Information about how to handle literal string warnings by using messages in Dexterity in Microsoft Dynamics GP

CONFIDENTIAL ARTICLE

(The information in this article is provided to you in accordance with your Confidentiality Agreement)

Partner Only Article
Article ID : 943178
Last Review : N/A
Revision : 1.2

INTRODUCTION

Dexterity is designed to enable you to create applications that can be easily translated and customized without requiring changes or access to actual source code. To enable this flexibility, you must make sure that any text that a user might see is not hard-coded into scripts.

MORE INFORMATION

Because a hard-coded string in a script cannot be changed without altering the source code itself, Dexterity generates a "Use of literal string" compiler warning message when a hard-coded string is used in a script. Dexterity enables you to create a separate message resource that you can use instead of a hard-coded string or a literal string. You can find messages under **Base resources** in the Dexterity Resource Explorer window.

When you create a new message, you must provide a message ID. Then, you must provide the message text. To make sure that the messages are extracted when a distribution chunk file is created, the message ID must be 22,000 or larger for a third-party product.

You can retrieve messages in the code by using the **getmsg()** function. You can also embed placeholders in a message. You can use the placeholders to substitute strings that are based on data or on calculated values. These placeholders or replacement markers are shown in the form of %1, %2, %3, and so on. When you use the placeholders or the replacement markers, you can use the substitute command to replace the placeholders with one or more values.

You can also use a special token together with messages to automatically show a product name that is based on the product ID that appears in the Dynamics.set launch file. If you include the "@PROD<ProdID>@" token in a message, the name of the product appears in the message. For example, the following code automatically uses this kind of token:

```
local string l_message;

l_message = getmsg(22000) {@PROD0@ cannot find customer %1.};

substitute l_message, 'Customer Number';

warning l_message;
```

For the first customer in the lesson company, this code causes the following message to appear:

> Microsoft Dynamics GP cannot find customer AARONFIT0001.

When you write code that contains pass-through SQL code, runtime-compiled Dexterity SanScript code, cross-dictionary trigger registrations, runtime-created macro files, or warning messages for the developer, you can use hardcoded strings without receiving compiler warnings about literal strings. In this situation, these hardcoded strings are not translated. The end user does not see these strings.

In this situation, you can use a special pragma pre-compiler directive to turn the warnings on and off. To do this, use the following lines before and after the acceptable literal strings in your code:

```
          pragma(disable warning LiteralStringUsed);
{turn off compiler generated string warnings}

          if Trigger_RegisterProcedure(script 'Add_Successful_Login_Record', TRIGGER_AFTER_ORIGINAL, script Set_Environment) <> SY_NOERR then
                    warning "Log on procedure trigger registration failed.";
          end if;
          pragma(enable warning LiteralStringUsed);
{turn on compiler generated string warnings}
```

Note If you do not enable the warnings before the end of the script, the compiler displays the following message:

> Notice: Warning 'LiteralStringUsed' was disabled without being re-enabled.

If you have other strings that are not visible to the user and that do not require translation, we recommend that you add these strings as constants at the form level or at the global level. Then, you can use the constant in the script without generating a warning.

A final option is to use a Dex.ini setting to disable the warnings for all scripts. Although this option prevents warnings about literal strings, the option does not encourage others to handle literal strings in scripts according to established best practices. We recommend that you use the other techniques that this article describes instead of using this setting. The following is the setting:

```
CompilerWarningLevel=2
```

REFERENCES

For more information about the **getmsg()** command and the **substitute** command in the Dexterity SanScript reference, see Chapter 18: Messages in Volume 1 of the Dexterity Programmer's Guide.

For more information about how to use Dexterity to write international and multilingual code in Microsoft Dynamics GP, click the following article number 942749 to view the article 942749 in the Microsoft Knowledge Base:

> https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;942749

For more information about the Pragma Option suppress or to turn off the Dexterity Compiler String Warning Messages, click the following article number 850877 to view the article 850877 in the Microsoft Knowledge Base:

> https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;850877

APPLIES TO

- Dexterity, when used with:
  Microsoft Dynamics GP 10.0
  Microsoft Dynamics GP 9.0
  Microsoft Business Solutions–Great Plains 8.0

Keywords: kbexpertiseadvanced kbmbspartner kbexpertiseinter kbhowto kbinfo kbmbsmigrate kbexpertisebeginner KB943178