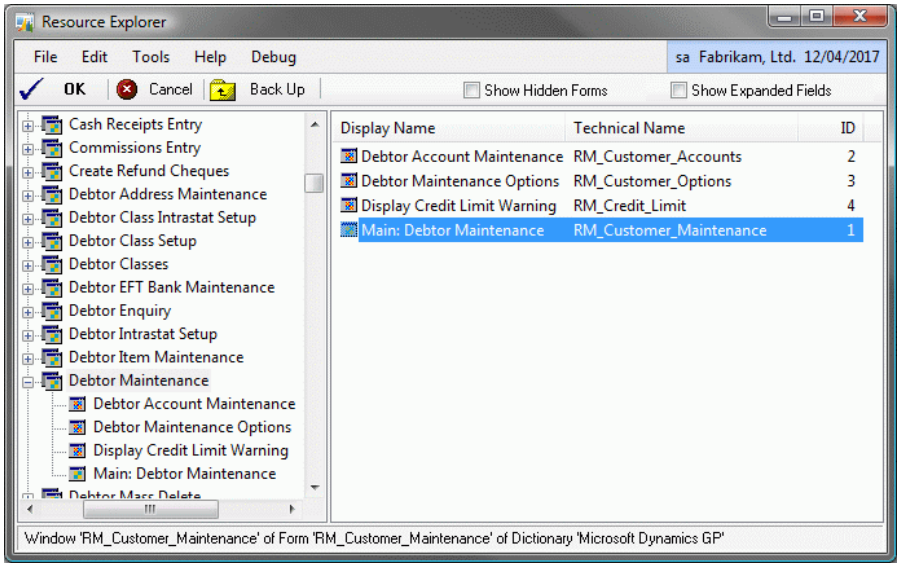


Helper functions can be used to read or write a window or table field in any window or table from any loaded dictionary. When setting a window field you can select whether to execute the field's change script. When setting a table field you can select whether adding a new record is allowed.

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

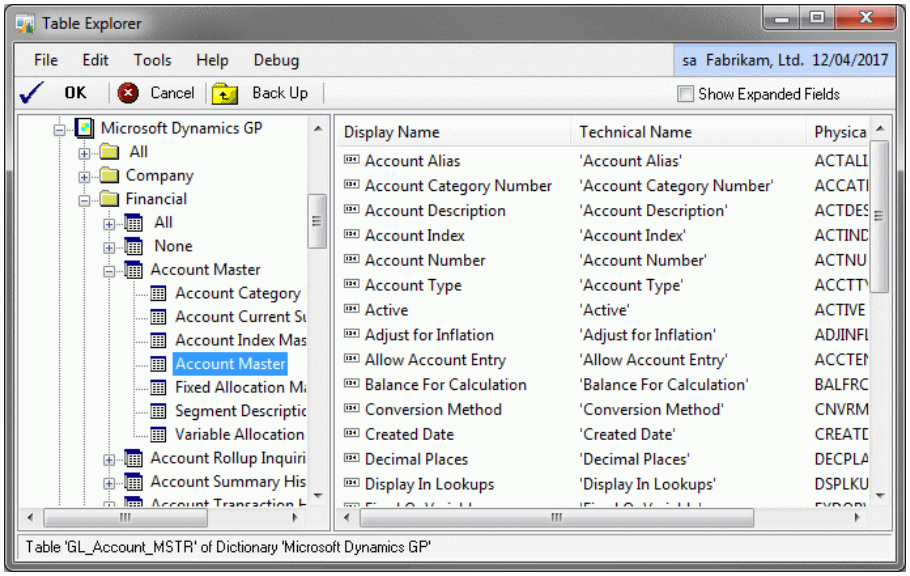
Use this button to insert a form, window or field resource into the script. Once clicked the Resource Explorer window will open.



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

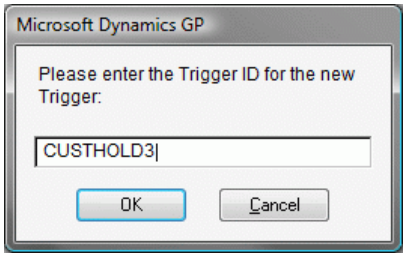
Use this button to insert a table or field resource into the script. Once clicked the Table Explorer window will open.



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.

Duplicate Button

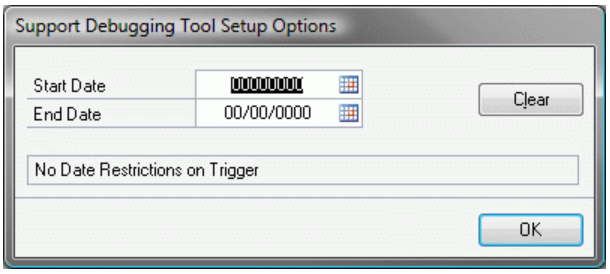
Use this button to duplicate the current trigger ID and create a new trigger ID. This is useful when an existing trigger ID is very similar to the new one you want to create.



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

Options Button

Use this button to specify additional options for the triggers. Once clicked the Support Debugging Tool Setup Options window will open.

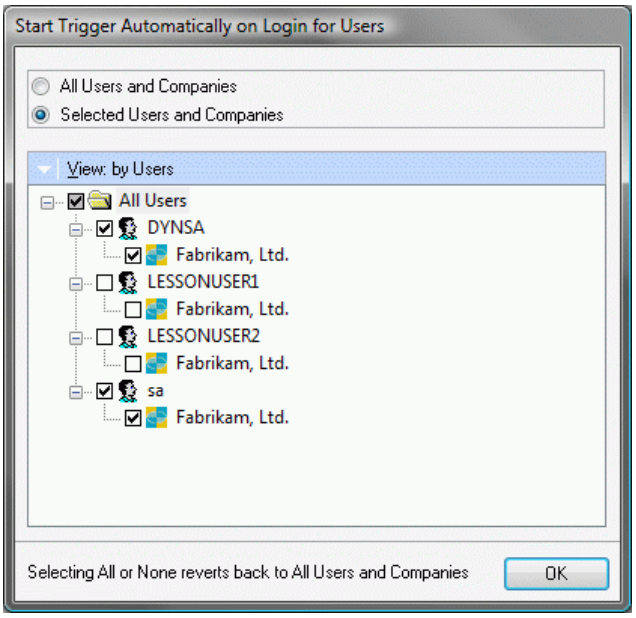


You can specify a Start Date and End Date to restrict the dates that a trigger will automatically start.

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.

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.



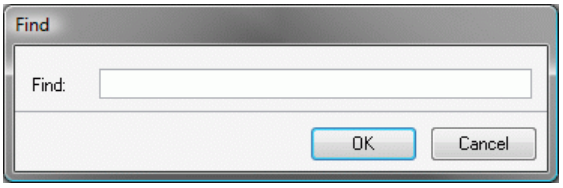
You can view this window by users or by companies and navigate the tree to select the user and company combinations as required.

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 following is a description of the Script menu available for the window. For v8.00 and v9.00 the menu can be found under the Extras menu:

Find …

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

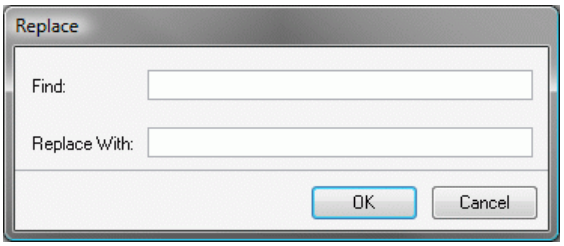


Find Next

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

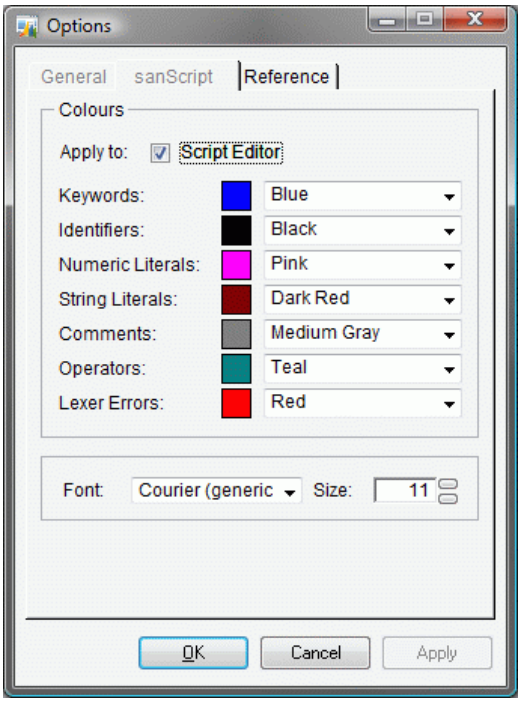
Use this menu option to replace and find the next occurrence. Control-B can be used as a shortcut.

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.



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.

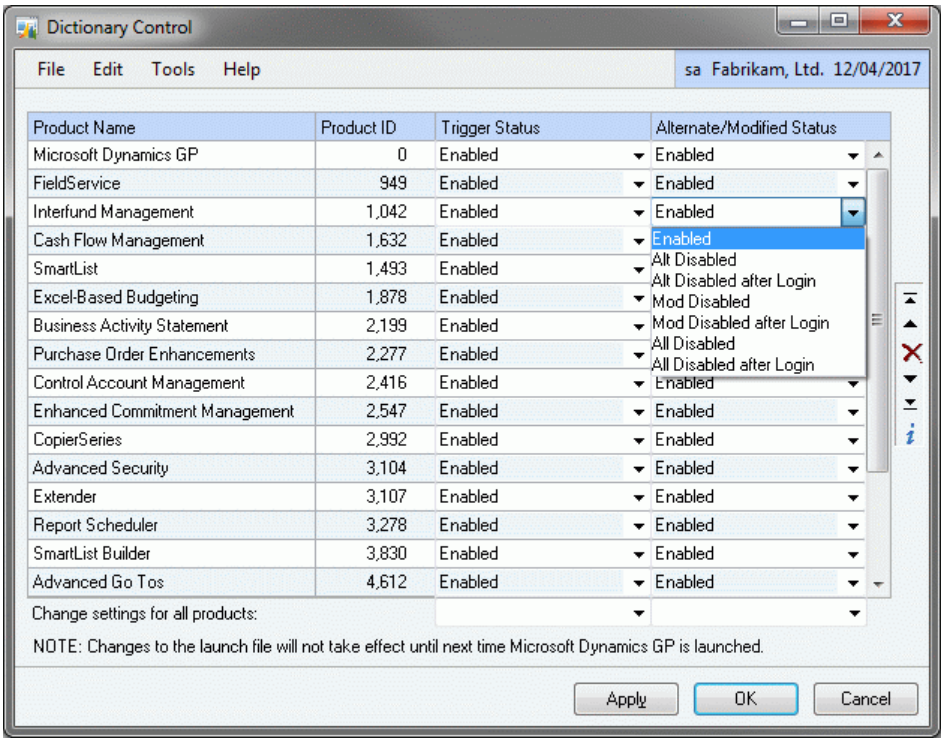
Changes made to tables using tools or applications other than Dexterity will not be picked up by the Support Debugging Tool. This can include table changes made directly by SQL Query Analyzer, pass-through SQL commands, SQL stored procedures, SQL triggers, or updates from eConnect, Integration Manager's 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.

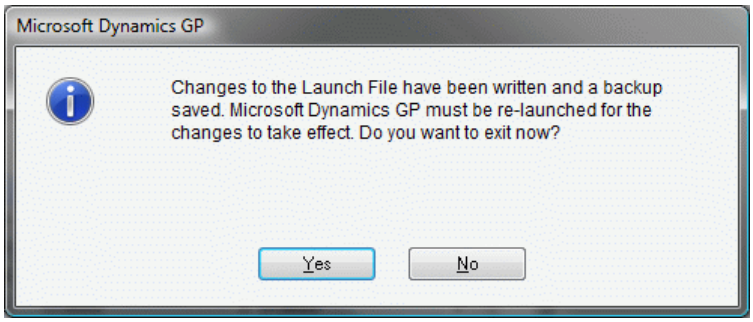
The drop-down lists at the bottom of the window can be used to change settings for all dictionaries.

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.

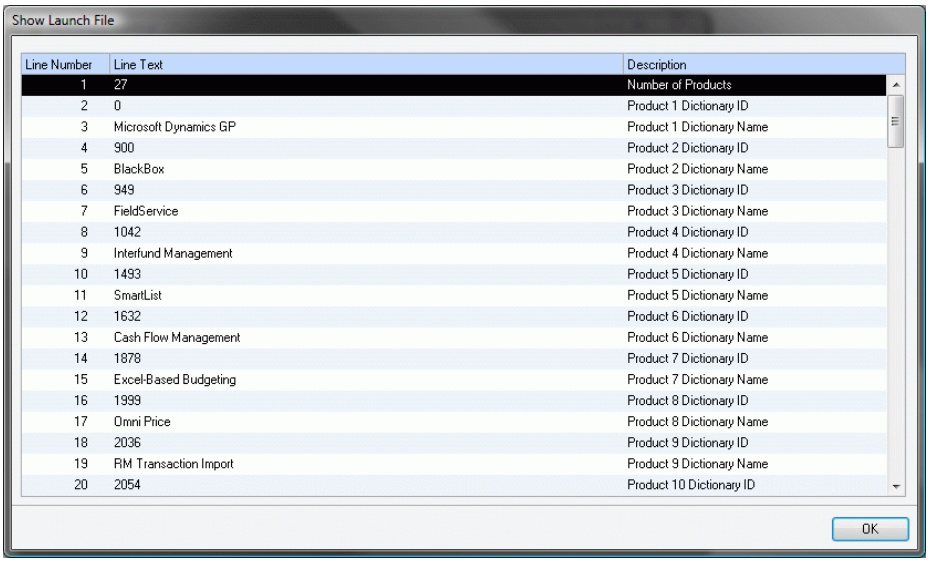
Dictionary Control allows the order of the products to be changed using the Top, Up, Down and Bottom buttons. You can also remove a product with the Delete Button. Any changes to the Dynamics.set launch file will be saved when OK is clicked. You will be requested to restart Microsoft Dynamics GP after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



Line Number	Line Text	Description
1	27	Number of Products
2	0	Product 1 Dictionary ID
3	Microsoft Dynamics GP	Product 1 Dictionary Name
4	900	Product 2 Dictionary ID
5	BlackBox	Product 2 Dictionary Name
6	949	Product 3 Dictionary ID
7	FieldService	Product 3 Dictionary Name
8	1042	Product 4 Dictionary ID
9	Interfund Management	Product 4 Dictionary Name
10	1493	Product 5 Dictionary ID
11	SmartList	Product 5 Dictionary Name
12	1632	Product 6 Dictionary ID
13	Cash Flow Management	Product 6 Dictionary Name
14	1878	Product 7 Dictionary ID
15	Excel-Based Budgeting	Product 7 Dictionary Name
16	1999	Product 8 Dictionary ID
17	Omni Price	Product 8 Dictionary Name
18	2036	Product 9 Dictionary ID
19	RM Transaction Import	Product 9 Dictionary Name
20	2054	Product 10 Dictionary ID

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.

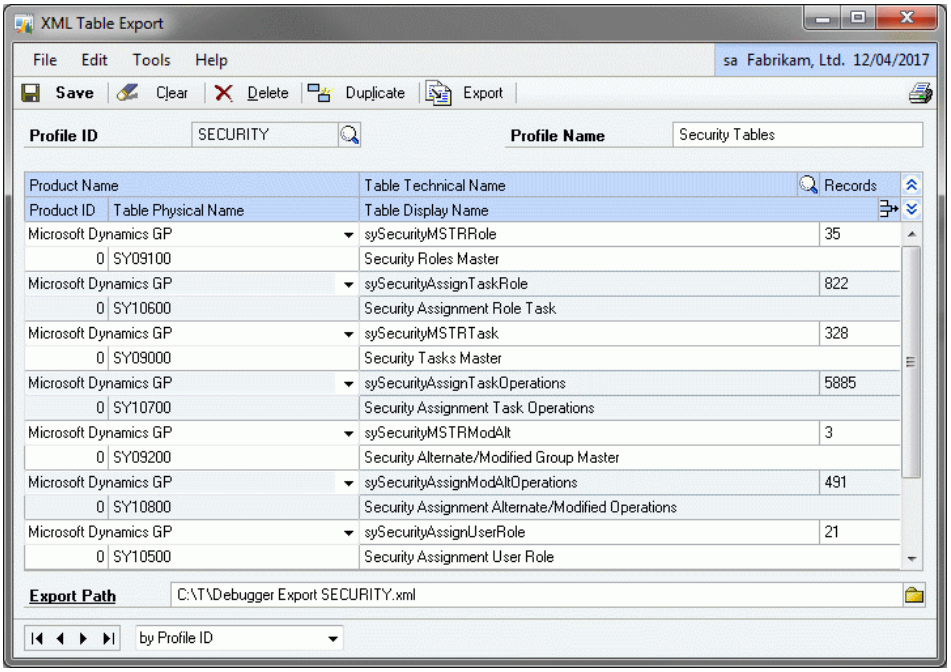
Dictionary Control cannot be used to disable alternate windows and forms or triggers in the Support Debugging Tool. As the core Microsoft Dynamics GP dictionary cannot have alternate windows, Dictionary Control cannot be used to disable alternate windows.

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.

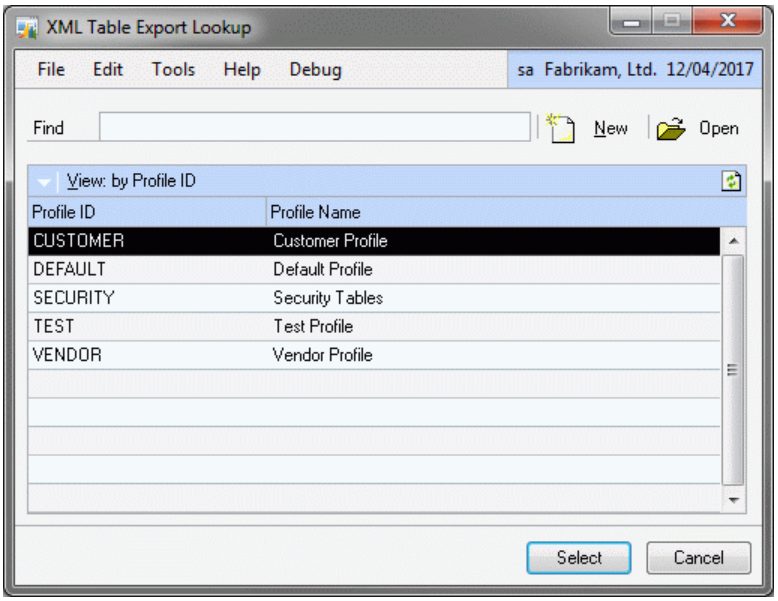
Using separate Profile IDs allows multiple sets of tables to be stored for particular related groups.



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.



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

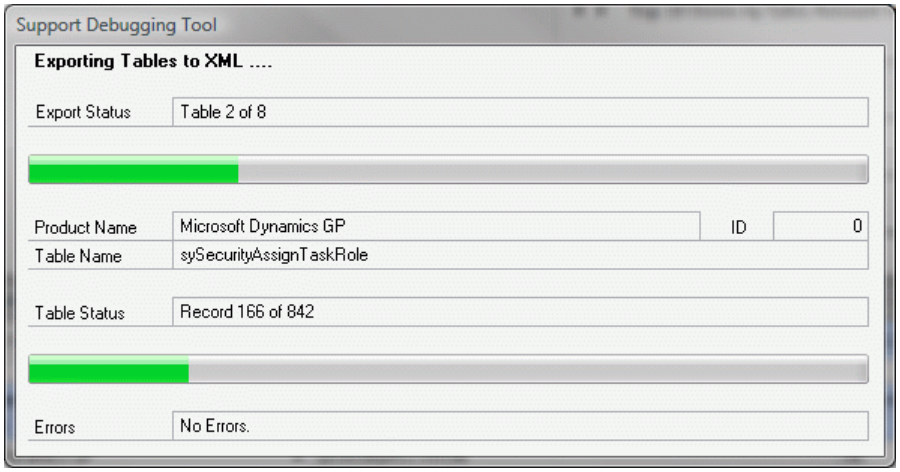
Use this button to duplicate the current profile ID to a new profile ID. This is useful when an existing profile ID is very similar to the new one you want to create.

Export Button

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

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.

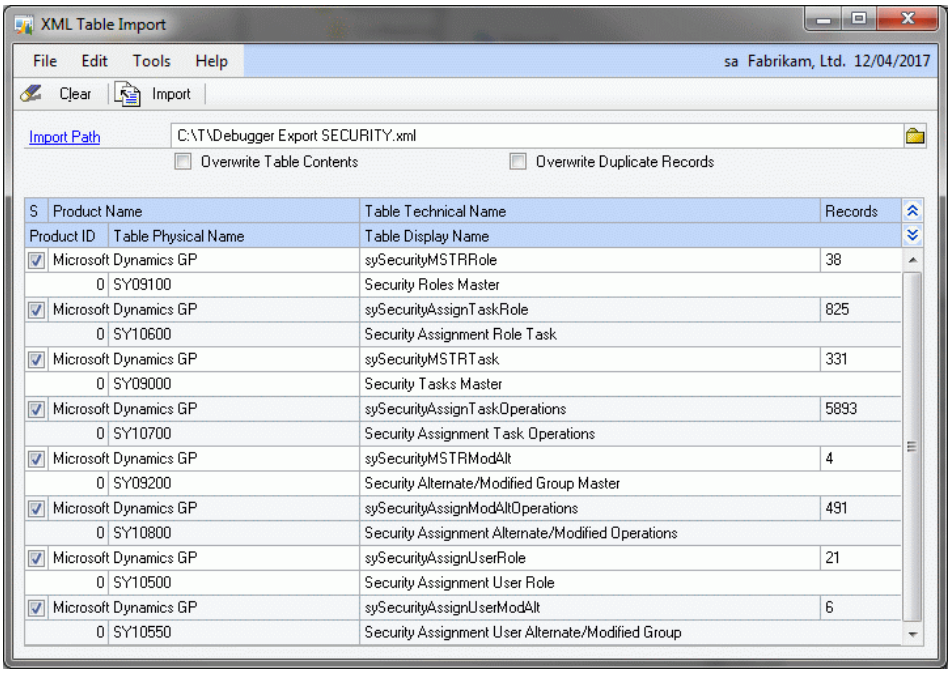
There is no data validation or business logic checking when data is imported using XML Table Import. This is similar to the Dexterity Table Import Utility. It is best to ensure that all related tables are exported by XML Table Export.

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.

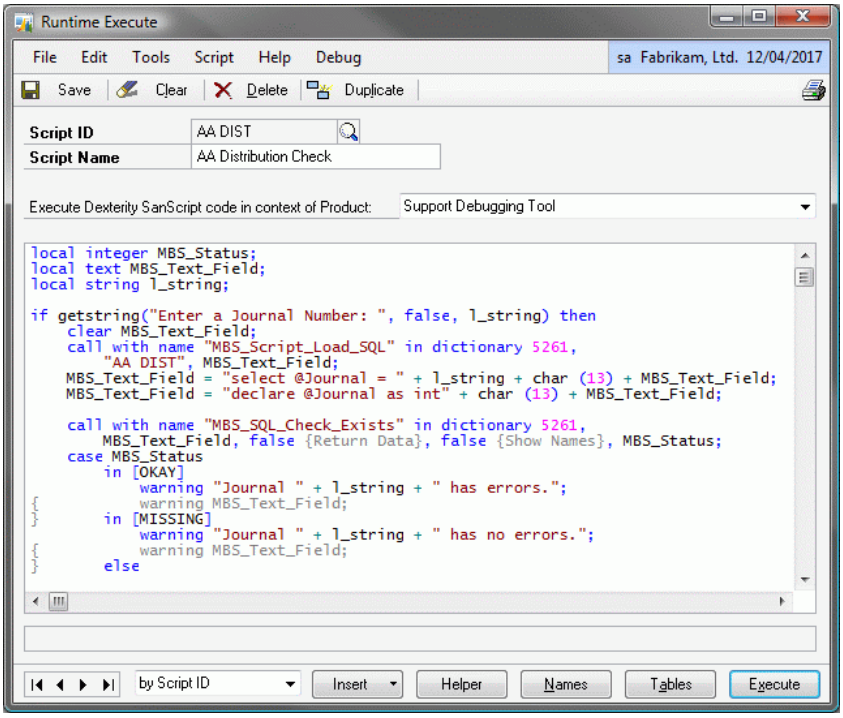
If both the XML Table Export window and XML Table Import window are open, the import path will default to the export path from the XML Table Export window.

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.

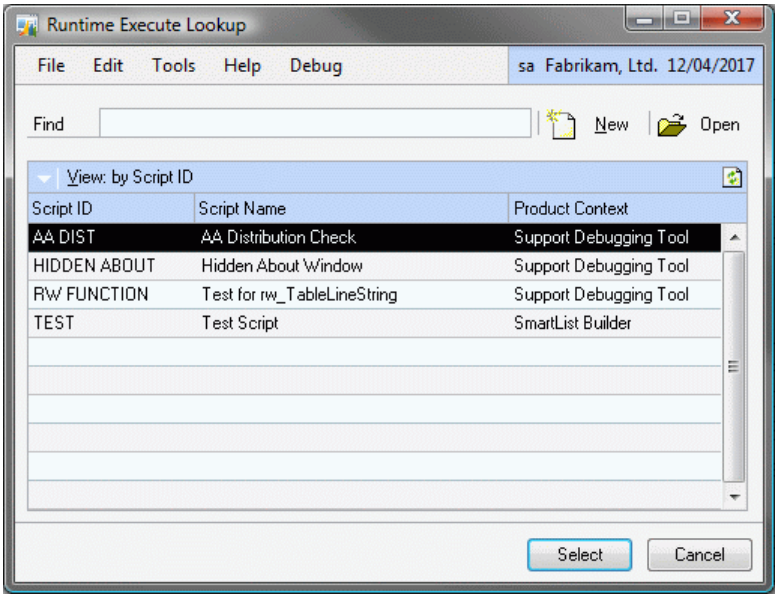
Script IDs created in this window can be loaded and executed from an Automatic Debugger Mode trigger or another Runtime Execute script. This allows code re-use in a similar fashion to having multiple procedure calls.



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.



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.

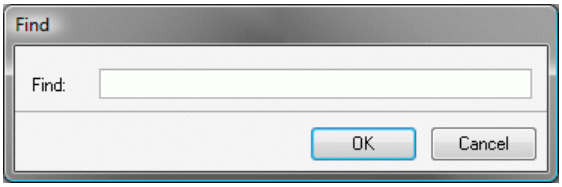
Duplicate Button

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. For v8.00 and v9.00 the menu can be found under the Extras menu:

Find …

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

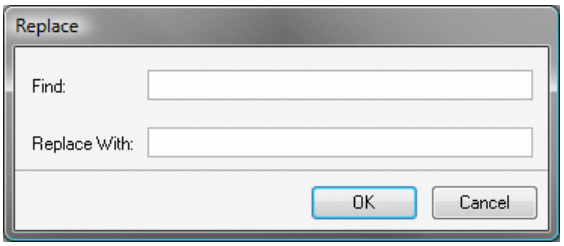


Find Next

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

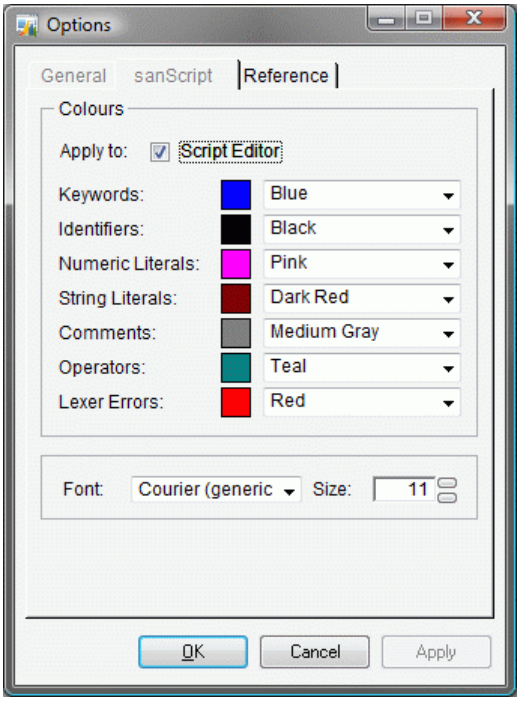
Use this menu option to replace and find the next occurrence. Control-B can be used as a shortcut.

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.

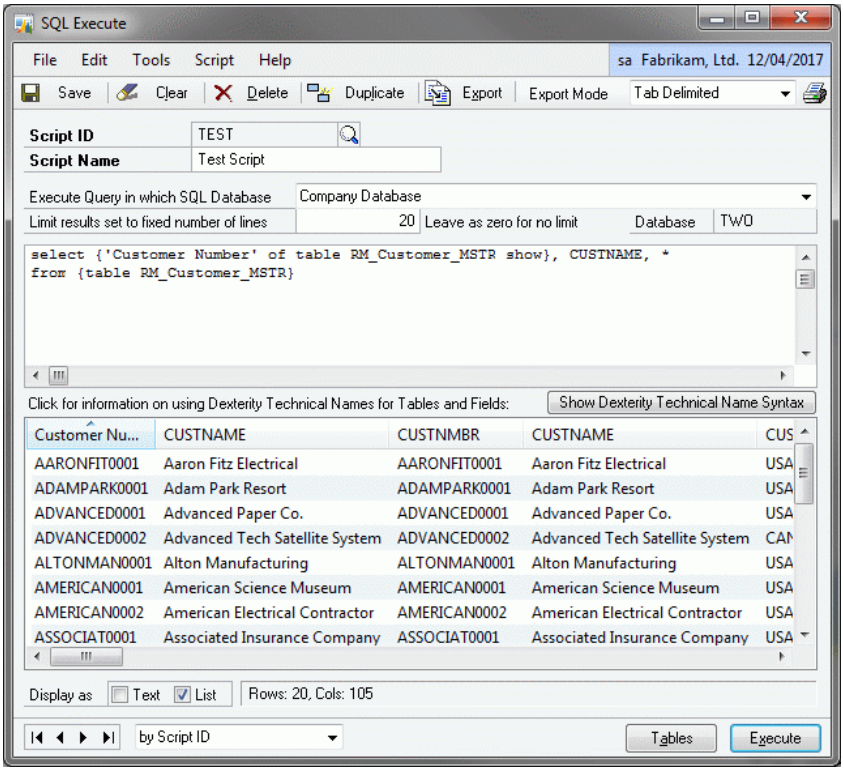


SQL Execute

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}
```

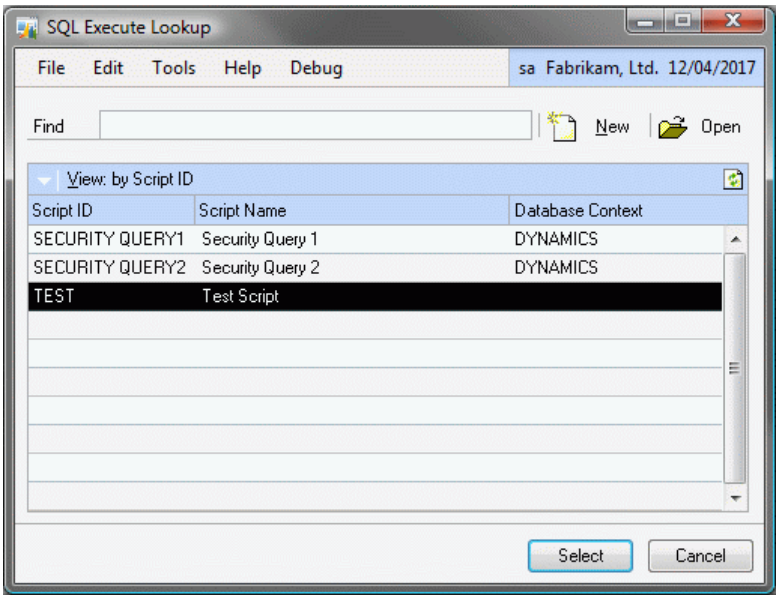
is converted to

```
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.



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

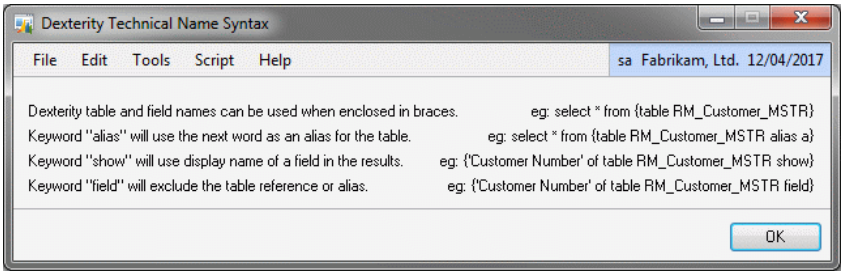
This text field contains the Transact SQL statements to be executed.

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.

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.

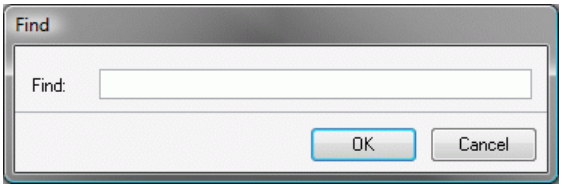
Export Mode

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. For v8.00 and v9.00 the menu can be found under the Extras menu:

Find …

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

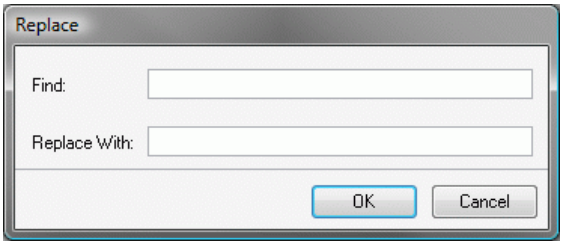


Find Next

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

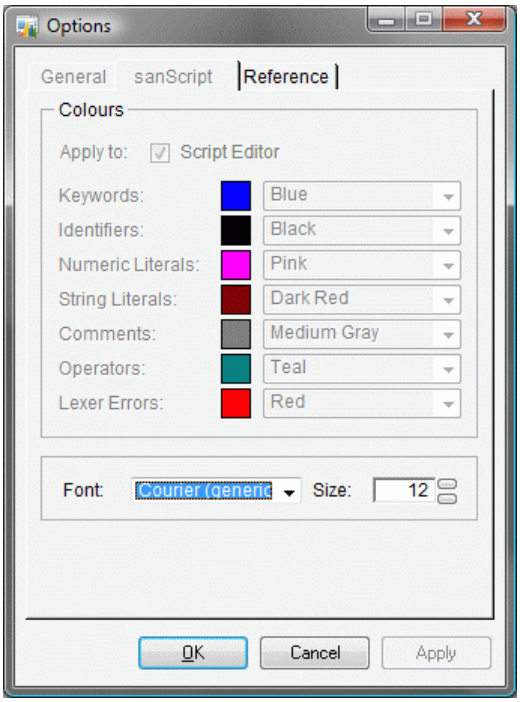
Use this menu option to check the syntax of Dexterity resource names contained in curly braces in the current script. Control-S can be used as a shortcut.

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.

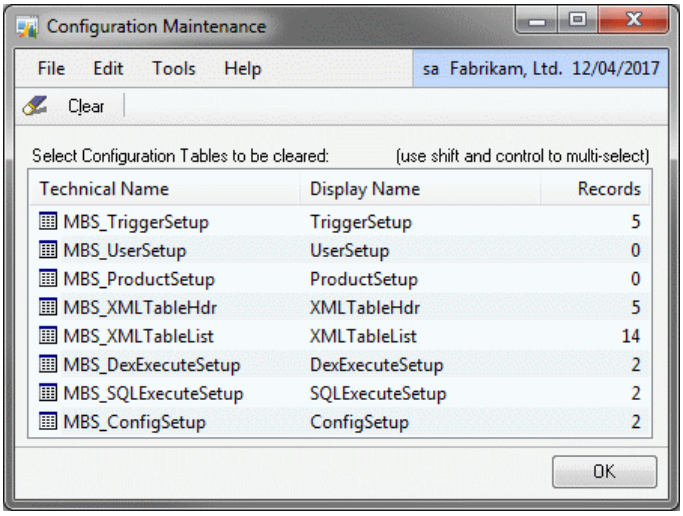


The Color options are disabled because the syntax highlighting is not available for SQL scripts.

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.

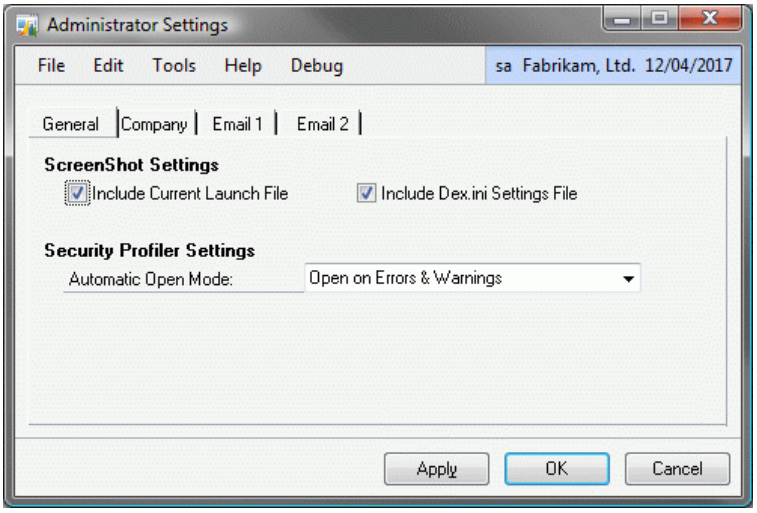
Administrator Settings

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:

Include Current Launch File

This checkbox specifies the default setting for ScreenShot.

Include Dex.ini Settings File

This checkbox specifies the default setting for ScreenShot.

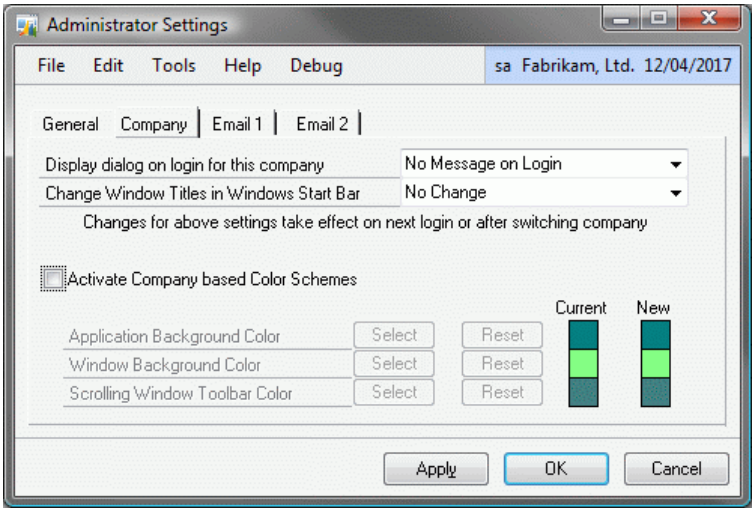
Automatic Open Mode

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.

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:

https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;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. For versions 8.00 and 9.00 this will only be the main application window. For version 10.00 and later all windows will be prefixed.

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.

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.

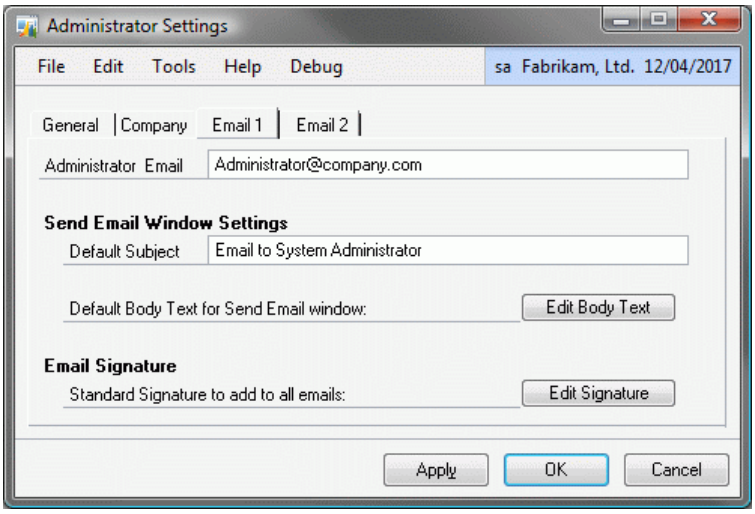
Once the option is activated, the colors for the Application Background Color, Window Background Color, and Scrolling Window Toolbar Color can be selected with the Select Buttons. The Reset Buttons can be used to restore the default colors.

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.

For version 10.00 and later, the application background is not visible because the homepage is displayed at all times. For these versions, the Application Background Color is used for the window Toolbars.

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)

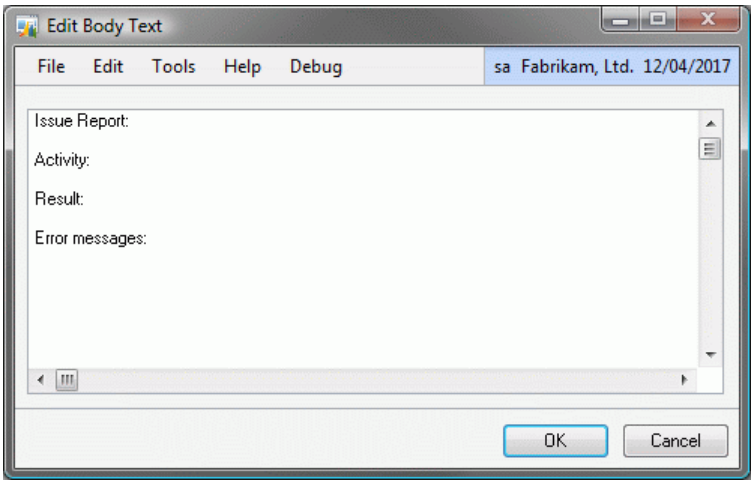
Default Subject

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.

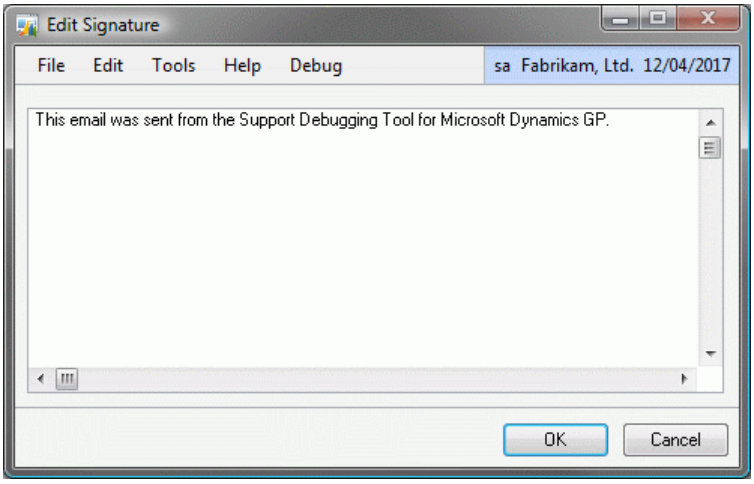
Click on the Edit Body Text Button to open the Edit Body Text window.



Standard Signature to add to all emails

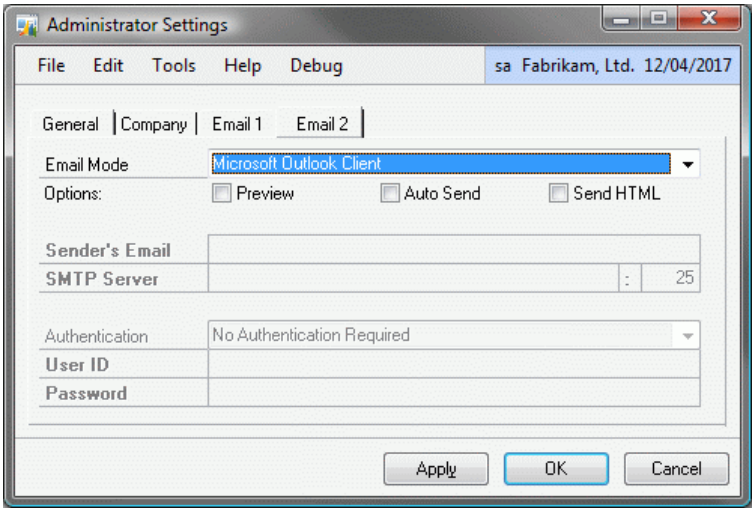
This button can be used to create a standard signature to add to the bottom of all emails sent from the Support Debugging Tool. If no signature is defined, the text in the screenshot below will be used.

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 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.

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.

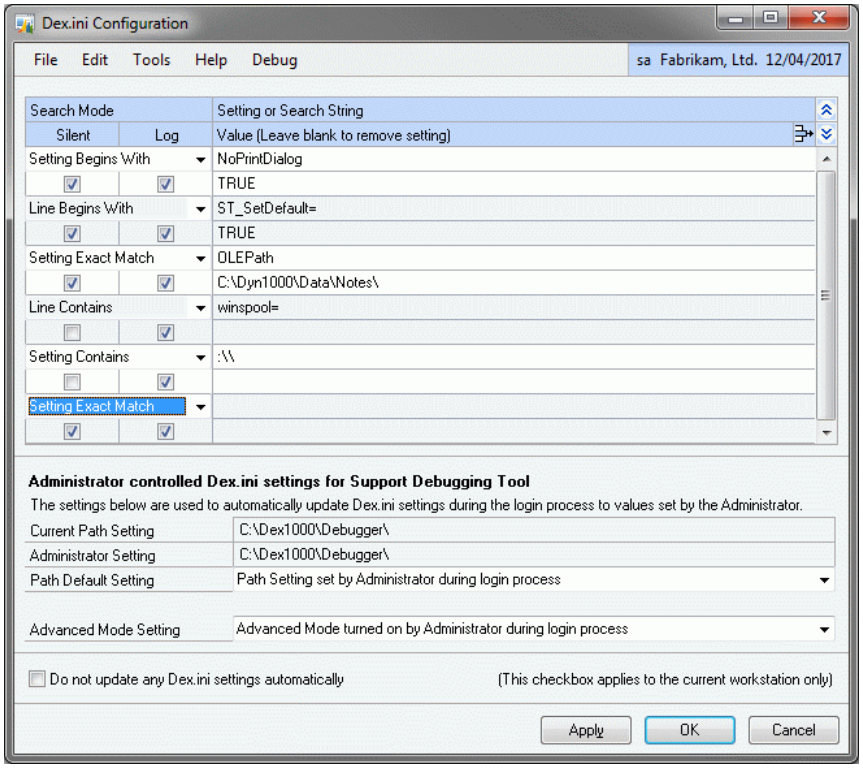
Dex.ini Configuration

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.

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.

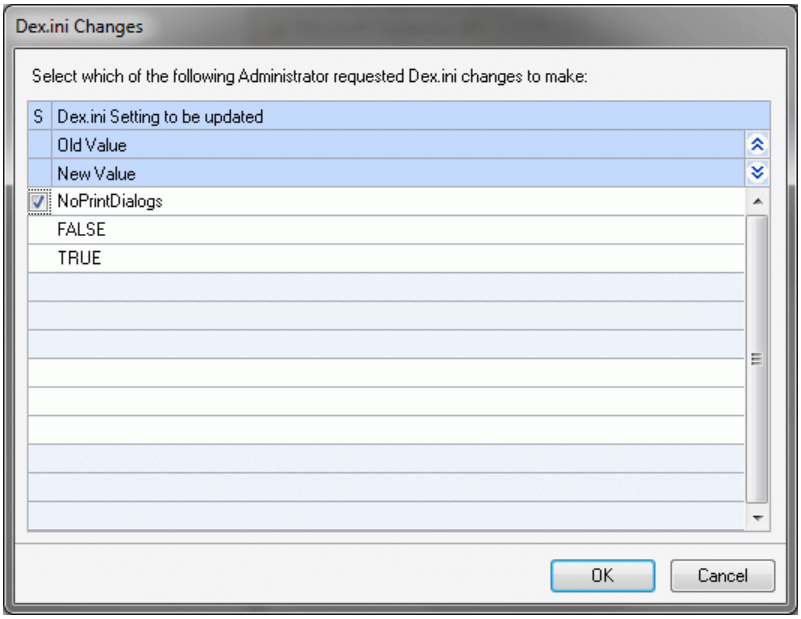
Do not update any Dex.ini settings automatically

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.

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 4: 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 Microsoft Dynamics GP application folder (v8.00 and v9.00), or the data subfolder beneath the Microsoft Dynamics GP application folder (v10.00 or later). 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 Microsoft Dynamics GP application folder (v8.00 and v9.00) or in the data subfolder beneath the Microsoft Dynamics GP application folder (v10.00 or later).

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_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_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.

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.

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_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_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_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.

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 FALSE.

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.

DebugRW
This setting is used to configure the Report Writer to create a debugging log file named DebugRW.txt that will appear in the Microsoft Dynamics GP application folder (v8.00 and v9.00) or in the data subfolder beneath the Microsoft Dynamics GP application folder (v10.00 or later).

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.

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 5: 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_Script_Load_Dex*
- *MBS_Script_Load_SQL*
- *MBS_Param_Set*
- *MBS_Param_Get*
- *MBS_Param_Del*
- *MBS_Param_DelAll*
- *MBS_Auto_Log*

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_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";
```

Chapter 6: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:

https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;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

About Support Debugging Tool, 13
Access Denied, 28
Activate Company based Color Schemes, 83
Activate Debug Logging for the Report Writer, 19
ActiveX Data Objects, 9, 61
Add Attachment Button, 42
Add Button, 42
Adds Allowed, 56, 101, 103, 105, 107
Administrator Email, 42, 54, 84
Administrator Setting, 89
Administrator Settings, 29, 30, 39, 40, 41, 42, 54, 77, 81, 82, 83, 84, 85, 86, 87, 93
  Activate Company based Color Schemes, 83
  Administrator Email, 41, 42, 54, 84
  Application Background Color, 83
  Apply Button, 83
  Authentication, 87
  Auto Send, 42, 86
  Automatic Open Mode, 30, 81
  Body Text, 41
  Change Window Titles in Windows Start Bar, 83
  Company Tab, 82
  Default Body Text, 41, 42
  Default Body Text for Send Email window, 85
  Default Subject, 41, 42, 84
  Display dialog on login for this company, 82
  Email 1 Tab, 84
  Email 2 Tab, 86
  Email Mode, 86
  General Tab, 81
  Include Current Launch File, 39, 81
  Include Dex.ini Settings File, 39, 81
  MAPI Compliant Client, 86
  Microsoft Outlook Client, 86
  OK Button, 83
  Outlook, 86
  Password, 87
  Preview, 41, 42, 86
  Reset Buttons, 83
  Scrolling Window Toolbar Color, 83
  Select Buttons, 83
  Send HTML, 86
  Sender’s Email, 41, 87
  SMTP Server, 87
  SMTP Server Port, 87
  SMTP Server via CDO, 86
  Standard Signature to add to all emails, 85
  Subject, 41
  Terminal Server, 86
  User ID, 87
  Window Background Color, 83
  Windows Start Bar, 83
ADO, 9, 61
Advanced Mode, 1, 6, 7, 12, 15, 43, 44, 45, 62, 65, 68, 70, 74, 80, 81, 88, 89, 91, 110, 111
  Access, 44
  Administrator Settings, 81
  Automatic Debugger Mode, 45
  Configuration Maintenance, 80
  Dex.ini Configuration, 88, 91
  Dictionary Control, 62
  Runtime Execute, 70, 110
  SQL Execute, 74, 111
  SQL Server, 44
  System Password, 44
  XML Table Export, 65
  XML Table Import, 68
Advanced Mode Setting, 89
Advanced Security, 6, 31
After Original, 53
After Table Event, 53
After Timed Event, 53
alias keyword, 74
All Except Disabled, 46
All Users and Companies, 59
Allow mismatched Dex.dic version, 17
AllowWrongDex, 17, 95
Alternate, 28, 62
Alternate/Modified Forms and Reports, 31, 33
Alternate/Modified Status, 62
Application Background Color, 83
Application Level Security, 28, 29
ApplicationName, 17, 94
Apply Button, 83
Associated Tables Button, 23
Attachments, 42
Authentication, 87
Auto Send, 42, 86
AutoInstallChunks, 17, 94
Automatic Debugger Mode, 45
  All Except Disabled, 46
  All Users and Companies, 59
  Automatic Debugger Mode Status, 47, 53
  Automatic Start Only, 46
  Check Syntax, 60
  Conditional Script, 45, 48, 54, 55
  DEFAULT, 46, 50, 80
  Default Button, 55
  DEFAULT only, 46
  DialogMessage, 54
  Dictionary ID, 53, 55
  Disabled, 53
  Display Message, 48, 54
  Do not activate Logging Mode, 53
  Duplicate Button, 58
  End Date, 58
  Export entire Table, 54
  Export entire Table when Condition is met, 48
  Export Table Record, 54
  Export Table Record when Condition is met, 48
  Extras Menu, 59, 72, 78
  Field, 54
  Field Name, 54
  Find …, 59
  Find Next, 59
  Font Size, 60
  Font Style, 60
  Form, 54
  Form Name, 54
  Function, 54
  Function Name, 54

Helper Button, 56
Helper Function Assistant, 56
How to Setup, 45
Insert Button, 55
Insert Helper Function, 56
Introduction, 45
Log File, 49
Message, 55
Names Button, 56
Non Logging All Except Disabled, 46
Non Logging Automatic Start Only, 46
Non Logging Triggers, 46, 47, 53, 54, 55
Old Field Value, 47
Options, 60
Options Button, 58
OUT_Condition, 55
Perform actions when conditions not met, 54
Procedure, 54
Procedure Name, 54
Product ID, 53
Product Name, 53
Registration, 46
Reject Record, 55
Reject Script, 54
Replace …, 59
Replace and Find Next, 60
Resource Explorer, 53, 56
Restriction of Scope, 61
Runtime Execute, 55, 70
Script Context, 55
Script Context ID, 55
Script Menu, 59, 72, 78
Selected Users and Companies, 59
Send Email, 54
Setup, 50, 116
SQL Execute, 55, 74
Start Date, 58
Start Trigger Automatically on Login, 15, 45, 53
Start Trigger Automatically on Login for Users, 58
Support Debugging Tool Setup, 116
Support Debugging Tool Setup, 50
Support Debugging Tool Setup Options, 58
Syntax Errors, 55
Table, 45, 54
Table Explorer, 53, 57
Table Name, 54
Tables Button, 57
Technical Name, 54
Trigger, 70, 74
Trigger Attach, 52
Trigger Description, 51
Trigger Event, 45, 51, 53, 55
Trigger ID, 45, 46, 48, 50, 51, 58, 80
Trigger Type, 51, 52, 53, 55, 61
Triggering, 48
Unregister, 47, 53
Users Button, 53, 58
Window, 54
Window Name, 54
Automatic Debugger Mode Status, 47, 53
Unregister, 47, 53
Automatic Open Mode, 30, 81
Automatic Start Only, 46
Automatically Install Chunk Files without displaying dialog, 17

B

Bcc Button, 42
Bcc Field, 42
Before Original, 53, 54
Body, 42, 85
Body Text, 42, 85
Bottom Button, 63

C

Cancel Button, 40, 42
Case Sensitive, 23
Cc Button, 42
Cc Field, 42
CDO, 86
Change Window Titles in Windows Start Bar, 83
Check Syntax, 60, 73, 78
Clear Button, 23, 36, 80
Collaboration Data Objects, 86
Comma Delimited, 77
Company, 33
Company Tab, 82
Conditional Script, 45, 48, 54, 55
Configuration Export/Import, 35, 66, 92
Clear Button, 36
Export Button, 35
File Name, 36
Import Button, 35
Import Settings File, 35
Transfer User and Company details with Triggers, 36
Configuration Maintenance, 80, 92
Clear Button, 80
Convert References, 78
Current Path Setting, 89
Customization Status, 62, 64

D

Data Source Name, 18
Database, 76
Debug Menu, 15
Debug Menu Product, 15
Debug Tab, 14
Debugger.cnk, 5
Debugger.log, 11, 49, 116
Debugger.pdf, 5
Debugger.txt, 5
Debugger.xml, 5, 16, 88, 89, 91
Debugger_<User>_<Company>.log, 11, 49, 116
DebugRW, 19, 95
DebugRW.txt, 19, 95
DEFAULT, 46, 50, 80
Default Body Text, 42
Default Body Text for Send Email window, 85
Default Button, 55
DEFAULT only, 46
Default Subject, 42, 84
Delete Button, 63
Delete Record, 52
Developer, 45
Dex.dic, 17, 95
Dex.ini, 14, 39, 88, 91
Dex.ini Configuration, 88, 89, 91, 93

Administrator Setting, 89
Advanced Mode Setting, 89
Current Path Setting, 89
Dex.ini, 88
Do not update any Dex.ini settings automatically, 89
Log, 89
Path Default Setting, 89
Search Mode, 88
Setting or Search String, 88
Settings List, 88
Silent, 89
Value, 89
Dex.ini Settings, 11, 13, 14, 44, 48, 89, 91
Activate Debug Logging for the Report Writer, 19
Allow mismatched Dex.dic version, 17
AllowWrongDex, 17, 95
ApplicationName, 17, 94
AutoInstallChunks, 17, 94
Automatically Install Chunk Files without displaying dialog, 17
Debug Tab, 14
DebugRW, 19, 95
Dexterity Debug Menu Product, 15
Display More Info button on Process Monitor, 21
Enable Debugger Advanced Mode Features, 15, 44, 89
Enable Debugger Setup Mode, 15
Enable Dexterity Debug Menu on next login, 15
Enable Enhanced Script Log on next login, 15
Enable SQL Logging on next login, 14
Export Body Section as One Line, 19
ExportOneLineBody, 19, 95
MBS_Debug_ConfigurationOverride, 89, 91
MBS_Debug_LogOnStartup, 16, 91
MBS_Debug_Mode, 15, 89, 91
MBS_Debug_Path, 16, 89, 91
MBS_Debug_RuntimeCheck, 91
MBS_Debug_SetupMode, 15, 91
MBS_Debug_WinAdminSettings, 93
MBS_Debug_WinConfigSettings, 93
MBS_Debug_WinConfigurationExportImport, 92
MBS_Debug_WinConfigurationMaintenance, 92
MBS_Debug_WinDebuggerSetup, 92
MBS_Debug_WinDictionaryControl, 92
MBS_Debug_WinFieldLookup, 93
MBS_Debug_WinMenuExplorer, 93
MBS_Debug_WinReportExplorer, 93
MBS_Debug_WinResourceExplorer, 93
MBS_Debug_WinResourceInformation, 92
MBS_Debug_WinRuntimeExecute, 92
MBS_Debug_WinScreenShot, 92
MBS_Debug_WinSecurityInfo, 92
MBS_Debug_WinSecurityProfiler, 92
MBS_Debug_WinSendEmail, 93
MBS_Debug_WinSQLExecute, 92
MBS_Debug_WinTableExplorer, 93
MBS_Debug_WinTableLookup, 93
MBS_Debug_WinXMLTableExport, 92
MBS_Debug_WinXMLTableImport, 92
Name shown on Application title bar during initial loading, 17
Other Tab, 20
Pathname location for Debugger Setup files, exports and logs, 11, 16, 48, 89
Pathname location for SQL Log file, 14
QueueMoreInfo, 21, 95
Rename DEXSQL.LOG at the beginning of each day, 15
Reports Tab, 19
Reset Window Positions, 16
SAMPLEDATEMSG, 18, 95
Script Editor Settings, 96
ScriptCommentColor, 96
ScriptDebugger, 15, 94
ScriptDebuggerProduct, 15, 94
ScriptEditorFontName, 96
ScriptEditorFontSize, 96
ScriptEditorSyntaxColoring, 96
ScriptErrorColor, 96
ScriptIdentifierColor, 96
ScriptKeywordColor, 96
ScriptLogEnhanced, 15, 94
ScriptNumberColor, 96
ScriptOperatorColor, 96
ScriptStringColor, 96
Show Advanced Macro Menu, 20
Show All Menu Items, 20
Show Debug Messages on next login, 15
ShowAdvancedMacroMenu, 20, 95
ShowAllMenuItems, 20, 95
ShowDebugMessages, 15, 94
Skip Version Checks for Dictionaries, 17
SkipVersionChecks, 17, 95
SQLLastCompany, 92
SQLLogAllODBCMessages, 94
SQLLoginCompatibilityMode, 18, 95
SQLLogODBCMessages, 14, 94
SQLLogPath, 14, 94
SQLLogRename, 15, 91
SQLLogSQLStmt, 14, 94
Start Logging on next startup only, 11, 16
Startup Tab, 17
Support Debugging Tool Settings, 91
Suppress Date Change Dialog, 20
Suppress Sample Company Date Warning, 18
Suppress Sound from Application, 20
SuppressChangeDateDialog, 20, 95
SuppressSound, 20, 95
System Settings, 94
Use SQL Login Compatibility Mode, 18
DEXSQL.LOG, 9, 11, 14, 94
DEXSQL_<User>_<Company>.LOG, 11
Dexterity, 8, 9, 15, 22, 45, 46, 47, 48, 50, 52, 55, 61, 62, 70, 71, 74, 76, 89
Customization Status, 62
Debug Menu, 15
Debug Menu Product, 15
Developer, 45
Dexterity Script Logging, 9
Dexterity Script Profiling, 9
Dictionary, 22
Display Name, 22, 74
Enable Enhanced Script Log on next login, 15
Field, 22, 45, 54, 74
Form, 22, 54
Function, 54
Physical Name, 22, 74
Procedure, 54
Report, 22
Resource ID, 22
Resources, 22
Sanscript, 45, 55, 70, 71, 76

Show Debug Messages on next login, 15
Table, 22, 45, 54, 74
Table Group, 22
Technical Name, 22, 54
Trigger, 45, 46, 47, 48, 50, 52, 61, 62
Window, 22, 54
Dexterity Debug Menu, 15
Dexterity Debug Menu Product, 15
Dexterity Script Logging, 9
Dexterity Script Profiling, 9
Dialog Message, 54
Dictionary, 22
Dictionary Control, 62, 92
Alternate/Modified Status, 62
Bottom Button, 63
Customization Status, 64
Delete Button, 63
Down Button, 63
Field Level Security, 64
Info Button, 64
Show Launch File, 64
Top Button, 63
Trigger Status, 62
Up Button, 63
Dictionary ID, 53, 55
Disabled, 53
Display dialog on login for this company, 82
Display Message, 48, 54
Display More Info button on Process Monitor, 21
Display Name, 22, 74
Do not activate Logging Mode, 53
Do not update any Dex.ini settings automatically, 89
Down Button, 63
DSN, 18
DUOS, 112, 113, 114, 115
Duplicate Button, 58, 66, 72, 77
Duplicate Records, 69
Dynamics.set, 13, 17, 39, 62, 63, 64

E

eConnect, 61
Email 1 Tab, 84
Email 2 Tab, 86
Email Button, 40
Email Mode, 86
Enable Debugger Advanced Mode Features, 15, 44, 89
Enable Debugger Setup Mode, 15
Enable Dexterity Debug Menu on next login, 15
Enable Enhanced Script Log on next login, 15
Enable SQL Logging on next login, 14
End Date, 58
Every 1 Minute, 52
Every 10 Minutes, 52
Every 15 Minutes, 52
Every 30 Minutes, 52
Every 5 Minutes, 52
Every 60 Minutes, 52
Examples of use, 2
Exception Error Dialog, 72, 77
Execute Button, 72, 77
Execute Change Script, 56, 99
Execute Dexterity SanScript code in the context of Product, 71
Execute Query in which SQL Database, 76
Export Body Section as One Line, 19
Export Button, 29, 35, 66, 77
Export entire Table, 54
Export entire Table when Condition is met, 48
Export Mode, 77
Export Path, 66, 67, 69
Export Table Record, 54
Export Table Record when Condition is met, 48
ExportOneLineBody, 19, 95
Extras Menu, 59, 72, 78

F

Field, 22, 45, 54, 74
Field
Pre, 52
Field
Change, 52
Field
Post, 52
field keyword, 74
Field Level Security, 64
Field Lookup, 93
Field Name, 54
File Name, 36
Filter Tables having field, 23
Find …, 59, 72, 78
Find Next, 59, 72, 78
Focus Event, 51, 52, 53, 54, 55
Font Size, 60, 73, 79, 96
Font Style, 60, 73, 79, 96
Form, 22, 28, 54
Form
Level, 52
Form
Level, 52
Form
Pre, 52
Form
Post, 52
Form Name, 54
From Field, 41
Function, 51, 52, 53, 54
Function Name, 54

G

General Tab, 81
Global
Level, 52
Go To Button, 31, 33

H

Helper Button, 56, 72
Helper Function Assistant, 56, 72, 97, 117, 118, 119, 120, 121, 123, 125
Adds Allowed, 56, 101, 103, 105, 107
Execute Change Script, 56, 99
Key Fields, 56
Runtime Execute, 72, 97
Helper Functions, 55, 56, 72, 97
Adds Allowed, 101, 103, 105, 107
DUOS, 112, 113, 114, 115

Execute Change Script, 99
MBS_Auto_Log, 116, 117
MBS_Get_Table_Value1, 100
MBS_Get_Table_Value2, 102
MBS_Get_Table_Value3, 104
MBS_Get_Table_Value4, 106
MBS_Get_Window_Value, 98
MBS_Param_Del, 114
MBS_Param_DelAll, 115
MBS_Param_Get, 112, 113, 117
MBS_Param_Set, 112, 113, 114, 115, 117
MBS_Runtime_Execute, 108, 110, 112, 113, 114, 115
MBS_Script_Load_Dex, 110
MBS_Script_Load_SQL, 111
MBS_Set_Table_Value1, 101
MBS_Set_Table_Value2, 103
MBS_Set_Table_Value3, 105
MBS_Set_Table_Value4, 107
MBS_Set_Window_Value, 99
MBS_SQL_Check_Exists, 109, 111
SY_User_Object_Store, 112, 113, 114, 115
SY90000, 112, 113, 114, 115
How to Setup, 45
HTML Table, 77

I

Import Button, 29, 35, 68
Import Path, 67, 68, 69
Import Settings File, 35
Include Current Launch File, 39, 81
Include Dex.ini Settings File, 39, 81
Individual Logging Control, 12
Info Button, 40, 64
Insert Button, 55, 71
Insert Helper Function, 56, 72
Adds Allowed, 56
Execute Change Script, 56
Key Fields, 56
Installation, 5
Integration Manager, 61
Introduction, 1, 45

K

Key Fields, 56

L

Launch File, 13, 17, 39, 62, 63, 64
Legend Button, 32, 33
Limit results set to fixed number of lines, 76
List, 77
Log, 89
Log File, 49

M

Manual Logging Mode, 9, 16, 48
MAPI Compliant Client, 86
Mark All Button, 40
MBS_Auto_Log, 116, 117
MBS_Debug_ConfigurationOverride, 89, 91
MBS_Debug_LogOnStartup, 16, 91
MBS_Debug_Mode, 15, 89, 91
MBS_Debug_Path, 16, 89, 91
MBS_Debug_RuntimeCheck, 91
MBS_Debug_SetupMode, 15, 91
MBS_Debug_WinAdminSettings, 93
MBS_Debug_WinConfigSettings, 93
MBS_Debug_WinConfigurationExportImport, 92
MBS_Debug_WinConfigurationMaintenance, 92
MBS_Debug_WinDebuggerSetup, 92
MBS_Debug_WinDictionaryControl, 92
MBS_Debug_WinFieldLookup, 93
MBS_Debug_WinMenuExplorer, 93
MBS_Debug_WinReportExplorer, 93
MBS_Debug_WinResourceExplorer, 93
MBS_Debug_WinResourceInformation, 92
MBS_Debug_WinRuntimeExecute, 92
MBS_Debug_WinScreenShot, 92
MBS_Debug_WinSecurityInfo, 92
MBS_Debug_WinSecurityProfiler, 92
MBS_Debug_WinSendEmail, 93
MBS_Debug_WinSQLExecute, 92
MBS_Debug_WinTableExplorer, 93
MBS_Debug_WinTableLookup, 93
MBS_Debug_WinXMLTableExport, 92
MBS_Debug_WinXMLTableImport, 92
MBS_Get_Table_Value1, 100
MBS_Get_Table_Value2, 102
MBS_Get_Table_Value3, 104
MBS_Get_Table_Value4, 106
MBS_Get_Window_Value, 98
MBS_Param_Del, 114
MBS_Param_DelAll, 115
MBS_Param_Get, 112, 113, 117
MBS_Param_Set, 112, 113, 114, 115, 117
MBS_Runtime_Execute, 108, 110, 112, 113, 114, 115
MBS_Script_Load_Dex, 110
MBS_Script_Load_SQL, 111
MBS_Set_Table_Value1, 101
MBS_Set_Table_Value2, 103
MBS_Set_Table_Value3, 105
MBS_Set_Table_Value4, 107
MBS_Set_Window_Value, 99
MBS_SQL_Check_Exists, 109, 111
Menu Explorer, 25, 93
Message, 55
Microsoft Outlook Client, 86
Modified, 28, 62, 63
Modified Alternate, 28
Modifier, 22

N

Name shown on Application title bar during initial loading, 17
Names Button, 56, 72
Navigation, 5
Options Button, 13, 14, 22, 28, 35, 37, 41, 44, 47, 50, 62, 65, 68, 70, 74, 80, 81, 88
Standard Toolbar, 37, 38
Tools Menu, 6, 37, 38
Non Logging All Except Disabled, 46
Non Logging Automatic Start Only, 46
Non Logging Triggers, 46, 47, 53, 54, 55

O

ODBC, 18
OK Button, 23, 29, 33, 83
Old Field Value, 47
Only include tables which contain data, 24
Open Button, 23, 29
Open Database Connectivity, 18
Open Windows, 39
Options, 60, 73, 79
Options Button, 13, 14, 22, 28, 35, 37, 41, 44, 47, 50, 58, 62, 65, 68, 70, 74, 80, 81, 88
Other Tab, 20
OUT_Condition, 55
Outlook, 86
Overwrite Duplicate Records, 69
Overwrite Table Contents, 69

P

Password, 87
Path Default Setting, 89
Pathname location for Debugger Setup files, exports and logs, 11, 16, 48, 89
Pathname location for SQL Log file, 14
Perform actions when conditions not met, 54
Physical Name, 22, 74
Preview, 42, 86
Print Button, 29
Procedure, 51, 52, 53, 54
Procedure Name, 54
Process Monitor, 21, 95
Product ID, 53
Product Name, 53, 71
Profile ID, 65, 66
Profile Name, 66
Profile.txt, 9, 11
Profile_<User>_<Company>_<Date>_<Time>.txt, 11
Progress Window, 67, 69

Q

Query Analyzer, 61, 74
QueueMoreInfo, 21, 95

R

Read Record, 52
Record.xml, 48
Record_<User>_<Company>_<Date>_<Time>.xml, 48
Redisplay Button, 33
Refresh Button, 39, 40
Registration, 46
Reject Record, 55
Reject Script, 54
Remove Attachment Button, 42
Remove Button, 42
Rename DEXSQL.LOG at the beginning of each day, 15
Replace …, 59, 73, 78
Replace and Find Next, 60, 73, 78
Report, 22, 28
Report Explorer, 26, 93
Report Writer, 22, 117, 118, 119, 120, 121, 122, 124
Report Writer Functions, 72, 97, 117
Reports Tab, 19
Reset Buttons, 83
Reset Window Positions, 16
Resource Explorer, 25, 53, 56, 72, 93
Resource ID, 22
Resource Information, 22, 92
  Associated Tables Button, 23
  Case Sensitive, 23
  Clear Button, 23
  Dexterity, 22
  Dictionary, 22
  Display Name, 22
  Field, 22
  Form, 22
  Menu Explorer, 25
  OK Button, 23
  Open Button, 23
  Physical Name, 22
  Report, 22
  Report Explorer, 26
  Resource Explorer, 25
  Resource ID, 22
  Resource Type, 23
  Right click enabled, 27, 31
  Search Again Button, 23
  Search Mode, 23
  Security Button, 23, 31
  Select Associated Table, 23
  Select Table Containing Field, 24
  Table, 22
  Table Explorer, 26
  Table Group, 22
  Tables Containing Field Button, 24
  Technical Name, 22
  Window, 22
Resource Type, 23
Restriction of Scope, 61
  ActiveX Data Objects, 61
  ADO, 61
  eConnect, 61
  Integration Manager, 61
  Query Analyzer, 61
  VBA, 61
  Visual Basic for Applications, 61
Right click enabled, 27, 30, 31
Runtime Execute, 70, 72, 74, 92, 97, 110, 117, 118, 119, 120, 121, 122, 124
  Check Syntax, 73
  Duplicate Button, 72
  Exception Error Dialog, 72
  Execute Button, 72
  Execute Dexterity SanScript code in the context of Product, 71
  Find …, 72
  Find Next, 72
  Font Size, 73
  Font Style, 73
  Helper Button, 72
  Helper Function Assistant, 72
  Insert Button, 71
  Insert Helper Function, 72
  Names Button, 72
  Options, 73
  Product Name, 71
  Replace …, 73

Replace and Find Next, 73
Resource Explorer, 72
Script, 71
Script ID, 71, 72, 110
Script Name, 71
SQL Execute, 70, 74
Syntax Errors, 71
Table Explorer, 72
Tables Button, 72
Transact SQL, 71
RW Functions, 72, 97, 117
Runtime Execute, 117, 118, 119, 120, 121, 122, 124
rw_ReportEnd, 119
rw_ReportStart, 118
rw_TableHeaderCurrency, 121
rw_TableHeaderString, 120
rw_TableLineCurrency, 124
rw_TableLineString, 122
rw_ReportEnd, 119
rw_ReportStart, 118
rw_TableHeaderCurrency, 121
rw_TableHeaderString, 120
rw_TableLineCurrency, 124
rw_TableLineString, 122

S

SAMPLEDATEMSG, 18, 95
Sanscript, 45, 55, 70, 71, 76
Save Button, 40
Save Path, 39, 40
Save Record, 52
ScreenShot, 37, 39, 81, 92
Cancel Button, 40
Email Button, 40
Include Current Launch File, 39, 81
Include Dex.ini Settings File, 39, 81
Info Button, 40
Mark All Button, 40
Open Windows, 39
Refresh Button, 39, 40
Save Button, 40
Save Path, 39, 40
System Status, 37, 40
Unmark All Button, 40
Script, 71, 76
Script Context, 55
Script Context ID, 55
Script Editor Settings, 96
Script ID, 71, 72, 76, 77, 110, 111
Script Menu, 59, 72, 78
Check Syntax, 60, 73, 78
Convert References, 78
Find …, 59, 72, 78
Find Next, 59, 72, 78
Font Size, 60, 73, 79, 96
Font Style, 60, 73, 79, 96
Options, 60, 73, 79
Replace …, 59, 73, 78
Replace and Find Next, 60, 73, 78
Syntax Highlighting, 96
Script Name, 71, 76
Script.log, 9, 11
Script_<User>_<Company>_<Date>_<Time>.log, 11
ScriptCommentColor, 96
ScriptDebugger, 15, 94
ScriptDebuggerProduct, 15, 94
ScriptEditorFontName, 96
ScriptEditorFontSize, 96
ScriptEditorSyntaxColoring, 96
ScriptErrorColor, 96
ScriptIdentifierColor, 96
ScriptKeywordColor, 96
ScriptLogEnhanced, 15, 94
ScriptNumberColor, 96
ScriptOperatorColor, 96
ScriptStringColor, 96
Scroll
Change, 52
Delete, 52
Fill, 52, 55
Insert, 52
Post, 52
Pre, 52
Scrolling Window Toolbar Color, 83
Search Again Button, 23
Search Mode, 23
Security, 5, 6, 7, 31, 33
Advanced Security, 6, 31
Alternate/Modified Forms and Reports, 31, 33
Security Role Setup, 31, 33
Security Roles, 7
Security Task Setup, 31, 33
Standard Security, 6, 31
User Class Setup, 31
User Security Setup, 7, 31, 33
Security Button, 23, 29, 31
Security Information, 23, 29, 31, 92
Company, 33
Go To Button, 31, 33
Legend Button, 32, 33
OK Button, 33
Redisplay Button, 33
Security Information Legend, 33
User ID, 33
Security Information Legend, 33
Security Privileges, 28
Security Profiler, 28, 30, 81, 92
Access Denied, 28
Alternate, 28
Application Level Security, 28, 29
Automatic Open Mode, 30, 81
Export Button, 29
Form, 28
Import Button, 29
Modified, 28
Modified Alternate, 28
OK Button, 29
Open Button, 29
Print Button, 29
Report, 28
Right click enabled, 30, 31
Security Button, 29, 31
Security Privileges, 28
Security Profiler Log, 29
SQL Server Security, 29
Table, 28
Windows Level Security, 29
Security Profiler Log, 29
Security Role Setup, 31, 33

Security Roles, 7
Security Task Setup, 31, 33
Select Accociated Table
  Filter Tables having field, 23
Select Associated Table, 23
Select Buttons, 83
Select Table Containing Field, 24
  Only include tables which contain data, 24
Selected Users and Companies, 59
Send Button, 42, 86
Send Email, 41, 54, 86, 93
  Add Attachment Button, 42
  Add Button, 42
  Administrator Email, 42
  Attachments, 42
  Bcc Button, 42
  Bcc Field, 42
  Body, 42, 85
  Body Text, 42, 85
  Cancel Button, 42
  Cc Button, 42
  Cc Field, 42
  Default Body Text, 42
  Default Subject, 42
  From Field, 41
  Remove Attachment Button, 42
  Remove Button, 42
  Send Button, 42, 86
  Sender's Email, 41
  Subject, 42, 84
  To Button, 42
  To Field, 42
Send HTML, 86
Sender's Email, 41, 87
Setting or Search String, 88
Settings List, 88
Setup, 50
Setup Mode, 15, 91
Show Advanced Macro Menu, 20
Show All Menu Items, 20
Show Debug Messages on next login, 15
Show Dexterity Technical Name Syntax Button, 77
show keyword, 74
Show Launch File, 64
ShowAdvancedMacroMenu, 20, 95
ShowAllMenuItems, 20, 95
ShowDebugMessages, 15, 94
Silent, 89
Skip Version Checks for Dictionaries, 17
SkipVersionChecks, 17, 95
SMTP Server, 87
SMTP Server Port, 87
SMTP Server via CDO, 86
SQL Database, 76
SQL Execute, 74, 92, 111
  alias keyword, 74
  Check Syntax, 78
  Comma Delimited, 77
  Convert References, 78
  Database, 76
  Display Name, 74
  Duplicate Button, 77
  Exception Error Dialog, 77
  Execute Button, 77
  Execute Query in which SQL Database, 76
  Export Button, 77
  Export Mode, 77
  field keyword, 74
  Find …, 78
  Find Next, 78
  Font Size, 79
  Font Style, 79
  HTML Table, 77
  Limit results set to fixed number of lines, 76
  List, 77
  Options, 79
  Physical Name, 74
  Query Analyzer, 74
  Replace …, 78
  Replace and Find Next, 78
  Script, 76
  Script ID, 76, 77, 111
  Script Name, 76
  Show Dexterity Technical Name Syntax Button, 77
  show keyword, 74
  SQL Database, 76
  Tab Delimited, 77
  Table Explorer, 77
  Tables Button, 77
  Text, 77
  Transact SQL, 74, 76
SQL Logging, 9
SQL Native Client, 18
SQL Server, 9, 22, 44
  SQL Logging, 9
SQL Server Security, 29
SQLLastCompany, 92
SQLLogAllODBCMessages, 94
SQLLoginCompatibilityMode, 18, 95
SQLLogODBCMessages, 14, 94
SQLLogPath, 14, 94
SQLLogRename, 15, 91
SQLLogSQLStmt, 14, 94
Standard mode, 91
Standard Mode, 1, 4, 7, 9, 11, 12, 13, 14, 22, 28, 31, 35, 37, 41, 48, 66, 89
  About Support Debugging Tool, 13
  Configuration Export/Import, 35, 66
  Dex.ini Settings, 11, 14, 48, 89
  Individual Logging Control, 12
  Manual Logging Mode, 9
  Resource Information, 22
  ScreenShot, 37
  Security Information, 23, 29, 31
  Security Profiler, 28
  Send Email, 41
Standard Security, 6, 31
Standard Signature to add to all emails, 85
Standard Toolbar, 37, 38
Start Date, 58
Start Logging on next startup only, 11, 16
Start Trigger Automatically on Login, 15, 45, 53
Start Trigger Automatically on Login for Users, 58
Startup Tab, 17
Subject, 42, 84
Support, 3
Support Debugging Tool Settings, 91
Support Debugging Tool Setup, 50, 92, 116
Support Debugging Tool Setup Options, 58
Suppress Date Change Dialog, 20

Suppress Sample Company Date Warning, 18
Suppress Sound from Application, 20
SuppressChangeDateDialog, 20, 95
SuppressSound, 20, 95
SY_User_Object_Store, 112, 113, 114, 115
SY90000, 112, 113, 114, 115
Syntax Errors, 55, 71
Syntax Highlighting, 96
System Password, 44
System Settings, 94
System Status, 37, 40

T

Tab Delimited, 77
Table, 22, 28, 45, 51, 52, 53, 54, 74
Table Explorer, 26, 53, 57, 72, 77, 93
Table Group, 22
Table List, 66
Table Lookup, 93
Table Name, 54
Table Physical Name, 66
Table Technical Name, 66
Table.xml, 48
Table_<User>_<Company>_<Date>_<Time>.xml, 48
Tables Button, 57, 72, 77
Tables Containing Field Button, 24
Technical Name, 22, 54
Terminal Server, 86
Text, 77
Third Party Dictionary, 62
Timed Event, 51, 52, 53
To Button, 42
To Field, 42
Tools Menu, 6, 37, 38
Top Button, 63
Transact SQL, 71, 74, 76
Transfer User and Company details with Triggers, 36
Trigger, 45, 46, 47, 48, 50, 52, 61, 62
Trigger Attach, 52
  After Original, 53
  After Table Event, 53
  After Timed Event, 53
  Before Original, 53, 54
Trigger Description, 51
Trigger Event, 45, 51, 53, 55
  Delete Record, 52
  Every 1 Minute, 52
  Every 10 Minutes, 52
  Every 15 Minutes, 52
  Every 30 Minutes, 52
  Every 5 Minutes, 52
  Every 60 Minutes, 52
  Field Change, 52
  Field Post, 52
  Field Pre, 52
  Form Level, 52
  Form Post, 52
  Form Pre, 52
  Global Level, 52
  Read Record, 52
  Save Record, 52
  Scroll Change, 52
  Scroll Delete, 52
  Scroll Fill, 52, 55
  Scroll Insert, 52
  Scroll Post, 52
  Scroll Pre, 52
  Warning Dialog, 52
  Window Activate, 52
  Window Post, 52
  Window Pre, 52
Trigger ID, 45, 46, 48, 50, 51, 58, 80
Trigger Status, 62
Trigger Type, 51, 52, 53, 55, 61
  Focus Event, 51, 52, 53, 54, 55
  Function, 51, 52, 53
  Procedure, 51, 52, 53
  Table, 51, 52, 53
  Timed Event, 51, 52, 53
  Warning Dialog, 51, 52, 53
Triggering, 48

U

Un-install, 13
Unmark All Button, 40
Unregister, 47, 53
Up Button, 63
Use SQL Login Compatibility Mode, 18
User Class Setup, 31
User ID, 33, 87
User Security Setup, 7, 31, 33
Users Button, 53, 58

V

Value, 89
VBA, 9, 61
Visual Basic for Applications, 9, 61

W

Warning Dialog, 51, 52, 53
Window, 22, 54
Window
  Pre, 52
Window
  Post, 52
Window
  Activate, 52
Window Background Color, 83
Window Name, 54
Windows Level Security, 29
Windows Start Bar, 83

X

XML Table Export, 65, 92
  Duplicate Button, 66
  Export Path, 66, 67, 69
  Profile ID, 65, 66
  Profile Name, 66
  Progress Window, 67, 69
  Table List, 66
  Table Physical Name, 66
  Table Technical Name, 66
XML Table Import, 68, 92
  Duplicate Records, 69

Import Button, 68
Import Path, 67, 68, 69
Overwrite Duplicate Records, 69
Overwrite Tables Contents, 69
Progress Window, 67, 69

** End of document - Debugger.doc - DM - 9 March 2010 **