

# G2-Database Bridge

User's Guide

Version 12.0 Rev. 1



G2-Database Bridge User's Guide, Version 12.0 Rev. 1  
June 2020

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Copyright © 1985-2020 Gensym Corporation

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Gensym Corporation.

Gensym®, G2®, Optegrity®, and ReThink® are registered trademarks of Gensym Corporation.

NeurOn-Line™, Dynamic Scheduling™, G2 Real-Time Expert System™, G2 ActiveXLink™, G2 BeanBuilder™, G2 CORBALink™, G2 Diagnostic Assistant™, G2 Gateway™, G2 GUIDE™, G2GL™, G2 JavaLink™, G2 ProTools™, GDA™, GFI™, GSI™, ICP™, Integrity™, and SymCure™ are trademarks of Gensym Corporation.

Telewindows is a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries. Telewindows is used by Gensym Corporation under license from owner.

This software is based in part on the work of the Independent JPEG Group.

Copyright © 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

License for Scintilla and SciTE, Copyright 1998-2003 by Neil Hodgson, All Rights Reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Gensym Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Ignite Technologies, Inc.  
401 Congress Ave., Suite 2650  
Austin, TX 78701 USA  
Telephone: +1-800-248-0027  
Email: [success@ignitetech.com](mailto:success@ignitetech.com)

Part Number: DOC047-1200

# Contents

---

	<b>Preface</b>	<b>ix</b>
	About this Guide	ix
	Version Information	ix
	Audience	ix
	A Note About the API	x
	A Note About Release Notes	x
	Conventions	x
	Related Documentation	xii
	Customer Support Services	xiv
		xv
<b>Chapter 1</b>	<b>Introduction</b>	<b>1</b>
	Introduction	1
	Capabilities of G2-Database Bridges	2
	Performing Database Operations	2
	Event Messages and Message Handling	2
	Preparing Your Application to Use a G2-Database Bridge	3
	Getting Started	3
<b>Chapter 2</b>	<b>Preparing Your KB for Using the Bridge</b>	<b>5</b>
	Introduction	5
	Installing the G2-Database Bridge	6
	Merging the Database Bridge KB into Your G2 Application	6
	Updating Your G2 Application with a New Database Bridge	6
	Updating a Modularized Knowledge Base	7
	Resolving KB Conflicts	7
<b>Chapter 3</b>	<b>Using the G2-Database Bridge Workspaces</b>	<b>9</b>
	User Modes	9

Keyboard Shortcuts	10
Standard Workspaces of G2-Database Bridges	10
G2-Database Connection Configuration Workspace	12
G2-Database SQL Object Classes Workspace	13
G2-Database Procedures Workspace	15

## **Chapter 4   Running the Bridge   17**

Introduction	17
Command-Line Options	18
Initial Bridge Memory Requirements	20
Text Conversion Styles	20
Starting the Bridge Process	22
On UNIX Systems	22
On Windows Systems	22
Bridge Process Output	23
Establishing a Connection between the Bridge and G2	24
Running a Bridge with Multiple Connections to G2	25
Running Multiple Copies of a Bridge	25

## **Chapter 5   Configuring Connections   27**

Introduction	27
Creating G2-Database-Interface-Objects	28
Attributes of G2-Database-Interface Objects	28
Using the Set Null Attributes	40
Inserting Values into a Table	40
The Set-Null Attributes of g2-database-interface	41
Using db-execute-immediate to Insert Null	41
Using db-exec-sql to Insert Null	41
Using db-exec-sql-obj to Insert Null	43
Sending Connection Configuration Information to the Bridge	44
Resetting the Interface Connection	45
Displaying the Connection Status	45
Changing Icon Colors	47

## **Chapter 6   DML Database Operations   49**

Introduction	49
--------------	----

	Components of a DML Database Operation	50
	Bind Variables in SQL Statements	50
	Procedures for DML Database Operations	51
	Database Operations Using Simple Values	52
	Database Operations Using Objects	53
	Database Operations without Bind Variables	54
<b>Chapter 7</b>	<b>Querying the Database</b>	<b>57</b>
	Introduction	57
	Bind Variables in Database Queries	58
	Returning Query Data to G2	58
	Creating a Cursor Object	59
	Returning Query Data in Query Items	60
	Returning Query Data to Existing G2 Items	61
	Returning Query Data to a User-Defined Object	61
	Returning Query Data to Query Objects	62
	Returning Query Data in Structures	62
	Copying Query Item Attribute Values	62
	Deleting Query Items	63
	Using Smart Fetch	63
<b>Chapter 8</b>	<b>Query Objects</b>	<b>65</b>
	Introduction	65
	Creating a Query Object Class Definition	66
	Specifying Direct Superior Classes	68
	Specifying Column Attributes	69
	Defining a Simple Attribute	69
	Defining Attributes as Parameters	70
	Defining an Attribute as a List	70
	Creating a Query Object	71
	Creating a Cursor Object	75
	Performing the Query	75
<b>Chapter 9</b>	<b>Bridge Procedures</b>	<b>77</b>
	Introduction	78

Summary of G2-Database Bridge Procedures	<b>78</b>
Connection and Initialization	<b>79</b>
SQL Operations	<b>79</b>
Query Operations	<b>81</b>
Error and Message Handling	<b>82</b>
Methods and Utilities	<b>82</b>
Invoking G2-Database Bridge Procedures	<b>83</b>
Invoking a Bridge Procedure from within a G2 Procedure	<b>83</b>
Status Values Returned by G2-Database Bridge Procedures	<b>84</b>
Omitting Return Arguments from Calls to G2-Database Bridge Procedures	<b>84</b>
Invoking a Bridge Procedure from a Rule, Action-Button, or User-Menu-Choice	<b>85</b>
Procedure Descriptions	<b>86</b>
db-commit	<b>87</b>
db-configuration	<b>89</b>
db-connect	<b>91</b>
db-context-event-msg	<b>93</b>
db-define-cursor	<b>95</b>
db-define-sql	<b>99</b>
db-define-sql-obj	<b>102</b>
db-disable-all-triggers	<b>104</b>
db-disconnect	<b>105</b>
db-exec-sql	<b>107</b>
db-exec-sql-obj	<b>109</b>
db-exec-stored-proc	<b>111</b>
db-exec-stored-proc-return	<b>113</b>
db-execute-immediate	<b>117</b>
db-fetch-object	<b>119</b>
db-fetch-query-item	<b>124</b>
db-fetch-records	<b>128</b>
db-fetch-structure	<b>131</b>
db-get-triggers	<b>134</b>
db-io-status	<b>136</b>
db-kill-bridge	<b>137</b>
db-logfile	<b>138</b>
db-ping	<b>141</b>
db-redirect-callback	<b>142</b>
db-refresh-cursor	<b>145</b>
db-rollback	<b>147</b>
db-set-connection-status	<b>149</b>
db-set-cursor	<b>150</b>
db-set-sql	<b>152</b>
db-set-sql-obj	<b>154</b>
db-set-trigger	<b>157</b>
db-sql-function	<b>159</b>

- db-startup 162
- db-text-to-text-list 163
- db-trigger-event 164
- db-update-object 166
- db-update-query-item 170

## **Chapter 10 Message Handling 173**

- Introduction 173
- Handling Messages 174
  - Enabling and Disabling Message Reporting 174
  - Editing Messages 175
- Trigger Events 176
  - Returning Trigger Messages to a G2 Procedure 177
  - Returning Trigger Messages to a Trigger Object 177
- Redirecting Messages 178
- Saving Messages in Log Files 178
  - Opening and Closing Log Files 178
  - Accessing the Log File 179
  - Filtering Log File Entries 179

## **Chapter 11 Troubleshooting 181**

- Introduction 181
- You Cannot Make Connections 182
- Query Does Not Return Expected Values 183
- Deadlocks - Hung or Not Responding Bridge 185
- Other Unexpected Behaviors 187
- Debugging Facility 187

## **Chapter 12 Performance 189**

- Introduction 189
- Complicated Queries 190
- Data Service Priority 190
- Distribution of Bridge Load 190
- Network Considerations 191
- Object Passing 191
- Bind Variables 191

**Appendix A Bridge Data Types 193**

Introduction 193

**Appendix B Status Values 197**

Introduction 197

**Glossary 201**

**Index 207**

# Preface

---

*Describes this guide and the conventions that it uses.*

About this Guide	ix
Version Information	ix
Audience	ix
A Note About the API	x
A Note About Release Notes	x
Conventions	x
Related Documentation	xii
Customer Support Services	xiv



## About this Guide

This guide explains how to install and use your G2-Database bridge.

Release Notes provided with each G2-Database bridge contain bridge-specific information that you need to supplement this guide, as well as information specific to each particular release.

## Version Information

The G2-Database Bridge is compatible with G2 8.0 Rev. 0 or later.

## Audience

To use this guide, you must have at least a limited knowledge of G2, and a thorough understanding of the database system that you want to connect to G2.

## A Note About the API

The G2-Database bridge API is not expected to change significantly in future releases, but exceptions may occur. A detailed description of any changes will accompany the G2-Database bridge release that includes them.

The techniques by which G2-Database bridges implement their capabilities, however, are subject to change at any time without notice or explanation, and are expected to change as the family of G2-Database bridges evolves. These techniques will not be described in any G2-Database bridge documentation.

Therefore, it is essential that you use your G2-Database bridge exclusively through its API, as described in this *G2 Database Bridge User's Guide*. If you bypass the API, you cannot rely on your code to work in the future, since the G2-Database bridge may change, or in the present, because the code may not correctly manage the internal operations of the G2-Database bridge.

## A Note About Release Notes

Release notes contain important information about new features, bug fixes, and anomalies. In addition, the release notes contain detailed, database specific information about establishing connections to the database and communicating with the database. The release notes supplement this user's guide and provide database specific information that is not included in this document.

## Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

### Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, "Burlington, MA"	G2 attribute values and values specified or viewed through dialogs

<b>Convention Examples</b>	<b>Description</b>
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
<i>new-argument</i>	User-specified values in syntax descriptions
<i>text-string</i>	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save Properties	GMS and native menu choices
<b>workspace</b>	Glossary terms
c:\Program Files\Gensym\	Windows pathnames
/usr/gensym/g2/kbs	UNIX pathnames
spreadsh.kb	File names
g2 -kb top.kb	Operating system commands
public void main() gsi_start	Java, C and all other external code

---

**Note** Syntax conventions are fully described in the *G2 Reference Manual*.

---

## Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure underlined. Each value is followed by its type:

```
g2-clone-and-transfer-objects
  (list: class item-list, to-workspace: class kb-workspace,
   delta-x: integer, delta-y: integer)
  -> transferred-items: g2-list
```

## Related Documentation

### G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

### G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*

- *G2 GUIDE User's Guide*
- *G2 GUIDE/UII Procedures Reference Manual*

### **G2 Developers' Utilities**

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

### **Bridges and External Systems**

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2 PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*

- *G2 CORBALink User's Guide*
- *G2 WebLink User's Guide*

### **G2 JavaLink**

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

### **G2 Diagnostic Assistant**

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

## **Customer Support Services**

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone and by email.

### **To obtain customer support online:**

➔ Access Ignite Support Portal at <https://support.ignitetechnology.com>.

You will be asked to log in to an existing account or create a new account if necessary. Ignite Support Portal allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

### **To obtain customer support by telephone or email:**

➔ Use the following numbers and addresses:

**United States Toll-Free** +1-855-453-8174

**United States Toll** +1-512-861-2859

**Email** [support@ignitetechnology.com](mailto:support@ignitetechnology.com)





# Introduction

---

*Describes the capabilities and important features of the G2-Database Bridge products.*

Introduction 1

Capabilities of G2-Database Bridges 2

Preparing Your Application to Use a G2-Database Bridge 3

Getting Started 3

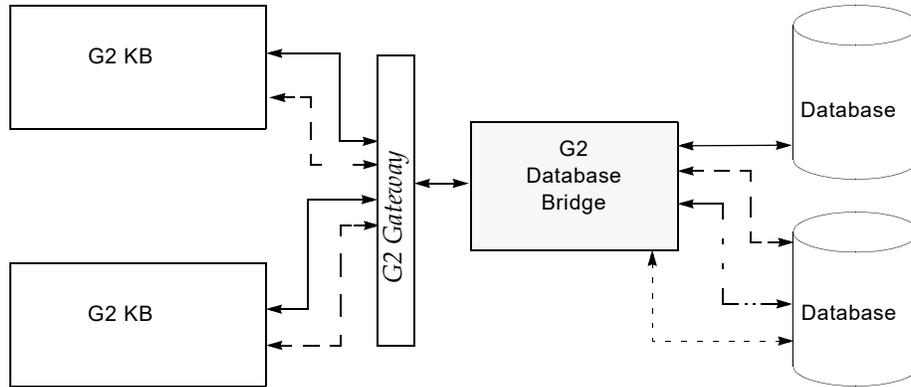


## Introduction

Gensym provides a family of G2-Database bridges that enable your G2 knowledge bases to communicate with a variety of different databases:

- Some G2-Database bridges are designed for use with a particular database, such as Sybase and Oracle.
- The Open Database Connectivity Bridge can communicate with any relational database for which there is an Open Database Connectivity (ODBC) driver.

A G2-Database bridge can run with connections to more than one G2 knowledge base (KB) and database. The following figure illustrates a G2-Database bridge with several connections to G2 KBs and databases:



## Capabilities of G2-Database Bridges

G2-Database bridges provide procedures that support standard database operations. They return messages that report errors and other events occurring during the operation of the bridge.

### Performing Database Operations

From within G2, you can use familiar SQL Data Definition Language (DDL) and SQL Data Manipulation Language (DML) statements to insert, delete, modify, and query data. All G2-Database bridges are fully compatible with the ANSI Structured Query Language (SQL) standards.

You can use the intelligent reasoning capability of G2 to update data within your G2 knowledge base before you change the corresponding data in your database.

### Event Messages and Message Handling

All G2-Database bridges provide a procedure, `db-context-event-msg`, that you can modify to specify how G2 handles messages reported by the bridge. In addition, you can instruct the bridge to return messages to *any* G2 procedure that conforms to the procedure signature of `db-context-event-msg`.

The bridge returns the completion status of database operations. It can also return the number of rows processed by an operation, enabling you to verify that the operation completed as you expected.

# Preparing Your Application to Use a G2-Database Bridge

All G2-Database bridges provide resources for configuring the bridge and monitoring its operation.

These resources are provided in the bridge knowledge base `g2-database.kb`, which is shipped with the G2-Database bridge software. This knowledge base includes:

- A predefined class for configuring connections. You create `g2-database-interface` objects as instances of this class and then set attributes of each G2-Database interface object to configure one connection between the G2 knowledge base and the G2-Database bridge.
- Local G2 procedures and remote procedure definitions. Your G2 knowledge base uses the local procedures as an API to the bridge, to fetch, insert, delete, or modify data in your database.
- Methods for manipulating data and data objects.

You access these resources from the `g2-database` workspace of the `g2-database` module.

You can merge the G2-Database bridge knowledge base, `g2-database.kb`, into an existing modularized G2 application, include it as a module of that application, or use it as the foundation for building a new knowledge base. For information about how to manage your G2-Database bridge knowledge base, see [Preparing Your KB for Using the Bridge](#), and the Release Notes for your particular bridge.

---

**Note** We recommend that you do not directly edit any components of the `g2-database.kb`. Doing so may interfere with future upgrades.

---

## Getting Started

**To enable data service between G2 and your database:**

- 1 Install the G2-Database bridge as part of your G2 installation.

---

**Note** The G2 Database Bridge requires `sys-mod.kb` and `uilroot.kb`.

---

- 2 Merge the bridge knowledge base into your G2 application.
- 3 Create one or more G2-Database interface objects.
- 4 Set the attributes of the G2-Database interface objects.

**5** Start the bridge.

**6** Start or reset G2.

This guide is arranged to lead you through these steps. To begin, see [Preparing Your KB for Using the Bridge.](#)

# Preparing Your KB for Using the Bridge

---

*Describes how to merge the G2-Database Bridge into your application.*

Introduction 5

Installing the G2-Database Bridge 6

Merging the Database Bridge KB into Your G2 Application 6

Updating Your G2 Application with a New Database Bridge 6



## Introduction

Before you install a G2-Database bridge:

- Make sure that there is either a terminal connected to a console port or a remote window available through the network.
- Determine that there is a device capable of reading your distribution medium, then load the medium onto the device.
- Make sure that your database environment has been properly configured and is available to the bridge.

---

**Note** The environment includes environment variables, path, configuration files, and so on.

---

- Read the Release Notes for the G2-Database bridge that you will use.

# Installing the G2-Database Bridge

You install the G2-Database bridge as part of the G2 Bundle installation. Your G2 Bundle license allows you to install two of the available bridges. You can install these bridges on any licensed machine. The installation process takes care of the authorization for you.

## Merging the Database Bridge KB into Your G2 Application

**To merge a database bridge into a G2 application for the first time:**

- 1 Choose Main Menu > Merge KB.
- 2 Navigate to the specific version of the `g2-database.kb` file to merge, depending on the database bridge you have installed.

The database bridge directories are:

- `odbc`
- `oracle`
- `sybase`

- 3 Choose the merge option, automatically resolve all conflicts.

---

**Note** The G2 Database Bridge requires `sys-mod.kb` and `uilroot.kb`.

---

- 4 Choose Save KB to save the resulting knowledge base.

## Updating Your G2 Application with a New Database Bridge

If your G2 application already uses a G2-Database bridge and you want to update it with a newer version of the bridge, you can load your new `g2-database.kb` into a modularized knowledge base, or merge it into an unmodularized knowledge base.

For information on updating your G2 application with a particular G2-Database bridge, see the Release Notes for that bridge.

## Updating a Modularized Knowledge Base

It is recommended practice to modularize any G2 application that uses a G2-Database bridge or bridge knowledge base.

### To modularize an existing KB before merging the database bridge KB:

- 1 Create a module named `g2-database`.  
To do this, choose Main Menu > Miscellany > Create New Module, then specify `g2-database` as the new module.
- 2 Specify a top-level module for the knowledge base.  
To do this, choose Main Menu > System Tables > Module Information and specify a name for the top-level module of your knowledge base.
- 3 Also, specify `g2-database` as a directly required module of the top-level module of the knowledge base.
- 4 Save your knowledge base.

You can then reload your knowledge base into G2. G2 automatically merges the module `g2-database` into the modularized knowledge base.

Each time you load the modularized knowledge base, the `g2-database` module is also loaded as a required module of that knowledge base. For information about knowledge base modules, see the *G2 Reference Manual*.

## Resolving KB Conflicts

When you load a knowledge base that may have definitions that conflict with a previous version of the knowledge base, it can be helpful to specify the load option `automatically resolve conflicts`. If you specify this option, G2 upgrades any existing objects or classes that have the same names but different definitions from objects that you are loading. G2 gives a full report about all objects that it upgrades to the new definitions.



# Using the G2-Database Bridge Workspaces

---

*Describes the contents of the standard workspaces provided with each G2-Database Bridge, and describes techniques for editing the objects and definitions that these workspaces contain.*

Introduction 9

User Modes 9

Keyboard Shortcuts 10

Standard Workspaces of G2-Database Bridges 10



The `g2-database.kb` is the knowledge base provided with your bridge. It has been designed to be merged into an existing modularized knowledge base or to be used as a foundation for building a new knowledge base. The KB contains only the essential components necessary for using the bridge, such as object definitions, RPC declarations, and procedures. It does not contain examples or tutorials.

The KB is organized as one top-level workspace, `g2-database` with several sub-workspaces on which database-specific support items are located.

## User Modes

The G2-Database bridge KB is designed to be used in developer mode. Each workspace contains features that are configured to be accessible in developer mode. These features include shortcuts for entering the editor, viewing item tables, cloning items, creating instances, and moving workspaces. These shortcut features are actually available in any mode except administrator mode. This

means that if you have created your own user mode, it may be used instead of developer mode.

## Keyboard Shortcuts

The following table summarizes the keystroke combinations that you can use to open attribute tables and editors for class definitions, objects, and procedures:

Object	Shift + Click	Ctrl + Click	Ctrl + Alt + Click
Class definition	Opens attribute table	No effect	No effect
Procedure definition	Opens attribute table	Opens editor for editing procedure	No effect
Object instance	Opens attribute table	No effect	No effect
g2-database-interface object	No effect	No effect	Reset connection

Additionally, you can:

- Move a workspace by clicking and dragging it.
- Hide a G2-Database workspace by clicking on the title bar.

Some shortcuts are invoked by simply clicking on the item. These items are usually designated by a red label. For example, clicking on the **g2-database-interface** class definition displays a menu with the **create instance** menu choice, while clicking on an instance of that class displays a menu with the **clone** menu choice.

## Standard Workspaces of G2-Database Bridges

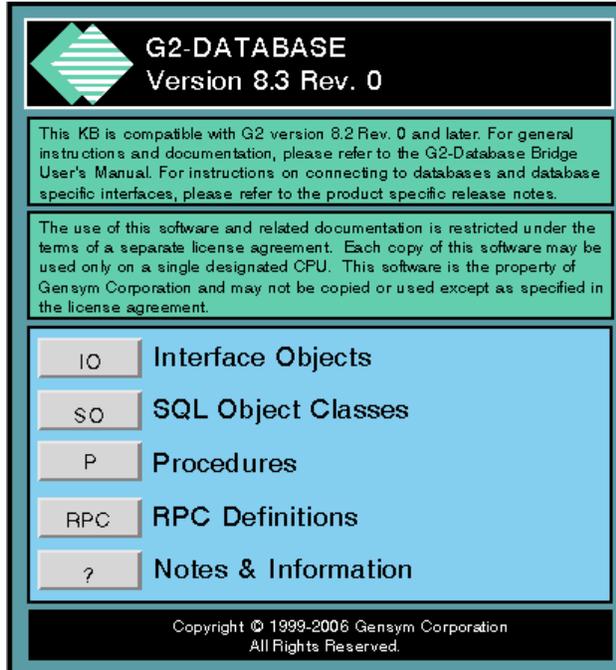
Every G2-Database bridge provides a set of standard workspaces that contain definitions of the object classes and procedures that you use to construct your G2-Database bridge application. The **g2-database** workspace is the top level workspace and provides access to the following workspaces:

- The G2-Database Connection Configuration workspace
- The G2-Database SQL Object Classes workspace
- The G2-Database Procedures workspace
- The G2-Database Remote Procedures Definitions workspace

- The G2-Database Notes & Information workspace

**To open the top-level workspace of your G2-Database bridge:**

➔ Select Main Menu > Get Workspace > g2-database.



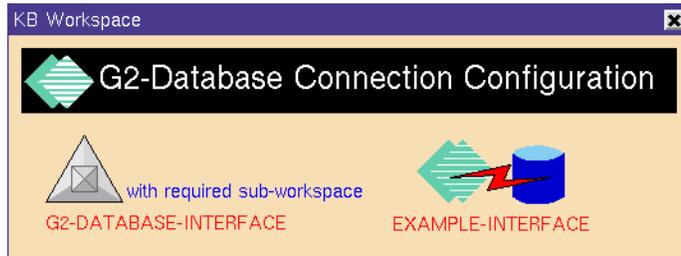
The g2-database workspace contains buttons that you can click to open workspaces containing the definitions that you use to create your G2-Database bridge. The following table lists the workspaces opened by each button:

Click this button...	To display a workspace that contains...
 Interface Objects	The class definition of g2-database interface and a sample instance.
 SQL Object Classes	Class definitions of SQL objects that a G2-Database bridge uses to: <ul style="list-style-type: none"> <li>• Perform SQL operations.</li> <li>• Support message reporting.</li> <li>• Display the current status of connections.</li> <li>• Set the debug option.</li> </ul>

Click this button...	To display a workspace that contains...
 Procedures	Definitions of all the standard G2-Database bridge API procedures and callbacks.
 RPC Definitions	Definitions of all remote procedure calls used internally by G2-Database bridges. The remote procedures are not designed to be directly called by end users.
 Notes & Information	General notes and information on using the G2-Database KB.

## G2-Database Connection Configuration Workspace

The G2-Database Connection Configuration workspace contains the class definition for G2-Database Interface objects, and a sample instance that you can clone to create an interface object for your application:



## G2-Database SQL Object Classes Workspace

The G2-Database SQL Object Classes workspace contains the class definitions that you use to perform database operations through a G2-Database bridge:

KB Workspace

### G2-Database SQL Object Classes

Items, or instances of items, with a red label may be cloned.

**DB-QUERY-OBJECT**

**DB-QO-RECORD** **DB-QO-TABLE**

DB-QUERY-ITEMs & DB-CONFIG-OBJECTs must not be cloned. These objects are created and registered automatically. They are provided to you as return arguments from calls to various API procedures.

**DB-QUERY-ITEM**

**DB-QUERY-ITEM-LIST** **DB-QUERY-ITEM-ARRAY**

**DB-CONFIG-OBJECT**

**DB-SQL-OBJECT** none 0 **DB-CURSOR-OBJECT** none 0

**DB-TRIGGER-OBJECT** none 0 **DB-COLORS-OBJECT**

**DB-ALLOW-OTHER-PROCESSING**  Set db-allow-other-processing to FALSE if you wish to disable "allow other processing" statements in g2-database.kb procedures. Check the check-box to manually disable "allow other processing".

**DB-GLOBAL-VALUE**

**DB-DEBUG-OBJECT** Set db-debug to TRUE if you wish to have status information from each g2-database.kb procedure displayed to the message-board. Check the check-box to manually enable debugging.

**DB-DEBUG**

These definitions include:

- `db-query-object`, `db-qo-record`, and `db-qo-table`, which you use to perform queries using query objects. For information about how to use these definitions, see [Query Objects](#).

---

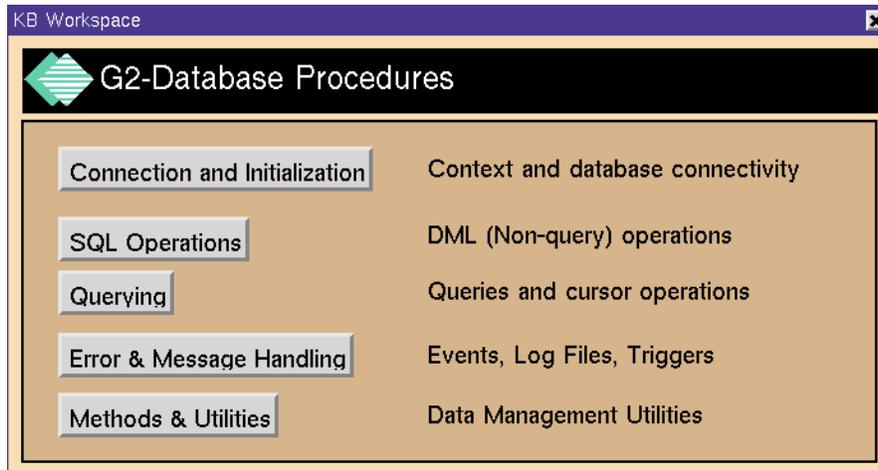
**Note** `db-query-objects` are provided for backward compatibility to applications developed with earlier versions of `g2-database`. Developers are no longer restricted to using `db-query-objects` to fetch data and may now use *any* G2 object as a receptacle for database data.

---

- `db-query-item`, `db-query-item-list`, `db-query-item-array`, and `db-cursor-object`, which you use to perform queries on lists or arrays of items. For information about how to use these definitions, see [Querying the Database](#).
- `db-sql-object`, which you use to perform DML operations. For information about how to use this definition, see [DML Database Operations](#).
- `db-trigger-object`, which you can use to report database trigger events to G2. For information about using this definition, see [Message Handling](#).
- `db-colors-object`, which you can use to display the current status of the connection between G2 and the G2-Database bridge and of various objects on the workspace. For information on using this definition, see [Displaying the Connection Status](#).
- `db-debug-object`, which you use to specify whether you want status information for each G2-Database procedure displayed on the G2 message board.

## G2-Database Procedures Workspace

The G2-Database Procedures workspace contains definitions for the standard API procedures and callbacks provided with each G2-Database bridge:



The procedures are grouped by function. Clicking a button on the G2-Database Procedures workspace displays a workspace of API procedures related to that function. Each procedure is displayed with text indicating the arguments that are required to call the procedure, as well as any returned values.

Any procedure that is displayed with a white border may be edited and modified. However, when upgrading your application with a new version of the G2-Database KB, you will lose all of your customizations. Therefore, we recommend that you create your own procedures that call the G2-Database procedures.

For information about these procedures, see [Bridge Procedures](#).



# Running the Bridge

---

*Describes how to start the bridge process, run a bridge with more than one connection to a G2 process, and run multiple copies of a bridge to improve performance.*

Introduction	17
Command-Line Options	18
Initial Bridge Memory Requirements	20
Text Conversion Styles	20
Starting the Bridge Process	22
Bridge Process Output	23
Establishing a Connection between the Bridge and G2	24
Running a Bridge with Multiple Connections to G2	25
Running Multiple Copies of a Bridge	25



## Introduction

This chapter describes how to:

- Set command-line options that specify how the bridge will process data sets and perform character set conversions.
- Start G2-Database bridges on different platforms. The examples use the default command-line prompt # (for UNIX).

- Run a bridge with more than one connection to a G2.
- Run more than one copy of the same bridge.

## Command-Line Options

Command-line options are used to establish bridge configuration boundaries and behavior, such as character set translation. These options have a global effect on the bridge in that they influence every bridge context. Command-line options are typically specified as option-value pairs, where the option contains a prepended hyphen "-".

For example:

```
-maxrows 1000
```

The following table summarizes bridge command-line options and default values:

Option	Description
-maxrows	<p>The maximum number of database rows that the G2-Database bridge can return to G2 in a single transaction.</p> <p>The default maximum number of rows is 200.</p> <p>You specify the number of records to fetch in the <i>rcds-to-fetch</i> argument of <i>db-fetch-object</i>, <i>db-fetch-query-item</i>, and <i>db-fetch-records</i>. For information about these procedures, see <a href="#">Bridge Procedures</a>.</p> <p>If you attempt to perform a query on a greater number of rows than the current value of <code>MAXROWS</code>, the G2-Database bridge returns a warning message, and the transaction ends when the bridge has returned the number of rows specified by <code>MAXROWS</code>. To obtain all of the rows associated with a query, the query operation must be repeated until an end-of-cursor status is obtained.</p> <p>This option directly affects the initial memory requirements of the bridge. For information on memory implications of this option, see <a href="#">Initial Bridge Memory Requirements</a>.</p>

Option	Description
<code>-maxcols</code>	<p>The maximum number of database columns that the G2-Database bridge can return to G2 in a single transaction.</p> <p>The default maximum number of columns is 30.</p> <p>You specify which columns are included in a query via an SQL statement. The SQL statement is contained within a cursor object. For information on how to create cursor objects, see <a href="#">Creating a Cursor Object</a>.</p> <p>If you attempt to perform a query on a greater number of columns than the current value of <code>MAXCOLS</code>, the transaction is terminated and no data is returned. The G2-Database bridge returns a message indicating that the query has attempted to return data from more columns than the number of columns specified by <code>MAXCOLS</code>.</p> <p>This option directly affects the initial memory requirements of the bridge. For information on memory implications of this option, see <a href="#">Initial Bridge Memory Requirements</a>.</p>
<code>-charset</code>	<p>The text conversion style that the bridge will use to convert between database character sets and G2 character sets.</p> <p>The default text conversion style is Latin-1 (ISO-8859-1)</p> <p>If the database character set is not included within the default character set, then you must set the appropriate text conversion style. This is true for both reading from and writing to the database.</p> <p>For information on supported text conversion styles, see <a href="#">Text Conversion Styles</a>.</p>
<code>-shutdown_on_disconnect</code>	<p>Normally, a database bridge should not shut down when it unexpectedly loses its connection to G2. However, there are systems where the bridge is started and shut down by G2. If G2 is aborted, the bridge must be shut down manually. To stop the bridge when the last connection to it is broken, use the <code>-shutdown_on_disconnect</code> command-line option when starting the bridge.</p>
<code>-help</code>	<p>Displays valid command-line options for your G2-Database bridge.</p>

## Initial Bridge Memory Requirements

You can set the maximum number of rows and columns that the G2-Database bridge can return to G2 in a single transaction, using the command-line options `MAXROWS` and `MAXCOLS`.

These options determine the amount of memory that the bridge allocates for the results of database queries performed through a G2-Database bridge. If you do not specify values for `MAXROWS` or `MAXCOLS` when you start a G2-Database bridge, the bridge uses default values for the maximum number of rows and columns.

Most of the memory that the bridge process uses is allocated when the bridge is initially started. The amount of memory that is allocated is directly related to the `MAXROWS` and `MAXCOLS` settings. The greater the value of the settings, the greater the amount of memory that is allocated. In addition, as more memory is allocated, fewer transactions are required to return data to G2.

For example, to launch a G2-Database bridge, such that its initial memory allocation accommodates up to 1000 rows and 25 columns you would use the following command:

```
# g2-database -MAXROWS 1000 -MAXCOLS 25
```

where:

`g2-database` is the name of the database bridge

## Text Conversion Styles

The bridge will translate character sets between G2 and a database based on the command-line option `CHARSET`. This functionality is achieved via translator routines that convert G2 internal character sets into character sets that are utilized by database vendors.

For example, if you want the bridge to translate the G2 Japanese language character set into the SHIFT-JIS-X-208 character set, you would use the following command to start the bridge:

```
# g2-database -CHARSET SHIFT-JIS-OR-KANJI
```

where:

`g2-database` is the name of the database bridge.

The following table describes the supported values for the `CHARSET` command-line option.

<b>Conversion Style</b>	<b>Character Set</b>	<b>Description</b>
LATIN-1	ISO-8859-1	8-Bit Single Byte
LATIN-2	ISO-8859-2	8-Bit Single Byte
LATIN-3	ISO-8859-3	8-Bit Single Byte
LATIN-4	ISO-8859-4	8-Bit Single Byte
CYRILLIC	ISO-8859-5	8-Bit Single Byte
ARABIC	ISO-8859-6	8-Bit Single Byte
GREEK	ISO-8859-7	8-Bit Single Byte
HEBREW	ISO-8859-8	8-Bit Single Byte
LATIN-5	ISO-8859-9	8-Bit Single Byte
LATIN-6	ISO-8859-10	8-Bit Single Byte
US-ASCII	ISO-646-IRV	7-Bit Single Byte
JIS-X-208	JIS-X-208	7-Bit, JIS X 0208 (Japanese)
JIS-X-208-EUC	JIS-X-208-EUC	8-Bit, JIS X 0208
SHIFT-JIS-OR-KANJI	SHIFT-JIS-X-208 MS-KANJI	Shift JIS Encoded JIS X 0208
KSC-5601	KS-C-5601	7-Bit, KS C 5601 (Korean)
KSC-5601-EUC	KS-C-5601-EUC	8-Bit, KS C 5601
UNICODE-UTF-7	UNICODE-UTF-7	7-Bit Unicode
UNICODE-UTF-8	UNICODE-UTF-8	8-Bit Unicode
ISO-2022	ISO-2022	X compound text with subset of ISO 2022 escapes

# Starting the Bridge Process

This section describes the steps to start the bridge process on each of the supported platforms.

## On UNIX Systems

**To start the bridge on a UNIX system:**

→ Use a command of the following form:

```
# g2-database port-number
```

where:

*g2-database* is the name of the database bridge  
*port-number* is the number of the TCP/IP port.

Defining a port number is not required. If you do not name one, the bridge will select and display a port number. However, it is recommended that you start your bridge on a known port number. When the bridge is started, it will try to use the default port number. If this port number is in use, the bridge will try to use the next sequential port number. The bridge will continue to look for port numbers until it finds one that is not in use. Unfortunately, this approach to selecting port numbers may result in your application not knowing the bridge port number.

The port number must be an integer that is not being used by any other network process on the host machine. Standard services generally use numbers below 3000, so numbers greater than 3000 are probably free. For example, to launch an Oracle bridge for Oracle 9.2, using 22033 as the port number, use the following command:

```
# g2-ora92 22033
```

If you do not know which port number to use, see your system administrator.

---

**Note** To run several copies of the same bridge simultaneously, launch the bridge using a different TCP/IP port number in each command to start the bridge.

---

## On Windows Systems

**To start a database bridge on Windows:**

→ Choose Start > Programs > Gensym G2 2015 > Bridges, then choose a bridge from the submenu.

---

**Note** The G2-Oracle Bridge and G2-Sybase Bridge require certain database libraries to be installed before you can launch the bridge.

---

When the bridge is started, it will try to use the default port number. If this port number is in use, the bridge will try to use the next sequential port number. The bridge will continue to look for port numbers until it finds one that is not in use. Unfortunately, this approach to selecting port numbers may result in your application not knowing the bridge port number. To avoid this problem, it is recommended that you start your bridge on a known port number. This port number can be incorporated directly into an Windows shortcut or can be specified on the command line. The following steps use Windows as an example.

**To change or examine the port number of a bridge from its shortcut:**

- 1 Start Explorer.
- 2 Right-click on the bridge shortcut.
- 3 Choose Properties from the menu.
- 4 Click the Shortcut tab.
- 5 Examine the command-line argument by clicking in the Target field, using the arrow keys to move left and right.
- 6 To change the port number, edit the command line by adding a port number to the end of the Target string and click on OK when you are finished.

If you make a mistake, click Cancel to preserve the original setting. For example:

```
"c:\Program Files\Gensym\g2-2015\oracle\g2-oracle.exe 22055"
```

**To start the bridge manually from a command window:**

➔ Type the fully qualified path and file name, for example:

```
c:\Program Files\Gensym\g2-2015\oracle\g2-oracle
```

Running the bridge from within an MSDOS window is very useful if a bridge does not seem to be starting properly – possibly because of problems with authorization, data files, environment configuration, or missing DLLs.

## Bridge Process Output

When you start the bridge, the window where you start it should display information similar to the following:

```
-----
Starting G2-ODBC Bridge   Version 8.3 Rev 0 (II05-8.3-3)
for Windows
-----
```

```
[28 Sep 2006 11:39:52]
GSI Version 8.3 Rev. 0 Intel NT (II04)
Allocating 200 x 30 data array for object passing...Done
2006-09-28 11:39:52   Waiting to accept a connection on:
2006-09-28 11:39:52   TCP_IP:NORWALK-N800C-2:22041
```

If you get a message similar to:

```
GSI Authorization failed -- GSI_ROOT environment variable not
set.
```

check to make sure your `GSI_ROOT` environment variable is set to the directory containing your `gsi.ok` file.

If you get a message similar to:

```
GSI Authorization -- Could not find valid entry in gsi.ok file
for machine_id=XXXXXXXXXXXXX product=G2-DATABASE version=XXXX
```

check to make sure that the `gsi.ok` file correctly authorizes your bridge.

If you get a message similar to:

```
GSI failed -- Authorization codes are not correct in gsi.ok
```

check to make sure that the `gsi.ok` file correctly authorizes your bridge.

The installation process sets the `GSI_ROOT` environment variable and authorizes your computer to run the G2-Database Bridge. If you receive any of these errors, the installation process failed in some way; reinstall the bridge and try running the bridge process again.

## Establishing a Connection between the Bridge and G2

After you start the bridge process, you must establish a connection between the bridge process and G2. To do this, you enable a `g2-database-interface` object that contains information for configuring the connection that you want to establish. The network information that you specify in the `gsi-connection-configuration` attribute of the interface object must apply to the bridge process that you started.

For more information about interface objects, see [Configuring Connections](#).

## Running a Bridge with Multiple Connections to G2

You can run a G2-Database bridge with connections to several different G2 knowledge bases. You can also run a G2-Database bridge with several different connections to the same G2 knowledge base.

Each connection between a G2 knowledge base and a bridge is configured by a separate `g2-database-interface` object. When you start G2, it establishes a connection to a running bridge for each interface object that is defined and enabled. Each connection between G2 and the bridge is called a **context**.

You can use different contexts to perform different kinds of transactions. For example, you can perform queries through one context, and perform inserts in another context.

You can specify a name for a context in the `context-name` attribute of the `g2-database-interface` object that configures that context. This context name is included in messages that report events occurring in that context, including messages that are reported to log files.

You can also run a G2-Database bridge with connections to more than one database. Each interface object includes configuration information that the bridge uses to establish a connection with a database.

Some database bridges enable you to change the database to which a context is connected by specifying the alias of the database in SQL statements. To find out whether your database bridge enables you to do this, see the Release Notes for your database bridge.

## Running Multiple Copies of a Bridge

To increase processing speed, you can run multiple copies of the same G2-Database bridge. Running multiple copies can increase efficiency if the G2 knowledge base is processing a large volume of transactions, or if several G2 knowledge bases need connections to the database.

A single running copy of the bridge can process only one database operation at a time. Thus, if all database operations are using one bridge, a large query can cause delays in the execution of other operations. If you run several copies of a bridge, one copy can process one request while another copy is processing another request.

You must create and configure at least one `g2-database-interface` object for each copy of a G2-Database bridge that you run.



# Configuring Connections

---

*Explains how to create and define a g2-database-Interface object to configure a connection between your G2 application and a G2-Database Bridge.*

Introduction **27**

Creating G2-Database-Interface-Objects **28**

Attributes of G2-Database-Interface Objects **28**

Sending Connection Configuration Information to the Bridge **44**

Resetting the Interface Connection **45**

Displaying the Connection Status **45**



## Introduction

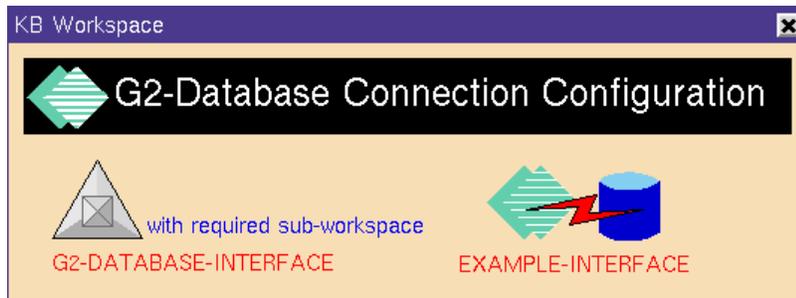
Each g2-database-interface object configures one connection between your G2 application and the bridge. It also includes information that the bridge uses to establish a connection to a database. Therefore, each g2-database-interface object represents a connection (or login) to the database.

# Creating G2-Database-Interface-Objects

**To create a g2-database-interface object:**

- 1 Open the g2-database connection configuration workspace by clicking the Interface Objects button on the g2-database workspace.

This workspace contains the class definition of the g2-database-interface and an instance of the class called example-interface:



- 2 Click on the example-interface and choose clone, or choose create instance from the g2-database-interface class definition.
- 3 Transfer the cloned interface object to a workspace in your application, other than a G2-Database KB workspace, clicking the mouse to release the object.
- 4 If you chose to clone the example-interface, then the cloned interface object is disabled by default.

You must enable the cloned G2-Database interface object by selecting enable from its property table.

- 5 Edit the attributes of the g2-database-interface object, as explained in the following section.

## Attributes of G2-Database-Interface Objects

You must specify values for attributes of a g2-database-interface object in order to configure a connection between G2 and the bridge and to enable the bridge to establish a connection to a database.

**To set the attributes of a g2-database-interface object:**

- ➔ Select table in the menu for the g2-database-interface object that you created.

The following table is displayed:

EXAMPLE-INTERFACE, a g2-database-interface	
Notes	OK, but DISABLED.
Names	EXAMPLE-INTERFACE
Interface warning message level	default to warning message level
Disable interleaving of large messages	no
Interface timeout period	30 seconds
Interface initialization timeout period	unlimited
GSI connection configuration	none
Remote process initialization string	""
GSI application name	default
GSI interface status	0
Interval to poll external system	use default
Maximum definable cursors	10
Null string	""
Null number	0
Set null string	""
Set null number	0
Set null options	100
Enable messaging	false
Log file	""
Context name	""
Database connection status	disconnected
Auto database reconnect	false
Database user	""
Database password	""
Database connect string	""

You can set the attributes of `g2-database-interface` objects as follows:

<b>Attribute</b>	<b>Description</b>
<b>names</b>	Specifies a unique name for this <code>g2-database-interface</code> object.
<i>Allowable values:</i>	Any symbol
<i>Default value:</i>	none
<i>Notes:</i>	You specify this name as an argument of G2-Database bridge procedures that perform queries or other database operations.  You also specify this name as the <code>gsi-interface-name</code> attribute of a Query Object.
<b>interface-warning-message-level</b>	The severity level of error and warning messages about which G2 provides information.
<i>Allowable values:</i>	default to warning message level 0 (no messages) 1 (serious error messages only) 2 (all error messages) 3 (all error and warning messages)
<i>Default value:</i>	default to warning message level
<i>Notes:</i>	Use this setting to control whether failure to connect or connection broken error messages are displayed.
<b>disable-interleaving-of-large-messages</b>	Controls whether the bridge interleaves (changes the transmission order of) message packets.
<i>Allowable values:</i>	<b>yes:</b> Transmits messages without interleaving, preserving their transmission order. With this setting, overall performance can suffer when the messages have very different lengths, because many short messages may have to wait for one long message to complete.  <b>no:</b> Transmits messages with interleaving, which reorders the message packets so that large messages do not lock out smaller messages during large message transmission.
<i>Default value:</i>	no

Attribute	Description
<b>interface-timeout-period</b>	Specifies how long G2 waits for a response to any request that it makes of the bridge.
<i>Allowable values:</i>	Time interval from 1 second to G2's maximum allowable time interval. Specify the time interval in the form: <i>integer</i> {second[s]   minute[s]   hour[s]   day[s]   week[s] }
<i>Default value:</i>	use default (equivalent to 30 seconds)
<i>Notes:</i>	Set this attribute to an appropriate amount of time for G2 to wait for a response to any request that it makes of the bridge. For example, if it typically takes 30 seconds for G2 to make a connection to the database, set the <b>Interface-timeout-period</b> to some time period greater than 30 seconds.  If a connection is not established within the amount of time that you specify, G2 assumes that the connection is timed-out and stops sending requests to the bridge over the connection configured by this g2-database-interface object. G2 then sets the <b>gsi-interface-status</b> attribute to -1.

Attribute	Description
<b>interface-initialization-timeout-period</b>	<p data-bbox="503 304 1266 367">Specifies how long G2 waits to initialize a connection using Gensym (ICP) protocols.</p> <p data-bbox="503 388 1266 451">The following three timeout intervals apply to G2-Database interfaces:</p> <ol data-bbox="503 472 860 619" style="list-style-type: none"> <li data-bbox="503 472 860 514">1 Establish a connection.</li> <li data-bbox="503 525 860 567">2 Initialize the connection.</li> <li data-bbox="503 577 860 619">3 Wait for a response.</li> </ol> <p data-bbox="503 640 1331 735">This attribute applies to the second interval. The <code>interface-timeout-period</code> attribute specifies the timeout period for the first and third intervals.</p> <p data-bbox="243 766 747 808"><i>Allowable values:</i> Possible values are:</p> <ul data-bbox="503 819 1299 997" style="list-style-type: none"> <li data-bbox="503 819 1153 861">• An integer specifying some number of seconds</li> <li data-bbox="503 871 1201 913">• <code>unlimited</code>: the initialization interval never times out</li> <li data-bbox="503 924 1299 997">• <code>use default</code>: the <code>interface-initialization-timeout-period</code> is the same as the <code>interface-timeout-period</code></li> </ul> <p data-bbox="284 1018 1299 1092"><i>Default value:</i> <code>unlimited</code>, which specifies that the initialization interval never times out</p>
<b>gsi-connection-configuration</b>	<p data-bbox="503 1176 1347 1249">Specifies the network protocol that G2 uses to communicate with the bridge.</p> <p data-bbox="243 1270 1201 1312"><i>Allowable values:</i> <code>tcp-ip host "hostname" port-number tcp-ip-port-number</code></p> <p data-bbox="284 1333 568 1375"><i>Default value:</i> <code>none</code></p> <p data-bbox="381 1396 1315 1501"><i>Notes:</i> For the TCP/IP network protocol, you must specify the host name of the machine that is running the bridge, and the socket that can be used to identify the bridge process. For example:</p> <p data-bbox="552 1512 1006 1551" style="text-align: center;"><code>tcp-ip host "xyz" port-number 22033</code></p>

Attribute	Description
<b>remote-process-initialization-string</b>	<p data-bbox="519 304 1430 514">Used to configure this interface object's connection to the database. Features are configured by putting a minus sign followed by a letter code followed, possibly, by a value. For example, if the string is "-A -b60 -F", the -A enables auto-commit, -b60 sets the maximum number of bind variables in a query to 60, and -F turns on smart fetch.</p> <p data-bbox="519 535 1430 598">The letter codes are case-sensitive, e.g., -F does not have the same meaning as -f.</p> <p data-bbox="292 630 519 661"><i>Allowable values:</i></p> <ul data-bbox="519 630 1430 1186" style="list-style-type: none"> <li>-A = auto-commit ON</li> <li>-b = maximum number bind variables</li> <li>-f = logfile filter</li> <li>-F = smart fetch</li> <li>-n = max bind name length</li> <li>-c, -C, -o = disable output to console</li> <li>-p = poll interval (in sec) for trigger checking</li> <li>-r = maximum number of registered items</li> <li>-t = turn logfile time-stamping OFF</li> <li>-T = turn iormsg time-stamping OFF</li> <li>-d = debugging flags</li> </ul> <p data-bbox="292 1218 519 1249"><i>Default value:</i> "" (empty string). Auto-commit off, enables the display of error and warning messages on the console and does not use "smart fetch."</p> <p data-bbox="422 1354 519 1386"><i>Notes:</i></p> <p data-bbox="519 1354 1430 1480">The bridge always displays startup messages that include information such as the context, configuration settings, and the status of the connection to the database. You cannot disable the display of these messages.</p> <p data-bbox="519 1501 1430 1606">The -c parameter does not disable the writing of bridge messages to log files, or prevent GSI messages from being printed on the screen when you start the bridge.</p> <p data-bbox="519 1627 1430 1690">For information on the "smart fetch" option, see <a href="#">Using Smart Fetch</a>.</p> <p data-bbox="519 1711 1430 1774">For information on the "debugging flags" option, see <a href="#">Debugging Facility</a>.</p>

Attribute	Description
<b>gsi-application-name</b>	The name of the G2-Database Bridge application.
<i>Allowable values:</i>	symbol
<i>Default value:</i>	default
<b>gsi-interface-status</b>	A value displayed by the bridge to indicate the current status of the connection between the bridge and the G2 application.
<i>Allowable values:</i>	<p>2 (OK): The connection has been made and is active.</p> <p>1 (Initializing): The external system is being initialized. When G2 receives this code, it refrains from sending messages to the bridge until it receives the OK code (2).</p> <p>0 (Inactive): The connection is disabled or inactive.</p> <p>-1 (Timeout): The connection has been timed out because G2 has not heard from the bridge within the time specified by the <code>Interface-timeout-period</code> attribute.</p> <p>-2 (Error): An error has occurred, and the connection is broken.</p>
<i>Default value:</i>	0
<i>Notes:</i>	See the <i>G2 Gateway Bridge Developer's Guide</i> for more information.
<b>interval-to-poll-external-system</b>	Controls the polling interval when the <code>poll-external-system-for-data</code> attribute is set to <code>yes</code> .
<i>Allowable values:</i>	<p>Time interval from 1 second to G2's maximum allowable time interval</p> <p>Specify the time interval in the form:</p> <p style="padding-left: 40px;"><i>integer</i> {second[s]   minute[s]   hour[s]   day[s]   week[s] }</p> <p>use default (equivalent to 1 second)</p>
<i>Default value:</i>	use default

Attribute	Description
<b>maximum-definable-cursors</b>	Defines the maximum number of open cursors allowed for the context defined by this g2-database-interface object.
<i>Allowable values:</i>	Any integer less than 201
<i>Default value:</i>	10
<i>Notes:</i>	Some databases also have a maximum number of cursors value. Make sure that the total number of bridge cursors and cursors required by other database clients does not exceed the maximum number configured for the database. This attribute is for application control and does not affect the bridge. See the database-specific Release Notes and your System Administrator for more information.
<b>null-string</b>	Specifies the return value for queries when NULL text values are returned to G2 from a database.
<i>Allowable values:</i>	Any text string
<i>Default value:</i>	""
<b>null-number</b>	Specifies the return value for queries when NULL numeric values are returned to G2 from a database.
<i>Allowable values:</i>	Any integer or floating point number
<i>Default value:</i>	0 (0.0 for float values)
<b>set-null-string</b>	Defines the textual bind value that represents NULL.
<i>Allowable values:</i>	text
<i>Default value:</i>	""
<i>Notes:</i>	See <a href="#">Using the Set Null Attributes</a> .
<b>set-null-number</b>	Defines the numeric bind value that represents NULL.
<i>Allowable values:</i>	integer

Attribute	Description
<i>Default value:</i>	0
<i>Notes:</i>	See <a href="#">Using the Set Null Attributes</a> .
<b>set-null-option</b>	Determines whether the prior two attributes are used and also whether a textual bind value of "NULL" represents NULL. A non-zero value in the ones digit means "use set-null-number". A non-zero value in the tens digit means "use set-null-string". A non-zero value in the hundreds digit means "NULL" represents NULL.
<i>Allowable values:</i>	See description
<i>Default value:</i>	100
<i>Notes:</i>	See <a href="#">Using the Set Null Attributes</a> .
<b>enable-messaging</b>	Specifies whether bridge messages are sent to G2.
<i>Allowable values:</i>	true (messaging enabled) false (messaging disabled)
<i>Default value:</i>	false
<i>Notes:</i>	If this attribute is set to true, the bridge sends error information, and warning messages to G2 by calling <a href="#">db-context-event-msg</a> . You can also indicate which G2 procedure receives messages by calling the procedure <a href="#">db-redirect-callback</a> .

Attribute	Description
<b>log-file</b>	Enables logging of information and error messages for the context configured by this g2-database-interface object.
<i>Allowable values:</i>	A text string containing the pathname of the file where messages are logged or the empty string ("")
<i>Default value:</i>	""(disables logging of messages)
<i>Notes:</i>	Enter, in quotation marks, the full path and name of the file where you want errors logged. Entering a log file name enables logging of information and error messages.
	After the log file name, you can specify a message filter by listing the categories of messages ( <b>fatal</b> , <b>error</b> , <b>warn</b> , or <b>info</b> ) that you want to be logged in the log file.
	You can separate the log file name and filters with spaces or commas.
	To specify a log file name with spaces, surround the name with double quotes, preceded by the @ character, in addition to the double quotes.
	Windows example (assumes the C: drive):
	<code>"C:\usr\test\test.log, ERROR, WARN"</code>
	UNIX example:
	<code>"/usr/test/test.log, ERROR, WARN"</code>
	Here is a Windows example using a file name that has a space:
	<code>"@"C:\usr\test\log file.log@", ERROR, WARN"</code>
	If you do not specify any categories, messages of all categories are logged.
	The bridge opens the log file when the connection between the bridge and G2 is established. You can subsequently open and close the log file or change the filters specified for it by calling <a href="#">db-logfile</a> .

Attribute	Description
<b>context-name</b>	Specifies a name to identify the connection configured by this g2-database-interface object.
<i>Allowable values:</i>	Any text string not exceeding 15 characters
<i>Default value:</i>	""
<i>Notes:</i>	All messages sent by <a href="#">db-context-event-msg</a> to G2 or to a log file include this context name.
	If you do not specify a name for <b>context-name</b> , the bridge automatically creates and assigns a context name in the format <b>context-<i>n</i></b> , where <i>n</i> is an integer. The first connection established from G2 to the bridge is <b>context-0</b> , the second is <b>context-1</b> , and so on.
<b>database-connection-status</b>	Indicates the current status of the connection between the G2-Database bridge and the database.
<i>Allowable values:</i>	connected or disconnected.
<i>Default value:</i>	disconnected
<i>Notes:</i>	This value is updated after every transaction, including both transactions initiated by the user and database triggers.

Attribute	Description
<b>auto-database-reconnect</b>	Determines whether to attempt to reestablish a connection to the database if that connection is broken.  If <b>true</b> , the bridge attempts to reestablish a connection to the database if that connection is broken. If <b>false</b> , the bridge does not attempt to reestablish the connection.  <i>Allowable values:</i> <b>true</b> (attempt to reconnect) <b>false</b> (do not attempt to reconnect)  <i>Default value:</i> <b>false</b>  <i>Notes:</i> The bridge determines that the connection has been broken when it receives a <b>disconnected</b> error code after a failed database operation.  If the bridge fails to reestablish the connection, it sends an information message to the <a href="#">db-context-event-msg</a> procedure (if it is enabled) and to the log file (if one exists).  Following a reconnect, the user must resubmit the last transaction.
<b>database-user</b>	Specifies the database user name, which the bridge uses when it establishes a connection to the database.  <i>Allowable values:</i> Any text string  <i>Default value:</i> ""  <i>Notes:</i> See the Release Notes for your particular G2-Database bridge for information about how to specify user names.
<b>database-password</b>	Specifies the database user password, which the bridge uses when it establishes a connection to the database.  <i>Allowable values:</i> Any text string  <i>Default value:</i> ""  <i>Notes:</i> See the Release Notes for your particular G2-Database bridge for information about how to specify user passwords.

Attribute	Description
<b>database-connect-string</b>	Specifies the database connection information that the bridge uses to build the complete database connection string to establish a connection to the database.
<i>Allowable values:</i>	Any text string
<i>Default value:</i>	""
<i>Notes:</i>	See the Release Notes for your particular G2-Database bridge for information about how to specify this information.

## Using the Set Null Attributes

The `null-number` and `null-string` attributes of the `g2-database-interface` class determine how NULLs that are fetched from database tables are presented to G2. However, they do not provide you with a way of inserting NULL into a table.

G2 provides a way of inserting NULLs into tables that is consistent for all the database bridges.

### Inserting Values into a Table

There are three common methods of inserting values into a table from G2:

- `db-execute-immediate`

This procedure directly executes a single SQL command just as if you submitted it to your database's SQL execution program. For example, you could execute the command:

```
INSERT INTO abts_emp VALUES('Smith',554,NULL)
```

- `db-exec-sql`

This is similar to `db-execute-immediate` in that it processes a single command at a time. However, unlike, `db-execute-immediate`, the single command may include bind variables. The most time-consuming portion of the procedure occurs when `db-define-sql` creates a `db-sql-object`. Values are attached to the bind variables either as part of the `db-define-sql` procedure or with the `db-set-sql` procedure. The values are actually inserted into the table when the procedure `db-exec-sql` is called.

- `db-exec-sql-obj`

Like `db-exec-sql`, the SQL statement that is executed by `db-exec-sql-obj` may contain bind variables. And like `db-exec-sql`, the entire procedure consists of three parts (`db-define-sql-obj`, `db-set-sql-obj`, and `db-exec-sql-obj`). However, unlike `db-exec-sql`, the bind values may come from a list or an array, and, as a

result, a single call to `db-exec-sql-obj` can result in the single SQL statement being executed numerous times instead of just once.

### The Set-Null Attributes of `g2-database-interface`

The `g2-database-interface` class provides these attributes:

- `set-null-string` – Defines the textual bind value that represents NULL.
- `set-null-number` – Defines the numeric bind value that represents NULL.
- `set-null-options` – Determines whether the prior two attributes are used and also whether a textual bind value of "NULL" represents NULL. A non-zero value in the ones digit means "use `set-null-number`". A non-zero value in the tens digit means "use `set-null-string`". A non-zero value in the hundreds digit means "NULL" represents NULL.

The database bridge reads the settings of these attributes when G2 first connects to the bridge. If you change the settings of the attributes, you must reset the interface object before the bridge will use the new values.

The following examples show how to use the new attributes with the three insertion methods to insert NULL into a table.

#### Using `db-execute-immediate` to Insert Null

Since SQL statements used with `db-execute-immediate` do not use bind variables, there has been no change to the behavior of this procedure. It is possible to insert NULL with `db-execute-immediate` by executing the same SQL statement you would use to insert NULL with your database's SQL execution program, for example, SQL\*Plus or `isql`. For example, if the statement to execute is:

```
INSERT INTO abts_emp VALUES('Smith',554,NULL)
```

NULL is inserted into the third column.

#### Using `db-exec-sql` to Insert Null

`db-exec-sql` can work with bind variables. If you sometimes want to insert NULLs into the columns associated with a bind variables and other times want to insert non-NULL values, you will use the `set-null-options` attribute and possibly the `set-null-string`.

An unusual aspect of working with this form of insertion is that values that are associated with bind variables are always text even when the bind value will be inserted into a numeric field. As a result, when working with this insertion method, we always use "NULL" or the value in the `set-null-string` attribute to insert a NULL, never the value of the `set-null-number` attribute. This is true even when we want to insert a NULL in a numeric field.

For example:

The table `abts_emp` has 3 columns: `name`, `employee ID`, and `supervisor's ID`. The `name` column holds text and the two ID columns hold numbers.

Suppose the value of `set-null-string` is "Stockholders", the value of `set-null-number` is 999, and the value of `set-null-options` is 111 (all options enabled). Suppose you execute the following statements with the ODBC bridge, where ? represents bind variables:

```
db-define-sql (the symbol SOX, "INSERT INTO abts_emp
VALUES(?, ?, ?)", "", dbio) ;
db-set-sql (SOX, "Smith,999,200",dbio) ;
db-exec-sql(SOX, true, dbio) ;
db-set-sql (SOX, "Jones,1000,Stockholders") ;
db-exec-sql(SOX, true, dbio) ;
db-set-sql (SOX, "Null",1001,1000) ;
db-exec-sql(SOX, true, dbio) ;
```

The following three rows would be added to the table:

"Smith"	999	200
"Jones"	1000	NULL
NULL	1001	1000

There are three points to notice about this example:

- In every case, the `db-set-sql` statement provides the values to be bound as part of a string. For example the bind values in the first `db-set-sql` statement are "Smith,999,200". Although `db-set-sql` eventually breaks this string into three parts and converts the 999 and 200 to numbers, the values are originally provided as part of a string. This is the reason `set-null-string` is used instead of `set-null-number` when working with `db-exec-sql`. This is the reason that 999 was inserted into the Employee ID field of the Smith record even though the setting of the `set-null-number` is 999.
- A NULL was inserted into the Supervisor ID field of the Jones record because the bind value, "Stockholders", was the same the value of the `set-null-string` and the tens digit of `set-null-options` was non-zero.
- The example inserted a database NULL instead of the text "Null" into the employee name column of the third row. This is probably not what you intended. If you want to insert Mr. Null's name into the table, you should have set the hundreds digit of `set-null-options` to zero. If `set-null-options` had been set to 11, the result would be the following, as expected:

"Null"	1001	1000
--------	------	------

If `set-null-options` had been set to 0, you would get an error when you tried to execute the second `db-set-sql` command, because it is not possible to insert the text "Stockholders" into the numeric supervisor's ID field.

### Using `db-exec-sql-obj` to Insert Null

From the standpoint of inserting NULLs, the difference between `db-exec-sql` and `db-exec-sql-obj` is that `db-exec-sql-obj` uses the `set-null-number` attribute for determining whether or not to insert NULLs into numeric fields.

This example uses `db-exec-sql-obj` to insert values into `abts_emp`, the table from the previous example.

First, define a class that will be used to hold the values to be inserted.

Next, create an instance of the class. In our example, we name the instance `src-obj`.

Now, set the `set-null-` attributes of the `g2-database-interface` to the same values they had in the prior example: `set-null-string` is "Stockholders", `set-null-number` is 999, and `set-null-options` is 111 (all options enabled).

Suppose you execute this procedure:

```
set-null-example()
s: symbol ; c: integer ; m: text ;
sox: class db-sql-obj ;
begin
  conclude that the g2-list-sequence of the employee-name of
    src-obj = sequence("Smith","Jones","Null");
  conclude that the g2-list-sequence of the employee-id of
    src-obj = sequence(999, 1000, 1001) ;
  conclude that the g2-list-sequence of the supervisor-id of
    src-obj = sequence(200, 999, 1000)
  sox, s, c, m = call db-define-sql-obj(the symbol SOX,
    "INSERT INTO abts_emp VALUES (:1, :2, :3)",dbio);
  s, c, m = call db-set-seq-obj(sox, "employee-name, employee-id,
    supervisor-id", src-obj, dbio) ;
  s, c, m = call db-exec-sql-obj(sox, true, dbio)
end
```

The result would be that the following three rows would be added to the table:

```
"Smith"  NULL  200
"Jones"  1000  NULL
NULL     1001  1000
```

Things to note about this example are:

- We are using Oracle-style bind variables in this example, i.e., :1, :2, and :3.
- The results were different from the previous example. The employee ID of "Smith" was set to NULL in this case. The reason is that this form of insertion uses `set-null-number` to determine whether NULL should be inserted into numeric fields. The prior example used `set-null-string` for all cases.
- Whereas in the prior example we bound "Stockholders" to the bind variable for supervisor ID, that was not possible in this case. If the third `conclude` statement had been:

```
conclude that the g2-list-sequence of the supervisor-id of
src-obj = sequence(200, "Stockholders", 1000)
```

a runtime error would have occurred when we tried to execute it, because it is not possible to store text in an integer-list.

- Once again, we inserted NULL in the name field of the third row. The same solution would work in this case: change the setting of `set-null-options` to 11.

## Sending Connection Configuration Information to the Bridge

The configuration information that you specify when you define a `g2-database-interface` object is sent to the bridge by a required rule on the subworkspace of the `g2-database-interface` class.

This rule sends configuration information to the bridge for every active connection. To determine which connections are active, the rule examines the `gsi-interface-status` attribute (2 = active) of each `g2-database-interface` object.

For each active connection, the rule calls the bridge procedure [db-startup](#). This procedure then calls the [db-configuration](#) procedure to send configuration information to the bridge. If `db-configuration` is successful, `db-startup` then calls [db-connect](#) to establish the connection between the bridge and the database. For information about these procedures, see [Bridge Procedures](#).

We recommend that you allow this rule to send all the connection configuration information to the bridge.

### To change the configuration for a connection:

- 1 Disable the `g2-database-interface` object that configures the connection.
- 2 Change the attributes of the `g2-database-interface` object to provide the new configuration values.
- 3 Reenable the `g2-database-interface` object.

## Resetting the Interface Connection

Sometimes, it might be necessary to reset the connection of the `g2-database-interface` object from G2 to the bridge. You can reset the connection in any user mode other than administrator.

**To reset the connection:**

→ Choose `reset interface` on the `g2-database-interface` object.

or

→ Press Control + Alt + Click on the `g2-database-interface` object.

## Displaying the Connection Status

You can cause regions of a `g2-database-interface` icon to change color when there is a change in the status of the connection that the `g2-database-interface` configures. The changing color provides an ongoing visual indication of changes to the status of the connection.

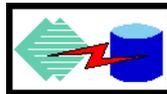
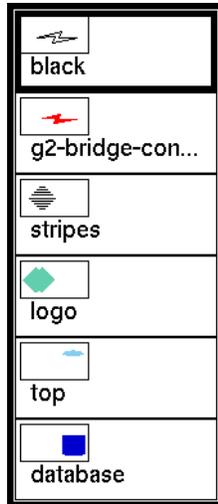
**To display the status of a connection on the `g2-database-interface` icon:**

- 1 Click the SQL Object Classes button on the `g2-database` workspace.
- 2 Click the check box next to the `db-colors-object` on the G2-Database SQL Object Classes workspace to enable color changing of the `g2-database-interface` icon. For example:



- 3 Click the Interface Objects button on the `g2-database` workspace.

- Choose **edit icon** from the **g2-database-interface** object definition menu on the G2-Database Connection Configuration workspace to display the regions of the g2-database-interface icon. For example:



Note the name of each region. For information about how to use the G2 Icon Editor, see the *G2 Reference Manual*.

- Choose **table** from the **db-colors** object menu on the G2-Database SQL Object Classes workspace to display its attributes.
- Specify colors for each icon region.

The following table describes the color attributes of the **db-colors** object:

Attribute	Description
default-color	Specifies the default color.
not-connected-body-color	Specifies the <b>database</b> region color of the icon when the G2-Database Bridge and the database are <i>not connected</i> .
not-connected-top-color	Specifies the <b>top</b> region color of the icon when the G2-Database Bridge and the database are <i>not connected</i> .
connected-body-color	Specifies the <b>database</b> region color of the icon when the G2-Database Bridge and the database are <i>connected</i> .
connected-top-color	Specifies the <b>top</b> region color of the icon when the G2-Database Bridge and the database are <i>connected</i> .

Attribute	Description
error-color	<p>Specifies the color of one or more icon regions when an error has occurred and the connection between the bridge and G2 is broken or the connection is inactive [i.e., when the <code>gsi-interface-status</code> attribute changes to -2 (error) or 0 (inactive)].</p> <p>You specify the region or regions to which this change applies, using the <code>db-io-status</code> method.</p>
warning-color	<p>Specifies the color of one or more icon regions when the external system is being initialized or the connection between the bridge and the G2 has timed out [i.e., when the <code>gsi-interface-status</code> attribute changes to 1 (Initializing) or -1 (Timeout)].</p> <p>You specify the region or regions to which this change applies, using the <code>db-io-status</code> method.</p>
success-color	<p>Specifies the color of one or more icon regions when the connection between the bridge and the G2 has been made and is active [(i.e., when the <code>gsi-interface-status</code> attribute changes to 2 (OK)].</p> <p>You specify the region or regions to which this change applies, using the <code>db-io-status</code> method.</p>

## Changing Icon Colors

If you wish to modify the way in which colors are set for `g2-database-interface` objects, you can edit the `db-io-status` method. This method allows you to set regions of the `g2-database-interface` icon to different colors whenever the value of the `gsi-interface-status` attribute of a `g2-database-interface` object changes.

### To change icon colors to reflect the connection status of the interface object:

- 1 Click the Procedures button on the `g2-database` workspace.
- 2 Click the Methods & Utilities button on the G2-Database Procedures workspace.

- 3 Edit the `db-io-status` procedure to specify color changes to regions of the `g2-database-interface` icon.

Through `db-io-status`, you can also change the color of the **status** region of cursor objects, SQL objects, trigger objects, query items, and query objects.

# DML Database Operations

---

*Describes how to perform DML (non-query) database operations through a G2-Database bridge.*

Introduction	49
Components of a DML Database Operation	50
Bind Variables in SQL Statements	50
Procedures for DML Database Operations	51
Database Operations Using Simple Values	52
Database Operations Using Objects	53
Database Operations without Bind Variables	54



## Introduction

Through a G2-Database bridge, a G2 application can perform inserts, deletes, updates, and other SQL DDL or DML operations (non-query) on values in a database.

The G2 application can perform the SQL operations on individual values, or on lists or arrays of values.

# Components of a DML Database Operation

Each G2-Database bridge provides a set of procedures that your G2 application can invoke to define and execute SQL operations.

If you want to execute an SQL statement that does not contain bind variables or that you do not intend to execute again in the future, then you can call `db-execute-immediate`.

If you want to use bind variables or want to save an SQL operation for future execution, then you must define an SQL object for each particular database operation that you want to perform. An **SQL object** is a G2 object that contains an SQL statement that defines the database operation. You define SQL objects by calling `db-define-sql`.

Your G2 application can use an SQL object any number of times to repeat a particular database operation. You can change the values of bind variables in an SQL statement or in an existing `db-sql-object` by calling [db-set-sql](#) or [db-set-sql-obj](#). These procedures are described in [Bridge Procedures](#).

---

**Note** You should not attempt to manually create or clone SQL objects. If you want to create a new SQL object then you should use the G2-Database API procedure `db-define-sql`.

---

## Bind Variables in SQL Statements

To increase the flexibility and efficiency of database operations, you can use **bind variables** in SQL statements. You assign values to the bind variables to specify the data that you want to insert, delete, or update.

You can change the *values* of bind variables in a previously-defined SQL statement without forcing the database to generate a new execution plan for the SQL statement when you define it again. However, if you make changes to a defined SQL statement itself, that is, if you change the sequence of characters, case, spaces, or punctuation in the statement, you force the database to generate a new execution plan for the SQL statement resulting in additional database processing and overhead.

You *must* include bind variables in SQL objects that you use to perform database operations on lists or arrays of values. For information about how to do this, see [Database Operations Using Objects](#).

The syntax that you must use to identify bind variables in SQL statements is specific to each database. For information about the syntax, see the documentation for your database or the Release Notes for your G2-Database bridge.

The following example illustrates an SQL statement for an Oracle database, which uses preceding colons to identify bind variables (in this example, :empvariable).

```
insert into employees values (:empvariable)
```

You can assign values to bind variables when you create db-sql-objects and also change the values of bind variables in existing SQL objects. For information on assigning values to bind variables, see the next section.

## Procedures for DML Database Operations

G2-Database bridges enable G2 applications to perform DML operations, using both simple values and objects (whose attributes are lists or arrays of values). The bridges provide one set of procedures for performing DML operations with simple values, and another set of procedures for performing DML operations using objects.

The following table lists the G2-Database bridge procedures that you can use to perform database operations on simple values or on objects:

Operation	For Operations on Simple Values	For Operations on Objects
Execute an SQL statement	<b>db-execute-immediate</b> Sends the bridge an SQL statement that is immediately executed by the database.	Not Applicable.
Create an SQL object.	<b>db-define-sql</b> Defines an SQL statement, within the database, that can contain bind variables and bind variable values.	<b>db-define-sql-obj</b> Defines an SQL statement, within the database, that <i>must</i> contain bind variables.  Values for these bind variables must be provided by a user-defined G2 object.

<b>Operation</b>	<b>For Operations on Simple Values</b>	<b>For Operations on Objects</b>
Set values of bind variables in an SQL object.	<b>db-set-sql</b> Sets the values of bind variables within a previously defined SQL statement.	<b>db-set-sql-obj</b> Sets the values of bind variables within a previously defined SQL statement.  The values for the bind variables are provided by a user-defined G2 object.
Perform the database operation defined by the SQL object.	<b>db-exec-sql-obj</b> Executes a previously defined SQL statement within the database.	<b>db-exec-sql-obj</b> Executes a previously defined SQL statement within the database.  A user-defined G2 object contains values for the bind variables.

For detailed descriptions of these procedures and examples of how to use them, see [Bridge Procedures](#).

## Database Operations Using Simple Values

You can perform operations with simple values in a database. Each operation is performed within a separate database transaction.

### To create an SQL object:

→ Call db-define-sql.

The following example creates an SQL object named `sql-obj` and specifies values for its `sql-stmt` and `bind-vars` attributes, where `myIO` is the interface object used to connect G2 to the bridge:

```
CreateSQLObj()
  sql-stmt: text = "insert into emp (empno, ename) values (:nr,:name)";
  bind-vars: text = "7250, Smith";
  sobX: class db-spl-object;
  s: symbol; c: integer; m: text;
begin
  sobX, s, c, m = call db-define-sql
    (the symbol sql-obj-name, spl-stmt, bind-vars, myIO);
  transfer sobX to this workspace
end
```

For additional information on using this procedure, see [db-define-sql](#).

#### **To change the values of bind variables in an existing SQL object:**

→ Call the procedure `db-set-sql`.

The following command binds `7267` to `:nr` and binds `Jones` to `:name` in the SQL object `sql-obj-name`, which was created in the prior example to create an SQL object:

```
s, c, m = call db-set-sql(SQL-OBJ-NAME, "7267,Jones", myIO)
```

For additional information on using this procedure, see [db-set-sql](#).

#### **To execute the database operation defined in the SQL object.**

→ Call the procedure `db-exec-sql`.

The following command executes the SQL statement defined in the SQL object `sql-obj-name`, which was created in the prior example to create an SQL object:

```
s, c, m, nrows = db-exec-sql(SQL-OBJ-NAME, auto-commit, myIO)
```

For additional information on using this procedure, see [db-exec-sql](#).

## Database Operations Using Objects

To reduce the number of procedure calls required for your database operations, you can use objects. If the attributes of an object are either lists or arrays of values, only one G2 procedure call is required to perform numerous database operations.

You must use bind variables in your SQL statements if you wish to perform multiple operations in a single step using lists or arrays values.

#### **To create an SQL object:**

→ Call `db-define-sql-obj`.

For example, the following call creates an SQL object named my-sql-obj:

```
sql-obj-name: symbol = the symbol MY-SQL-OBJ;
sql-obj: class db-sql-object;
sql-stmt: text = "insert into emp (ename,age) values (:ename,:age)";

sql-obj,status,code,msg, = call db-define-sql-obj(sql-obj-name, sql-stmt,
myIO);
```

For additional information on using this procedure see [db-define-sql-obj](#).

**To associate a user-defined G2 object with the SQL object, and map attributes of the G2 object to bind variables in the SQL statement:**

→ Call db-set-sql-obj.

For example:

```
status,code,msg, = call db-set-sql-obj(sql-obj, bind-var-names,
myOBJ, myIO);
```

For additional information on using this procedure see [db-set-sql-obj](#).

**To execute the SQL operation defined in the SQL object:**

→ Call db-exec-sql-obj.

This procedure passes to the bridge an SQL statement that includes bind variables, and a user-defined G2 object that contains values for the bind variables. The bridge extracts values from attributes of the G2 object and assigns them to the bind variables.

For example, the following call executes the SQL operation defined in the SQL object sql-obj:

```
status, code, msg, nrows = call db-exec-sql-obj(sql-obj, auto-commit,
myIO) ;
```

For additional information on using this procedure see [db-exec-sql-obj](#).

## Database Operations without Bind Variables

You can execute SQL operations that do not contain bind variables by calling the db-execute-immediate procedure. The SQL statement that it executes is not saved in the KB, and the database must devise an execution plan each time this procedure is executed. In addition, a SQL object is not created.

For example, the following statement executes the SQL statement assigned to sql-stmt across the interface object named myIO:

```
status: symbol;
code: integer;
msg: text;
nrows: integer;
```

```
sql-stmt: text = "insert into emp (ename) values ('Smith')";  
auto-commit: truth-value = TRUE;
```

```
status, code, msg, nrows = call db-execute-immediate(sql-stmt,  
auto-commit, myIO);
```

You can also execute SQL operations that do not contain bind variables by calling [db-exec-sql](#), also described in [Database Operations Using Simple Values](#). This approach is useful if you wish to repeatedly execute the same SQL statement.



# Querying the Database

---

*Provides an overview of the basic methods of querying a database.*

Introduction	57
Bind Variables in Database Queries	58
Returning Query Data to G2	58
Creating a Cursor Object	59
Returning Query Data in Query Items	60
Returning Query Data to Existing G2 Items	61
Returning Query Data in Structures	62
Copying Query Item Attribute Values	62
Deleting Query Items	63
Using “Smart Fetch”	63



## Introduction

Through a G2-Database bridge, your G2 application can query databases for individual values or for lists, arrays or sequences of values. To perform a query through a G2-Database bridge, your G2 KB must invoke one of the G2-Database bridge API procedures for performing database queries.

Every G2-Database Bridge procedure that performs a database query *must* reference an existing cursor object. You must create a G2 object known as a **cursor**

**object** for each query that you want to perform. You create cursor objects by calling the procedure `db-define-cursor`.

The cursor object contains an SQL statement that defines the query. The cursor object also provides a reference to the database cursor for the query. A **database cursor** is a table that the database maintains internally to identify the rows and columns included in the query.

## Bind Variables in Database Queries

To increase the flexibility and efficiency of database queries, you can use **bind variables** in SQL statements. You assign values to bind variables in order to specify the data for which you want to query.

You can change the *values* of bind variables in a previously-defined SQL statement without forcing the database to generate a new execution plan for the SQL statement when you define it again. However, if you make changes to a defined SQL statement itself, that is, if you change the sequence of characters, case, spaces, or punctuation in the statement, you force the database to generate a new execution plan for the SQL statement, resulting in additional database processing and overhead.

The syntax that you must use to identify bind variables in SQL statements is specific to each database. For information about the syntax, see the documentation for your database or the release notes for your G2-Database bridge. The following example illustrates an SQL statement for an Oracle database, which uses preceding colons to identify bind variables (in this example, `:id`):

```
select * from employees where empid = :id
```

You assign values to bind variables when you create cursor objects. You can also change the values of bind variables in existing cursor objects. For information about how to assign values to bind variables, see [Creating a Cursor Object](#).

## Returning Query Data to G2

G2-Database bridges can return query data to G2 by:

- Passing the query data back to G2 in query items that the bridge generates dynamically. The database bridge creates these item, fills them with the results of your query, and sends them to G2 through object-passing.

When G2 receives the query data, it generates an item of the class `db-query-item-array` or `db-query-item-list`. G2 writes the query data to attributes of these items.

- Returning a user-defined object or a list of objects to G2.

- Returning a G2 structure where the elements of the structure contain name/sequence pairs.
- Returning query data to attributes of an existing user-defined G2 object.
- Returning query data to attributes of an existing predefined G2 object, known as a **query object**. For information on creating and using query objects, see [Query Objects](#).
- Returning a single text or quantity value as a result of a remote procedure call return value.

The following table summarizes the G2-Database Bridge procedures that you can use to perform queries on a database.

To...	And return data to an existing G2 object, call...	And return data within a new or dynamically created object, call...
Create a Cursor object	db-define-cursor	db-define-cursor
Set values of bind variables in a cursor object	db-set-cursor	db-set-cursor
Perform the database operation defined by the cursor object	db-fetch-records db-update-object	db-fetch-query-item db-update-query-item db-fetch-object db-fetch-structure

For detailed descriptions of these procedures and examples of how to use them, see [Bridge Procedures](#).

## Creating a Cursor Object

Your G2 application can use a cursor object any number of times to perform a particular query repeatedly. You can create a new cursor object or change the SQL statement in an existing cursor object by calling the procedure `db-define-cursor`.

---

**Caution** Do not attempt to create a cursor object by cloning, or to modify a cursor object directly by editing values in its attribute table. If you need to create or modify a cursor object, call `db-define-cursor` or `db-set-cursor` respectively.

---

**To create a cursor object:**

➔ Call `db-define-cursor`.

This procedure creates a cursor object and specifies values for its `sql-stmt` and `bind-vars` attributes. For example:

```
cursor-obj,status,code,msg = call db-define-cursor(cursor-obj-name,  
sql-stmt, bind-vars, myIO);
```

For complete details on using this procedure, see [db-define-cursor](#).

**To change the values of bind variables in the SQL statement associated with an existing cursor object:**

→ Call `db-set-cursor`.

This procedure sets the value of the `bind-vars` attribute of the cursor object. For example:

```
status,code,msg, = call db-set-cursor(cursor-obj, bind-vars, myIO);
```

For complete details on using this procedure, see [db-set-cursor](#).

## Returning Query Data in Query Items

G2-Database bridges enable G2 applications to perform queries that return results by creating a new query item or updating an existing query item. G2 writes the query data to attributes of these query items, which are instances of the class `db-query-item-array` or `db-query-item-list`. An advantage of using query items is that they provide a means for obtaining database data without building database schema information into your application. In other words, attributes of query items are not required to be mapped to column names of database tables.

**To query a database and return the data to G2 in a *new* query item:**

→ Call `db-fetch-query-item`.

This procedure references an existing cursor object and executes the query defined by that cursor object, and then returns the resulting data within a newly created query item object. For example:

```
query-item,status,code,msg,nrows,cursor-pos =  
call db-fetch-query-item(cursor-obj, return-format, batch-size, myIO);
```

For complete details on using this procedure see [db-fetch-query-item](#).

**To query a database and update an *existing* query item in G2:**

→ Call `db-update-query-item`.

This procedure references an existing cursor object and executes the query defined by that cursor object, and then updates the data within an existing query item with the results from the query. For example:

```
status,code,msg,nrows,cursor-pos =  
call db-update-query-item(cursor-obj, query-item, update-action,  
batch-size, myIO);
```

For complete details on using this procedure, see [db-update-query-item](#).

## Returning Query Data to Existing G2 Items

G2-Database bridges enable G2 applications to perform queries that return results to G2 items. These items can be instances of virtually any user-defined G2 class.

### Returning Query Data to a User-Defined Object

**To query a database, returning the data to G2 in a *new* user-defined G2 item:**

- 1 Define the G2 item to which you want to return the query data. The attributes of the item must map to the column names of the database table(s) from which you will be querying.

For example, if you want to fetch from a database table that contains a column named `last_name` and defined as a `varchar(30)`, then you must define an attribute in your G2 item named `last_name` of type `text`. Information for mapping G2 data types to database data types can be found in [Appendix A, Bridge Data Types](#).

- 2 Define a cursor object for the query as described in [Creating a Cursor Object](#).
- 3 Call `db-fetch-object`.

This procedure queries the database and returns the query data *within* a user-defined G2 item of the type that you specify. For example:

```
myItem,status,code,msg,nrows,cursor-pos
= call db-fetch-object(cursor-obj, myOBJ, return-format,
batch-size, myIO);
```

For complete details on using this procedure see [db-fetch-object](#).

**To query a database, *updating* an existing user-defined G2 item:**

- 1 Define a cursor object for the query as described in [Creating a Cursor Object](#).
- 2 Call `db-update-object`.

This procedure queries the database and updates the query data in an existing user-defined G2 item that you specify. For example:

```
myItem,status,code,msg,nrows,cursor-pos
= call db-update-object(cursor-obj, myOBJ, update-action, batch-size,
myIO);
```

For complete details on using this procedure, see [db-update-object](#).

## Returning Query Data to Query Objects

You can perform queries that return query data to instances of predefined G2 query object classes. This feature is provided for backward compatibility for applications built with earlier versions of G2-Database. For information on creating and using query objects to perform database queries, see [Query Objects](#).

## Returning Query Data in Structures

You can perform queries that return query data within a G2 structure. G2 structures are considered composite value types. Composite types are those that are composed of one or more values of any general, specific, or composite type.

Structures consist of one or more pairs of names and values. The G2-Database bridge returns structures to G2 such that the names in the structure represent database column names and the values in the structure represent the values or data for the database column. Values are represented as G2 sequences which are themselves composite types. A sequence is a list-like value that can contain any value of any data type.

Since structures are represented as values in G2, there is no need to worry about memory leaks. If you query information from a database within a structure, you do not need to delete the structure. G2 will automatically reclaim any memory associated with the structure when it is no longer needed.

## Copying Query Item Attribute Values

You can copy attribute values of a query item array or list object to a specified G2 object, using the `db-copy` method. The `db-copy` method copies all attribute values of the query item array or list object to a target object. The method compares the attributes of the target object with the query item and only copies values for attributes that exist in the target object.

The syntax of this method is:

```
db-copy(source:class db-query-item-array, target:class object)
```

or

```
db-copy(source:class db-query-item-list, target:class object)
```

Argument	Description
<i>source</i>	The <code>db-query-item-array</code> or <code>db-query-item-list</code> whose attributes you want to copy.
<i>target</i>	The G2 object to which you copy the <code>db-query-item-array</code> or the <code>db-query-item-list</code> attribute values.

## Deleting Query Items

To delete a query item array or list object, always use the `delete-query-item` user menu choice of the object. This user menu choice invokes the `db-delete` method, which deletes not only the object, but also all items contained in attributes of the object. You can also invoke this method programmatically.

---

**Caution** If you use some way other than the user menu choice or `delete-query-item` to delete a query item array or list object, the items contained in attributes of the object may remain in your G2 KB after you delete the object. This will result in a memory leak.

---

## Using Smart Fetch

By default, to avoid G2 integer overflow problems when the retrieved value is too large, the database bridge “fetch” procedures return floats under these conditions:

- When fetching DECIMAL types when:
  - The precision is 9 or greater.
  - The precision is 0.
  - The scale is non-zero.
- When any of the returned INTEGER values would cause integer overflow in G2.

Note that in Oracle, the INTEGER type is defined as DECIMAL(38,0); thus, the Oracle INTEGER type always returns a float, by default.

You can use the “smart fetch” feature to better handle the case when fetching INTEGER or DECIMAL types might cause integer overflow in G2.

To use smart fetch, add `-F` to the initialization string of any interface object that should use the smart fetch behavior. For details, see `remote-process-initialization-string` in [Attributes of G2-Database-Interface Objects](#).

When smart fetch is enabled and when retrieving INTEGER or DECIMAL types from the database, in general, the fetch procedures return values as integers, or as floats if returning them as an integer would cause G2 integer overflow. For details, see the description of each of the following fetch procedures:

- [db-fetch-object](#)
- [db-fetch-query-item](#)
- [db-fetch-records](#)
- [db-fetch-structure](#)
- [db-update-object](#)
- [db-update-query-item](#)

# Query Objects

---

*Describes how to use query objects to query a database.*

Introduction	65
Creating a Query Object Class Definition	66
Creating a Query Object	71
Creating a Cursor Object	75
Performing the Query	75



## Introduction

To remain compatible with earlier versions of database bridges, the G2-Database Bridge software supports the use of query objects for performing database queries. In general, any operation that can be performed on a query object can be performed on any user defined item.

### **To perform database queries:**

- 1 [Create a class definition for the query object.](#)
- 2 [Create an instance of the query object.](#)
- 3 [Create a cursor object that defines the query.](#)
- 4 [Perform the query.](#)

This procedure executes the query defined by the cursor object and returns the query data directly to the attributes of the query object.

# Creating a Query Object Class Definition

You create a query object by defining a subclass of one of the following classes:

- db-query-object
- db-qo-record
- db-qo-table
- Any G2 variable class

The class definitions are on the G2 Database SQL Object Classes workspace.

The subclasses that you define inherit the icons from their parent classes (db-qo-table or db-qo-record). You can edit these icons or create new ones. For information on editing and creating icons, see the *G2 Reference Manual*.

## To create a new subclass of query objects:

- 1 Create an object definition on your workspace by selecting:

KB Workspace > New Definition > class-definition > object-definition

A triangular icon representing the object definition appears on the workspace.

- 2 Choose **table** from the object definition menu to display its table.

For example:

an object-definition	
Notes	INCOMPLETE, and note that (1) no class name is specified; (2) no direct superior classes are specified
Authors	none
Change log	0 entries
Class name	none
Direct superior classes	none
Class specific attributes	none
Instance configuration	none
Change	none
Instantiate	yes
Include in menus	yes
Class inheritance path	none
Inherited attributes	none
Attribute initializations	none
Icon description	inherited
Attribute displays	inherited
Stubs	inherited

The following table summarizes the attributes that you need to complete:

Attribute	Description
<b>class-name</b>	The name of this class of query object.
<i>Allowable values:</i>	A symbol
<i>Default value:</i>	none
<i>Notes:</i>	For example, <code>employee-record</code> might name a query object that stores a record of data relating to a particular employee.
<b>direct-superior-classes</b>	The name of one or more direct superior classes.
<i>Allowable values:</i>	db-query-object db-qo-record db-qo-table gsi-data-service <i>and</i> a G2 class that inherits from the g2-variable class
<i>Default value:</i>	none
<i>Notes:</i>	See <a href="#">Specifying Direct Superior Classes</a> .
<b>class-specific-attributes</b>	The column attributes that correspond to columns in the database.
<i>Allowable values:</i>	Any symbol that is not the name of a system-defined attribute
<i>Default value:</i>	none
<i>Notes:</i>	A query cannot return a column of data to a query object that has no attribute corresponding to the column. However, not all column attributes need to receive data from a particular query. See <a href="#">Specifying Column Attributes</a> .

Attribute	Description
<b>attribute-initializations</b>	Sets the initial values for user-defined attributes and certain system-defined variable attributes.
<i>Allowable values:</i>	If you create the query object class definition as a <b>g2-variable</b> class, you <i>must</i> specify an initial value for the G2-variable, based on its type.
<i>Default value:</i>	none
<i>Notes:</i>	For example, if the class definition inherits from the <b>symbolic-variable</b> class, you might specify the initial value as:  initial value for symbolic variable: OK  If you do not specify an attribute initialization value for a <b>g2-variable</b> class, G2 schedules the query object for an update at a fixed interval, resulting in unnecessary network traffic.

The values of all other attributes are provided by G2. For information about these attributes, see the *G2 Reference Manual*.

## Specifying Direct Superior Classes

For the **direct-superior-classes** attribute, you can specify either of the following:

- One of the predefined object classes **db-query-object**, **db-qo-record**, or **db-qo-table**.
  - If you intend to query the database for single records, specify **db-qo-record**.
  - If you intend to query the database for records in lists, specify **db-qo-table**.
- The G2 mixin class **gsi-data-service** *and* one of the G2 classes that inherit from the **g2-variable** class, such as **text-variable** or **symbolic-variable**.

If you specify **gsi-data-service** and a **g2-variable** class as the direct superior classes, you must also specify the following values for the **attribute-initializations** attribute:

- An initial value for the variable
- indefinite for validity interval
- none for default update interval

For example, assuming the `direct-superior-classes` attribute equals:

```
symbolic-variable, gsi-data-service
```

you could set the `attribute-initializations` attribute to:

```
initial value for symbolic-variable:GSI;
validity interval:indefinite;
default update interval:none
```

## Specifying Column Attributes

In each class definition, you define column attributes that correspond to the database columns from which you want to fetch data, using the `class-specific-attributes` attribute. A query object created from the object definition:

- Queries the database for the data in those columns.
- Stores the returned values in the attributes that correspond to the columns.

Each attribute must have the same name as the corresponding column in the database. If the database supports column-name aliases in SQL Select statements, you can use aliases as names of column attributes.

---

**Note** You *must* use an alias for any column name that is the same as a G2 reserved symbol. For a list of G2 reserved symbols, see the *G2 Reference Manual*.

---

You can define a column attribute as:

- A simple attribute.
- A g2-parameter.
- A g2-list.

### Defining a Simple Attribute

A simple column attribute holds a single value. Each new record fetched into the attribute replaces the previous record.

When defining simple attributes, consider the following:

- The query object should be a subclass of `db-qo-record`.
- Simple column attributes retain their values even if you reset G2 or disable the query object.
- A simple column attribute has no type specification, and G2 does not perform type checking on it.

Here is an example of **class-specific-attributes** for a query object named `employee-record` that uses simple attributes:

```
empname;  
empid;  
deptname;  
salary
```

### Defining Attributes as Parameters

A column attribute defined as a parameter holds a single value. The column data from each newly fetched record replaces the previous content of the parameter. The value of a parameter column attribute is lost if you disable the query object or reset G2.

When defining attributes as parameters, consider the following:

- The query object should be a subclass of `db-qo-record`.
- You must specify a bridge data type for each parameter column attribute. See [Appendix A, Bridge Data Types](#) for a list of database data types and the corresponding bridge data types that you must use.

For example, if the database stores employee names as `varchar` values in a column named `empname`, you can configure this column as a **text-parameter**, because text is the bridge data type that corresponds to `varchar`:

```
empname is given by a text-parameter
```

Here is an example of **class-specific-attributes** for a query object named `employee-record` that uses parameters:

```
empname is given by a text-parameter,  
initially is given by a text-parameter;  
empid is given by an integer-parameter,  
initially is given by an integer-parameter;  
deptname is given by a text-parameter,  
initially is given by a text-parameter;  
salary is given by a float-parameter,  
initially is given by a float-parameter
```

Notice that a semi-colon separates the column attributes.

### Defining an Attribute as a List

A column attribute defined as a list can contain more than one value. The column data from each newly fetched record is appended to the end of each corresponding attribute list.

If you are defining an attribute as a list, consider the following:

- The query object should be a subclass of `db-qo-table`.
- Use a `g2-list` of the type `text-list`, `integer-list`, `float-list`, or `quantity-list`.
- You must specify a bridge data type for each list column attribute. See [Appendix A, Bridge Data Types](#) for a list of database data types and the corresponding bridge data types that you must use.

For example, if the database stores employee names as `varchar` and you want to store employee names in a list, you can configure the column attribute as a `text-list`:

`empname` is an instance of a `text-list`

## Creating a Query Object

When you complete the object definition for your query objects, you can create instances of query object that are based on that definition.

### To create an instance of a query object definition:

- 1 Choose `create instance` from the query object definition menu.

An icon representing the new query object appears on top of the object definition.

- 2 Choose `table` from query object menu to display the attribute table.

For example:

an employee-record	
Notes	OK
Names	none
Data server	GSI data server
GSI interface name	none
GSI variable status	0
Db status	g2
Db code	0
Db message	""
Db rows processed	0
Db cursor position	0
Empname	none
Empid	none
Deptname	none
Salary	none

The following table describes the attributes of the query object, whose values you set, or the G2-Database bridge sets to indicate the result of the query:

Attribute	Description
<b>names</b>	The name of the query object.
<i>Allowable values:</i>	A symbol
<i>Default value:</i>	none
<i>Notes:</i>	These names are referenced by procedures that access the query object.  For information about these procedures, see <a href="#">Bridge Procedures</a> .

Attribute	Description
<b>gsi-interface-name</b>	The g2-database-interface object through which this query object communicates with the database.
<i>Allowable values:</i>	The name in the names attribute of the interface object
<i>Default value:</i>	none
<b>db-cursor-position</b>	Displays the relative position within the database cursor.
<i>Allowable values:</i>	An integer
<i>Default value:</i>	0
<b>db-status</b>	Specifies the status of the result of the transaction.
<i>Allowable values:</i>	<p><b>success:</b> The last operation performed on this query object was completed successfully</p> <p><b>error:</b> A database, SQL, or bridge error</p> <p><b>warning:</b> Warnings from database or bridge</p> <p><b>info:</b> Informational messages from bridge</p> <p><b>ecursor:</b> The last row of a cursor has been fetched. To fetch again, call db-refresh-cursor.</p> <p><b>fatal:</b> Non-recoverable error; connection is broken</p>
<i>Default value:</i>	g2
<b>db-code</b>	A number that corresponds to the values returned to db-status by the database or the G2-Database Bridge.
<i>Allowable values:</i>	An integer
<i>Default value:</i>	0
<i>Notes:</i>	For information about the status codes, see <a href="#">Appendix B, Status Values.</a>

Attribute	Description
<b>db-message</b>	Displays error, warning, or information messages resulting from a query.
<i>Allowable values:</i>	A text string
<i>Default value:</i>	""
<i>Notes:</i>	<p>An error message is reported when the query did not complete successfully.</p> <p>A warning message means the query completed successfully, but there is additional information. For example, the last record in the cursor has been fetched, or a column was truncated. If there is more than one warning message, the messages are combined into one text message. For information about the status codes, see <a href="#">Appendix B, Status Values</a>.</p>
<b>db-rows-processed</b>	Displays the number of rows processed by the query as reported by the database during the fetch operation.
<i>Allowable values:</i>	An integer
<i>Default value:</i>	0

The column attributes are the attributes that you specified under **class-specific-attributes** when you created the object definition from which you created this query object. In the example, the column attributes are `empname`, `empid`, `deptname`, and `salary`.

---

**Note** The G2-Database bridge sets the `db-status`, `db-code`, `db-message`, and `db-rows-processed` attributes to the values returned by the database procedure `db-fetch-records`.

---

## Creating a Cursor Object

You must create a cursor object to define the query that you want to perform using the query object. For information on creating and using cursor objects, see [Creating a Cursor Object](#).

## Performing the Query

**To query data using query objects:**

- Call `db-fetch-records` to fetch a specified number of rows (records) from a database cursor and return them to a specified query object in G2.

It can be useful for your G2 application to fetch the records in several small batches, rather than all at once. For example, if there are 2000 rows of data in a result cursor, you can choose to call `db-fetch-records` repeatedly, fetching only 100 at a time. The *rcds-to-fetch* argument of `db-fetch-records` specifies the number of records (rows) that a single execution of the procedure fetches from the database cursor.

For more information on using this procedure, see [db-fetch-records](#).

**To set the pointer in a database cursor back to the beginning of the cursor.**

- Call `db-refresh-cursor`.

You can then call `db-fetch-records` to start fetching data from the beginning of the cursor.

For information on using `db-refresh-cursor`, see [db-refresh-cursor](#).



# Bridge Procedures

---

*Describes how to invoke the API procedures and provides a detailed description of each procedure, its arguments, and usage.*

Introduction	78
Summary of G2-Database Bridge Procedures	78
Invoking G2-Database Bridge Procedures	83
Procedure Descriptions	85
db-commit	86
db-configuration	88
db-connect	90
db-context-event-msg	92
db-define-cursor	94
db-define-sql	98
db-define-sql-obj	101
db-disable-all-triggers	104
db-disconnect	106
db-exec-sql	108
db-exec-sql-obj	110
db-exec-stored-proc	112
db-exec-stored-proc-return	114
db-execute-immediate	118
db-fetch-object	120
db-fetch-query-item	125
db-fetch-records	129
db-fetch-structure	132
db-get-triggers	135
db-io-status	137
db-kill-bridge	138
db-logfile	139
db-ping	142
db-redirect-callback	143
db-refresh-cursor	146
db-rollback	148

db-set-connection-status 150  
db-set-cursor 151  
db-set-sql 153  
db-set-sql-obj 155  
db-set-trigger 158  
db-sql-function 160  
db-startup 163  
db-text-to-text-list 164  
db-trigger-event 165  
db-update-object 167  
db-update-query-item 171



## Introduction

Each G2-Database bridge provides a set of procedures that your G2 KB can invoke to perform database transactions. The database bridge procedures can execute SQL statements and perform other operations.

## Summary of G2-Database Bridge Procedures

The following tables summarize the G2-Database bridge procedures.

---

**Note** To permit future upgrades and patches, we do not recommend that you modify any components of the `g2-database.kb`. See the procedure descriptions for details.

---

## Connection and Initialization

Procedure	Description
<a href="#">db-configuration</a>	Called by db-startup to send configuration information for a specified context to the bridge.
<a href="#">db-connect</a>	Establishes a connection between a bridge and a database for use by a specified context.
<a href="#">db-disconnect</a>	Disconnects a bridge context from a currently connected database.
<a href="#">db-kill-bridge</a>	Terminates the bridge process.
<a href="#">db-ping</a>	Returns the status of the connection between a specified bridge context and the database.
<a href="#">db-startup</a>	Calls db-configuration and db-connect to configure and establish a connection between G2 and a G2-Database bridge.

## SQL Operations

Procedure	Description
<a href="#">db-commit</a>	Commits (makes permanent) all changes made since the last commit to your database.
<a href="#">db-define-sql</a>	Creates a re-usable SQL object for use in non-query database operations. The object specifies an SQL statement that can include bind variables. See also db-set-sql and db-exec-sql.
<a href="#">db-define-sql-obj</a>	Creates a re-usable SQL object for use in non-query database operations. The object specifies an SQL statement that <i>must</i> include bind variables. Values for these bind variables must be provided by attributes of a user-defined G2 object. See also db-set-sql-obj and db-exec-sql-obj.

<b>Procedure</b>	<b>Description</b>
<a href="#"><u>db-exec-sql</u></a>	Executes a non-query SQL operation, using an SQL object created by <code>db-define-sql</code> . Sends to the bridge an SQL statement that can include bind variables and their values.
<a href="#"><u>db-exec-sql-obj</u></a>	Executes a non-query SQL operation, using an SQL object created by <code>db-define-sql-obj</code> . Passes to the bridge an SQL statement that includes bind variables, and a user-defined G2 object that contains values for the bind variables.
<a href="#"><u>db-exec-stored-proc</u></a>	Executes a procedure stored in the database.
<a href="#"><u>db-exec-stored-proc-return</u></a>	Executes a procedure stored in the database that has a return value.
<a href="#"><u>db-execute-immediate</u></a>	Executes a non-query SQL statement that neither contains bind variables nor uses an SQL object.
<a href="#"><u>db-rollback</u></a>	Cancels (undoes) all changes made in a specified context since the last commit (save) to your database.
<a href="#"><u>db-set-sql</u></a>	Sets or changes the values of bind variables in an SQL object created by <code>db-define-sql</code> .
<a href="#"><u>db-set-sql-obj</u></a>	Specifies names of bind variables in an SQL object created by <code>db-define-sql-obj</code> and provides a user-defined G2 object whose attributes contain values for these bind variables.
<a href="#"><u>db-sql-function</u></a>	Used with a select statement to return a single value result from a query or SQL function, such as <code>SUM</code> , <code>MAX</code> , <code>MIN</code> , <code>AVG</code> , or <code>COUNT</code> .

## Query Operations

Procedure	Description
<a href="#"><u>db-define-cursor</u></a>	Creates a reusable cursor object that defines a query and provides a reference to the database cursor for that query.
<a href="#"><u>db-fetch-object</u></a>	Performs a query and returns the result to G2 in a user-defined G2 object.
<a href="#"><u>db-fetch-query-item</u></a>	Performs a query and returns the result to G2 in a query item.
<a href="#"><u>db-fetch-records</u></a>	Fetches a specified number of rows (records) from a database cursor and returns them to a specified Query Object in G2.
<a href="#"><u>db-fetch-structure</u></a>	Performs a query and returns the result to G2 in a structure.
<a href="#"><u>db-refresh-cursor</u></a>	Refreshes the cursor data and repositions the cursor pointer to the first record in the database cursor.
<a href="#"><u>db-set-cursor</u></a>	Sets or changes the values of bind variables in an existing cursor object.
<a href="#"><u>db-update-object</u></a>	Performs a query and updates an existing G2 object with the results. The update can either replace the values within the G2 object or append to the values within a G2 object.
<a href="#"><u>db-update-query-item</u></a>	Performs a query and updates an existing query item with the results. The update can either replace the values within the query item or append to the values within a query item.

## Error and Message Handling

Procedure	Description
<a href="#">db-context-event-msg</a>	Receives messages from the bridge about events that occur during bridge operation. You can modify this procedure to perform customized message handling. See <a href="#">Message Handling</a> .
<a href="#">db-disable-all-triggers</a>	Disables every trigger watch for every context.
<a href="#">db-get-triggers</a>	Returns a list of currently active trigger watches for every context.
<a href="#">db-logfile</a>	Opens, closes, or reopens a log file, or modifies logfile filters. See <a href="#">Saving Messages in Log Files</a> .
<a href="#">db-redirect-callback</a>	Allows you to redirect messages that would normally be sent to <code>db-context-event-msg</code> and <code>db-trigger-event</code> to any G2 procedure.
<a href="#">db-set-trigger</a>	Enables or disables a trigger watch on a specified trigger name.
<a href="#">db-trigger-event</a>	Called by the bridge when a trigger event specified by <code>db-set-trigger</code> occurs in the database. Sends a trigger message to G2.

## Methods and Utilities

Procedure	Description
<a href="#">db-copy</a>	Copies a <code>db-query-item</code> object and all items contained in the object. For information about this method, see <a href="#">Copying Query Item Attribute Values</a> .
<a href="#">db-delete</a>	Deletes a <code>db-query-item</code> object and all items contained in attributes of that object. For information about this method, see <a href="#">Deleting Query Items</a> .

Procedure	Description
<a href="#">db-io-status</a>	Changes the colors of different regions of the G2-Database Interface icon when the status of the connection between G2 and the G2-Database bridge changes. For information about how to modify this procedure, see <a href="#">Displaying the Connection Status</a> .
<a href="#">db-set-connection-status</a>	Sets the icon colors and <code>db-connection-status</code> attribute for <code>g2-database-interface</code> objects. This procedure is called from most G2-Database procedures upon returning from a transaction.
<a href="#">db-text-to-text-list</a>	Converts a comma delimited text string to a text-list.

## Invoking G2-Database Bridge Procedures

You can call G2-Database bridge procedures from within G2 procedures, or start them using G2 rules, action buttons, or user-menu choices:

- Within user-defined G2 procedures, you invoke G2-Database bridge procedures using the `call` procedure statement. API functions invoked through `call` procedure statements can return values to the G2 procedures.
- Within G2 rules, action buttons, or user menu choices, you invoke API functions using the G2 `start` action. API functions invoked through a `start` action do not return values.

### Invoking a Bridge Procedure from within a G2 Procedure

To invoke a G2-Database bridge procedure from within a user-defined G2 procedure, you must use the `call` procedure statement, with this syntax:

```
return-argument [ , return-argument ] . . . = call procedure-name(arg1, arg2,...);
```

Most G2-Database bridge procedures are defined to return values to the status arguments *status*, *code*, and *message*. Some G2-Database bridge procedures return other values, or objects.

## Status Values Returned by G2-Database Bridge Procedures

The following table describes the *status*, *code*, and *message* values returned by most G2-Database bridge procedures.

Argument	Type	Description
<i>status</i>	symbol	The status of the operation performed by the API function. The possible values are:  SUCCESS ERROR, WARNING INFO DISCONNECTED FATAL EOCURSOR
<i>code</i>	integer	The error or information code.
<i>message</i>	text	The text of the message associated with the <i>code</i> value.

## Omitting Return Arguments from Calls to G2-Database Bridge Procedures

You can omit return arguments from a call to a G2-Database bridge procedure, if you preserve the order of the return arguments as shown in the procedure descriptions. G2-Database bridges return values to the return arguments in the order shown in the descriptions.

For example, the following examples of syntax are all valid, because the return arguments are listed in the order shown in the procedure descriptions (*status*, *code*, *message*):

```
status, code, message = call procedure-name(arg1, arg2,...);  
status, code = call procedure-name(arg1, arg2,...);  
status = call procedure-name(arg1, arg2,...);
```

However, a procedure call using the following syntax fails, because the bridge attempts to return a symbol (the *status* value) to the first and only return argument, which is a text variable (*message*):

```
message = call procedure-name(arg1, arg2,...);
```

## Invoking a Bridge Procedure from a Rule, Action-Button, or User-Menu-Choice

To invoke a bridge G2-Database bridge procedure from within a G2 rule, action-button, or user-menu-choice, you must use the G2 `start` action. For example:

```
start procedure-name (arg1, arg2, ...);
```

G2 does not return values to procedures invoked by the `start` action.

# Procedure Descriptions

The rest of this chapter provides detailed a description of each G2-Database bridge procedure. The descriptions are presented in alphabetical order.

[db-commit](#)  
[db-configuration](#)  
[db-connect](#)  
[db-context-event-msg](#)  
[db-define-cursor](#)  
[db-define-sql](#)  
[db-define-sql-obj](#)  
[db-disable-all-triggers](#)  
[db-disconnect](#)  
[db-exec-sql](#)  
[db-exec-sql-obj](#)  
[db-exec-stored-proc](#)  
[db-exec-stored-proc-return](#)  
[db-execute-immediate](#)  
[db-fetch-object](#)  
[db-fetch-query-item](#)  
[db-fetch-records](#)  
[db-fetch-structure](#)  
[db-get-triggers](#)  
[db-io-status](#)  
[db-kill-bridge](#)  
[db-logfile](#)  
[db-ping](#)  
[db-redirect-callback](#)  
[db-refresh-cursor](#)  
[db-rollback](#)  
[db-set-connection-status](#)  
[db-set-cursor](#)  
[db-set-sql](#)  
[db-set-sql-obj](#)  
[db-set-trigger](#)  
[db-sql-function](#)  
[db-startup](#)  
[db-text-to-text-list](#)  
[db-trigger-event](#)  
[db-update-object](#)  
[db-update-query-item](#)

# db-commit

Commits all changes made since the last commit (save) to your database.

## Synopsis

```
db-commit
(interface:class g2-database-interface)
-> status, code, message
```

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<u>status</u> , <u>code</u> , <u>message</u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

Use `db-commit` to send a commit (save) statement to the database. When the commit is issued, all transactions, such as inserts, deletes, updates, performed on the database through this context since the last commit, are saved and all transactions are ended.

Committing changes to the database makes the changes permanent. Until outstanding changes are committed, other users or contexts may not be able to access the changed data. In addition, resources allocated by the database may be in a *locked* state as a result of your transaction. The commit will release and *unlock* these resources.

The commit statement is issued for the context identified by the G2-Database interface object that you specify in the *interface* argument.

## Related Procedures

[db-rollback](#) Cancels (or undoes) all changes made in a specified context since the last commit to your database.

---

**Note** The actual behavior of `db-commit` may vary depending upon the database being used. Consult the Release Notes and the database specific documentation for details on using `db-commit`.

---

## Example

The following call to `db-commit` sends a commit statement to the bridge for the context configured by the G2-Database interface object `db1-interface`.

```
status: symbol;  
code: integer;  
msg: text;
```

```
status, code, msg = call db-commit(DB1-INTERFACE);
```

The following call to `db-commit` makes the transaction submitted by `db-execute-immediate` permanent in the database. `db-execute-immediate` is called with `auto-commit` set to `false`, and thus does not itself perform a commit.

```
status: symbol;  
code: integer;  
msg: text;  
rows-processed: integer;  
sql-stmt: text = "insert into emp (ename) values ('Smith')";  
auto-commit: truth-value = FALSE;
```

```
...
```

```
status,code,msg,rows-processed = call db-execute-immediate(sql-stmt,  
    auto-commit, DB1-INTERFACE);  
status,code,msg = call db-commit(DB1-INTERFACE);
```

# db-configuration

Called by db-startup to send configuration information for a specified context to the bridge.

## Synopsis

db-configuration  
 (*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure db-configuration sends configuration information to the bridge. The configuration information applies only to the context identified by the G2-Database interface object specified in the *interface* argument.

Do not attempt to execute this procedure after a connection has been established between G2 and the bridge. Attempting to do so results in an error, and the bridge displays the message: **Context has already been configured**. In order to change the configuration of a connection, you must disable and then enable a G2-Database interface object, or restart your G2 knowledge base.

---

**Note** Recommended practice is to call db-configuration from within db-startup. This is done automatically by a rule located on the subworkspace of the g2-database-interface class definition. db-configuration must only be called once while G2 is connected to the bridge through a given context.

---

The configuration information that this procedure sends is contained in attributes of the G2-Database interface object specified in the *interface* argument. For information about these attributes, see [Attributes of G2-Database-Interface Objects](#).

---

**Note** Before you call `db-configuration`, you must create and set the attributes of the G2-Database interface object that you specify in the *interface* argument.

---

# db-connect

Establishes a connection between a bridge and a database for use by a specified context.

## Synopsis

db-connect  
 (*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

This procedure sends database connection information to the bridge and establishes a connection between the bridge and the database. The connection between the bridge and G2 must be established and configured.

You can call this procedure at any time while G2 is connected to the bridge to establish a new connection to a database, or to re-establish a connection that has been broken or disconnected.

If a connection already exists between the bridge and the database when you call db-connect, the procedure returns the warning connection already established.

The information that this procedure sends to the bridge is contained in attributes of the G2-Database interface object specified in the *interface* argument. These attributes are:

- database-user
- database-password
- database-connect-string

See [Attributes of G2-Database-Interface Objects](#) for information about these attributes.

## Related Procedures

[db-disconnect](#)

Disconnects a bridge context from a currently connected database.

## Example

The following call to `db-connect` establishes a connection between the bridge and the database for the context configured by the G2-Database interface object `db1-interface`:

```
status:symbol;  
code:integer;  
msg:text;
```

```
status, code, msg = call db-connect(db1-interface);
```

## db-context-event-msg

Reports the occurrence of significant events during the execution of the G2-Database bridge. This procedure may be used as a foundation procedure for developing a custom message handler.

### Synopsis

db-context-event-msg

(*status:symbol, code:integer, message:text, facility:symbol, timestamp:text*)

Argument	Description
<i>status</i>	The category of the message. The possible values are: FATAL, ERROR, WARNING, and INFO.
<i>code</i>	The message code. This is an integer greater than 9000 for WARNING and INFO messages, or an integer less than -9000 for bridge error messages. It can also be a database-specific error code.
<i>message</i>	The message associated with the message code that db-context-event-msg reports. The format of messages is described in the Description section below.
<i>facility</i>	The facility from which the message originates. Possible values:  INTERNAL: The message originates from the bridge.  EXTERNAL: The message originates in the database or another external system.
<i>timestamp</i>	The date and time when the reported event occurred.

### Description

If enabled, by setting the enable-messaging attribute of your g2-database-interface object to true, db-context-event-msg is called by the bridge to report the occurrence of significant events during the execution of the G2-Database bridge. When the bridge calls db-context-event-msg, it provides the procedure with values for all its arguments.

You must edit the definition of `db-context-event-msg` to enable it to manage event messages within your G2 knowledge base. You can do this by including calls to other procedures in the definition or adding other code to it.

The messages in the *message* argument have the following format:

*bridge-name : context-name : status : message-body*

The following table describes these fields:

<b>Message Field</b>	<b>Description</b>
<i>bridge-name</i>	The name of the G2-Database bridge, such as G2-Oracle or G2-Sybase.
<i>context-name</i>	The name specified in the <code>context-name</code> attribute of the G2-Database interface object. This name identifies the connection for which the message is reported.
<i>status</i>	A status value for the event reported. For information about the possible status values, see <a href="#">Appendix B, Status Values</a> .
<i>message-body</i>	The text of the message, describing the event for which the message is reported.

Messages can be up to 512 bytes in length.

### **Recommended Approach**

Alternatively, you can instruct the bridge to send messages to any user defined G2 procedure by calling `db-redirect-callback`. This is the recommended approach as it does not require you to modify `db-context-event-msg`. For information about `db-redirect-callback` see [db-redirect-callback](#).

### **Related Procedures**

[db-redirect-callback](#) Allows you to redirect messages that would normally be sent to `db-context-event-msg` to any G2 procedure.

## db-define-cursor

Creates a re-usable cursor object that defines a query and provides a reference to the database cursor for that query.

### Synopsis

db-define-cursor

(*cursor-object-name*:symbol, *sql-statement*:text,  
*bind-variables*:text-list or text, *interface*: class g2-database-interface)  
 -> *cursor-object*:class db-cursor-object, *status*, *code*, *message*

Argument	Description
<i>cursor-object-name</i>	The name of a cursor object. If you specify the name of an existing cursor object, this procedure redefines that object. If no cursor object named <i>cursor-object-name</i> exists, this procedure creates a cursor object with that name.
<i>sql-statement</i>	The SQL query statement. The SQL statement can contain bind variables.  The syntax of the SQL statement is specific to the database. For information about the syntax to use, see the release notes for your G2-Database bridge or the documentation for the database.
<i>bind-variables</i>	Values for the bind variables in the query SQL statement. Bind variables must be supplied as a text-list. If you prefer, you can specify bind values as a text string with each value separated by a comma. Note: This approach is not as efficient as providing a text-list.  If you do not want to assign a value to a bind variable from this call to <b>db-define-cursor</b> , specify "" for <i>bind-variables</i> . You must then call <b>db-set-cursor</b> to specify a value(s) for the bind variable(s).
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<u><i>cursor-object</i></u>	The cursor object created or redefined by this procedure.
<u><i>status</i></u> , <u><i>code</i></u> , <u><i>message</i></u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

Creates a cursor object that:

- Contains an SQL statement that defines a query.
- Contains a list of the bind variables set.
- Provides a reference to a cursor in the database.

The procedure `db-define-cursor` does not perform queries. To perform the query defined in the SQL statement, you pass the cursor object to `db-fetch-query-item`, `db-fetch-records`, `db-fetch-object`, `db-update-query-item`, `db-update-object`, or `db-fetch-structure`. You can use the same cursor object to perform any number of different queries. The cursor object may be reused.

To change the SQL statement or bind variables in an existing cursor object, you can call `db-define-cursor`, using the name of the existing object as the *cursor-object-name* argument and specifying a new *sql-statement* or *bind-variables* argument.

To assign new values to bind variables in the SQL statement of a cursor object, call `db-set-cursor`.

The database cursor that is created by this procedure will not be deleted until the connection to the database has been terminated.

## Related Procedures

<a href="#"><u>db-set-cursor</u></a>	Sets or changes the values of bind variables in an existing cursor object.
<a href="#"><u>db-fetch-query-item</u></a>	Performs a query and returns the result to G2 in a query item.
<a href="#"><u>db-fetch-records</u></a>	Fetches a specified number of rows (records) from a database cursor and returns them directly to the attributes of an object in G2.

<a href="#">db-fetch-object</a>	Performs a query and returns the result to G2 in a user-defined G2 object.
db-fetch-structure	Performs a query and returns the results within a structure where each element of the structure is a name/sequence pair containing one column of database data.
db-update-query-item	Returns data associated with a database cursor to an existing query item in G2. The data within the query item may be either replaced or appended.
db-update-object	Returns data associated with a database cursor to an existing object in G2. The data within the object may be either replaced or appended.

## Example

The following call to `db-define-cursor` creates a new cursor object or modifies an existing one. This example does not provide values for the bind variables (:id and :j). For this reason, the call does not process the bind variables. Your G2 application must call `db-set-cursor` to set the bind variables before it can perform the query defined by the cursor object.

```

status: symbol;
code: integer;
msg: text;
cursor-obj-name: symbol = the symbol MY-CURSOR-OBJ;
cursor-obj: class db-cursor-object;
sql-stmt: text = "select ename from emp where empid = :id and job = :j";
bind-vars: text = "";

cursor-obj,status,code,msg = call db-define-cursor(cursor-obj-name, sql-stmt,
bind-vars, myIO);

```

The following example assigns values to the bind variables. Thus, the call to `db-define-cursor` can process the bind variables, and your G2 application does not need to call `db-set-cursor` unless you want to change the values of the bind variables.

```
status: symbol;  
code: integer;  
msg: text;  
cursor-obj-name: symbol = the symbol MY-CURSOR-OBJ;  
cursor-obj: class cursor-object;  
sql-stmt: text = "select ename from emp where empid = :id and job = :j";  
bind-vars: text-list;  
  
insert "557" at the end of bind-vars;  
insert "Salesman" at the end of bind-vars;  
cursor-obj,status,code,msg = call db-define-cursor(cursor-obj-name, sql-stmt,  
bind-vars, myIO);
```

The `bind-var` list must contain values in the same order as the corresponding bind variables in the `sql-stmt`. Thus, the first value (557) is assigned to the first bind variable (:id) and the second value (salesman) is assigned to the second bind variable (:j).

In the example above, you could also have assigned the bind variables to a text as indicated below. However, this is much less efficient:

```
bind-vars: text = "557,Salesman";
```

## db-define-sql

Creates an SQL object for use in DML (non-query) database operations where bind variables are represented as simple values.

### Synopsis

db-define-sql

(*sql-object-name*:symbol, *sql-statement*:text,  
*bind-variables*:text-list or text, *interface*: class g2-database-interface)  
 -> *sql-object*:class sql-object, *status*, *code*, *message*

Argument	Description
<i>sql-object-name</i>	The name of an SQL object. If you specify the name of an existing SQL object, this procedure redefines that object. If no SQL object named <i>sql-object-name</i> exists, this procedure creates an SQL object with that name.
<i>sql-statement</i>	The query SQL statement.
<i>bind-variables</i>	Values for the bind variables in the query SQL statement. Bind variables must be supplied as a text-list. If you prefer, you can specify bind values as a text string with each value separated by a comma. Note: This approach is not as efficient as providing a text-list.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>sql-object</i>	The SQL object that this procedure creates or redefines.
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

Enables you to perform DML operations involving simple values. It creates an SQL object containing an SQL statement that defines a non-query operation, such as: insert, delete, or update.

To perform the operation defined in the SQL statement, you pass the SQL object to `db-exec-sql`. You can use the same SQL object to perform any number of different SQL operations. You can also reuse the SQL object.

To change the SQL statement in an existing SQL object, you can call `db-define-sql`, specifying the name of the existing SQL object as the *sql-object-name* argument. In the call to `db-define-sql`, you specify the new SQL statement that you want to associate with the SQL object.

To assign new values to bind variables in the SQL statement, you can call `db-set-sql`.

The database cursor that is created by this procedure will not be deleted until the connection to the database has been terminated.

## Related Procedures

### [db-set-sql](#)

Sets or changes the values of bind variables in an SQL object created by `db-define-sql`.

### [db-exec-sql](#)

Executes a non-query database operation using an SQL object created by `db-define-sql`.

## Examples

**Example 1:** The following call to `db-define-sql` creates an SQL object or modifies an existing one. This example does not provide values for the bind variables (:n and :a). You must call `db-set-sql` to set the values of the bind variables before you can use this SQL object to perform the database operation defined by the SQL object.

```
status: symbol;
code: integer;
msg: text;
sql-obj-name: symbol = the symbol MY-SQL-OBJ;
sql-obj: class db-sql-object;
sql-stmt: text = "insert into emp (ename,age) values (:n,:a)";
bind-vars: text = "";
...
sql-obj,status,code,msg = call db-define-sql(sql-obj-name, sql-stmt,
bind-vars, myIO);
```

**Example 2:** The following call to `db-define-sql` creates an SQL object and assigns values to the bind variables. You do not need to call `db-set-sql` unless you want to change the values of the bind variables.

```

status: symbol;
code: integer;
msg: text;
sql-obj-name: symbol = the symbol MY-SQL-OBJ;
sql-obj: class db-sql-object;
sql-stmt: text = "insert into emp (ename,age) values (:n,:a)";
bind-vars: text-list;

insert "Smith" at the end of bind-vars;
insert "37" at the end of bind-vars;
sql-obj,status,code,msg = call db-define-sql(sql-obj-name, sql-stmt,
      bind-vars, myIO);

```

The `bind-var` list must contain values in the same order as the corresponding bind variables in the `sql-stmt`. Thus, the first value (`Smith`) is assigned to the first bind variable (`:n`) and the second value (`37`) is assigned to the second bind variable (`:a`).

In the example above, you could also have assigned the bind variables to a text as indicated below. However, this is much less efficient:

```

bind-vars: text = "Smith,37";

```

# db-define-sql-obj

Creates an SQL object for use in DML (non-query) database operations where bind variables will be supplied within an object.

## Synopsis

db-define-sql-obj

(*sql-object-name*:symbol, *sql-statement*:text,  
*interface*:class g2-database-interface)  
-> *sql-object*:class db-sql-object, *status*, *code*, *message*

Argument	Return Value
<i>sql-object-name</i>	Specify a name for the SQL object that this procedure creates. If you specify the name of an existing SQL object, it is redefined.
<i>sql-statement</i>	The SQL statement that contains bind variables.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<i>sql-object</i>	The SQL object that this procedure creates or redefines.
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

This procedure enables you to perform DML operations on numerous database rows by specifying bind variables within a G2 object. It creates an SQL object containing an SQL statement that defines a non-query operation, such as insert, delete, or update.

Note that db-define-sql-obj does not assign values to bind variables in the SQL statement. To assign values to the bind variables, you must call db-set-sql-obj, which obtains values for the bind variables from attributes of a user-defined G2 object. The bind variable names must match the names of the attributes of the user-defined G2 object from which db-set-sql-obj extracts bind variable values.

It also does not execute the SQL statement defined in the SQL object.

To perform the database operation defined in the SQL statement, you pass the SQL object to `db-exec-sql-obj`. You can use the same SQL object for any number of separate database operations.

To change the SQL statement in an existing SQL object, you can call `db-define-sql-obj`, specifying the name of the existing SQL object as the *sql-object-name* argument. In the call to `db-define-sql-obj`, you specify the new SQL statement that you want to associate with the SQL object.

## Related Procedures

### [db-set-sql-obj](#)

Sets or changes the values of bind variables in an SQL object, by using attributes of a user-defined G2 object.

### [db-exec-sql-obj](#)

Executes a non-query database operation on lists or arrays of values.

## Example

The following call to `db-define-sql-obj` creates an SQL object named `my-sql-obj` or modifies an existing SQL object named `my-sql-obj`:

```
status: symbol;
code: integer;
msg: text;
sql-obj-name: symbol = the symbol MY-SQL-OBJ;
sql-obj: class sql-object;
sql-stmt: text = "insert into emp (ename,age) values (:n,:a)";
```

```
sql-obj,status,code,msg, = call db-define-sql-obj(sql-obj-name, sql-stmt,
myIO);
```

The SQL statement `sql-stmt` contains two bind variables, `:n` and `:a`, and associates these bind variables with the G2 object attribute names `ename` and `age`, respectively:

Bind Variable	Corresponding Object Attribute Name	Data Type of Attribute
<code>:n</code>	<code>ename</code>	text-list
<code>:a</code>	<code>age</code>	integer-list

To assign values to the bind variables, your G2 application must call `db-set-sql-obj`, which must reference a user-defined G2 object that has attributes named `ename` and `age`.

# db-disable-all-triggers

Disables all enabled trigger watches.

## Synopsis

db-disable-all-triggers  
(*interface*: class g2-database-interface)  
-> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures the connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

Disables every trigger watch for a bridge that was enabled by calling `db-set-trigger`. This procedure disables every trigger watch regardless of the context in which the trigger was set.

---

**Note** The procedure `db-disable-all-triggers` does not affect triggers within the database or within other bridges. It does cause the bridge to ignore any triggers that fire within the database.

---

If the connection between the bridge and the database is broken then all trigger watches defined in that bridge are automatically disabled.

See the Release Notes for your G2-Database bridge for more information about disabling and enabling triggers.

## Related Procedures

[db-set-trigger](#) Enables or disables a trigger watch.

[db-get-triggers](#) Returns a list of currently active trigger watches

# db-disconnect

Disconnects a bridge context from a database.

## Synopsis

```
db-disconnect
(interface: class g2-database-interface)
-> status, code, message
```

Argument	Description
<i>interface</i>	The G2-Database interface object that configures the connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure `db-disconnect` removes the connection between a bridge context and the database that was established by a call to `db-connect`. It does not break connections configured by other G2-Database Interface objects, or disconnect the bridge from G2.

It also cleans-up and frees all resources associated with the database connection of the context specified by *interface*.

You can disconnect your database from the bridge at any time. However, until the connection to the database is re-established, G2 cannot issue any requests to the database over that connection.

See the Release Notes for your G2-Database bridge for more information about connecting and disconnecting the bridge.

## Related Procedures

[db-connect](#) Establishes a connection between a bridge and a database for use by a specified context.

## Example

The following code calls `db-disconnect` to break the connection between a bridge context and the database that is configured by the G2-Database interface object `db1-interface`.

```
status: symbol;  
code: integer;  
msg: text;  
...  
status, code, msg = call db-disconnect (db1-interface);
```

# db-exec-sql

Executes a DML (non-query) database operation defined by `db-define-sql`.

## Synopsis

`db-exec-sql`

(*sql-object*:class db-sql-object, *auto-commit*:truth-value,  
*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*, *rows-processed*:integer

Argument	Description
<i>sql-object</i>	An existing SQL object that was created by a call to <code>db-define-sql</code> .
<i>auto-commit</i>	Specify <code>true</code> to cause the transaction to be committed automatically if the operation completes successfully.  Specify <code>false</code> to disable automatic committing. In this case, you must commit or rollback the transaction manually.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of rows effected in this transaction.

## Description

Executes a DML statement contained in the SQL object *sql-object*.

## Related Procedures

<a href="#">db-commit</a>	Commits all changes made since the last commit (save) to your database.
db-define-sql	Creates an SQL object for use in DML database operations on individual values.
db-rollback	Cancels all changes made in a specified context since the last commit (save) to your database.
db-set-sql	Sets or changes the values of bind variables in an SQL object used for operations on individual values.

## Example

The following call to `db-exec-sql` executes the SQL statement associated with the sql-object named `my-sql-obj`.

```
status: symbol;  
code: integer;  
msg: text;  
rows-processed: integer;  
sql-obj: class sql-object = my-sql-obj; {created by db-define-sql};  
auto-commit: truth-value = true;  
...  
status,code,msg,rows-processed = call db-exec-sql(sql-obj, auto-commit, myIO);
```

## db-exec-sql-obj

Executes a DML (non-query) database operation defined by `db-define-sql-obj`.

### Synopsis

`db-exec-sql-obj`

(*sql-object*:class db-sql-object, *auto-commit*:truth-value,  
*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*, *rows-processed*: integer

Argument	Description
<i>sql-object</i>	An existing SQL object that was created by a call to <code>db-define-sql-obj</code> .
<i>auto-commit</i>	Specify <code>true</code> to cause the transaction to be committed automatically if the operation completes successfully.  Specify <code>false</code> to disable automatic committing. In this case, you must commit or rollback the transaction manually.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of rows effected in this transaction.

### Description

The procedure `db-exec-sql-obj` performs DML database operations and uses a user-defined G2 object to supply values for bind variables. This lets you perform several database operations (such as a multi-row insert) in one transaction. It executes an SQL statement (specified in the SQL object) that includes bind variables which represent attributes of a user-defined G2 object.

To create an SQL object, call `db-define-sql-obj`.

To assign values to the bind variables in the SQL object, call `db-set-sql-obj`. This procedure sets the bind variables to the values of attributes in an existing user-defined G2 object.

## Related Procedures

[db-define-sql-obj](#) Creates or redefines an SQL object for use in non-query database operations that uses a G2 object to supply values for bind variables.

[db-set-sql-obj](#) Sets or changes the values of bind variables in an SQL object that uses a G2 object to supply values for bind variables.

## Example

The following call to `db-exec-sql-object` executes an SQL operation defined in the SQL object named `my-sql-obj`.

```
status: symbol;
code: integer;
msg: text;
rows-processed: integer;
sql-obj: class sql-object = my-sql-obj; {created by db-define-sql};
auto-commit: truth-value = true;

begin
  { create my-sql-obj with db-define-sql }
  . . .
  { bind the bind variables in the SQL statement in my-sql-obj }
  { to the attributes of a user defined object with db-set-sql-obj }
  . . .
  status, code, msg, rows-processed =
    call db-exec-sql-obj(my-sql-obj, auto-commit, myIO);
```

# db-exec-stored-proc

Executes a procedure stored in the database. Returns no data.

## Synopsis

db-exec-stored-proc

(*sql-stmt*:text, *interface*: class g2-database-interface)

-> *status*, *code*, *msg*, *rows-processed*:integer

Argument	Description
<i>sql-stmt</i>	Valid stored procedure call to a database-stored procedure, which does not return any values.  The syntax that you must use in the <i>sql-stmt</i> is specific to the database. For information about the syntax, see the release notes for your G2-Database bridge or the manual for your database.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>msg</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	Not used.

## Description

The procedure `db-exec-stored-proc` accepts any valid SQL call, to a database-stored procedure, as an argument, and sends the statement to the bridge for processing.

The stored database procedure must *not* attempt to return values. Consequently, `db-exec-stored-proc` returns no data, only status information.

For information on executing stored procedures with return values, see [db-exec-stored-proc-return](#).

## Example

The following call to `db-exec-stored-proc` executes the database stored procedure `add_name` across the interface object named `myIO`, with the single argument `"Gensym"`:

```
status: symbol;  
code: integer;  
msg: text;  
sql-stmt: text = "add_name ('Gensym')";  
  
status,code,msg = call db-exec-stored-proc(sql-stmt, myIO);
```

## db-exec-stored-proc-return

Calls a stored function via the user-defined stored procedure named `sp_handler`.

### Synopsis

db-exec-stored-proc-return

(*proc-name*: text, *args*: text, *interface*: class g2-database-interface)

-> *return-value*: text, *status*, *code*, *message*

Argument	Description
<i>proc-name</i>	A command to <code>sp_handler</code> . Typically, the <i>proc-name</i> is the name of the stored function that <code>sp_handler</code> should call. However, since you write the <code>sp_handler</code> stored procedure, you can use the <i>proc-name</i> argument as any type of command to <code>sp_handler</code> . For details, see the example below.
<i>args</i>	The arguments to the stored function as a text string. The text string can have a maximum length of 4,000 characters. The string contains one or more arguments to the stored function. The parsing and interpretation of this string is the responsibility of the stored procedure <code>sp_handler</code> .
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<u><i>return-values</i></u>	The value returned by the stored function, which can include up to 4,000 characters.
<u><i>status</i></u> , <u><i>code</i></u> , <u><i>msg</i></u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

To use this procedure, you must:

- Call `db-exec-stored-proc-return` from your G2 program, passing to it the name of the stored function to execute and the arguments to the stored function.
- Write a stored procedure called `sp_handler`.

The syntax of `sp_handler` is:

```
sp_handler(proc-name IN VARCHAR2, args IN VARCHAR2,  
          return-value OUT VARCHAR2) .
```

`sp_handler` uses *proc-name* to determine which stored function to call, performs any necessary decoding and data-type conversion of *args*, calls the requested stored function, converts the returned values to text, and stores the results in *return-value* before returning to G2.

---

**Note** The `db-exec-stored-proc-return` procedure does not work in the G2-Sybase Bridge. See the *G2-Sybase Bridge Release Notes* for details.

---

## Example

The following example uses the demo tables that are normally installed when you create a new database. This example shows how to:

- Define 2 stored functions:
  - `Lup`, which takes a department name as input and returns the associated department number and the city where it is located.
  - `CtSal`, which returns the number of employees who have a monthly salary equal to or greater than a specified number.
- Define `sp_handler` to use these functions.
- Call the stored functions from G2.

In your database, you might declare the stored functions named `Lup` and `CtSal`, as follows:

```
-- Return the department number and location of the named department  
-- =====  
CREATE OR REPLACE FUNCTION Lup (DeptName IN VARCHAR2) RETURN VARCHAR2  
IS  
    nrDept  NUMBER(2) ;  
    vcLoc   VARCHAR2(13) ;  
BEGIN  
    SELECT DeptNo, Loc INTO nrDept, vcLoc FROM Dept WHERE DName =  
           UPPER (DeptName) ;  
    RETURN TO_CHAR(nrDept) || ',' || vcLoc ;
```

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'X' ;
    WHEN TOO_MANY_ROWS THEN
        RETURN '>' ;
    WHEN OTHERS THEN
        RETURN 'E' ;
END Lup;
/

-- Return the number of employees who have a salary of TargSal or greater
-- =====
CREATE OR REPLACE FUNCTION CtSal (TargSal IN NUMBER) RETURN INTEGER
IS
    inRet INTEGER ;
BEGIN
    SELECT count(*) INTO inRet FROM Emp WHERE Sal >= TargSal ;
    RETURN inRet ;
EXCEPTION
    WHEN OTHERS THEN
        RETURN -1 ;
END CtSal;
/

```

In addition to declaring the stored functions, you must declare the `sp_handler` procedure to call the correct stored function and convert data types of return values and arguments, as follows:

```

-- Dispatch a call from db-exec-stored-proc-return to the correct
-- function
-- =====
CREATE OR REPLACE PROCEDURE sp_handler (FuncName IN VARCHAR2, Args IN
VARCHAR2, RetVal OUT VARCHAR2) IS
BEGIN
    IF LOWER(FuncName) = 'lup' THEN
        RetVal := Lup(Args) ;
    ELSE IF LOWER(FuncName) = 'ctsal' THEN
        RetVal := TO_CHAR(CtSal(TO_NUMBER(Args))) ;
    ELSE
        RetVal := '?' ;
    END IF ;
    END IF ;
EXCEPTION
    WHEN OTHERS THEN
        RetVal := 'X' ;
END ;
/

```

The following call to `db-exec-stored-proc-return` in G2 executes the database stored function named `Lup` across the interface object named `myIO`, with the single argument `"Accounting"`. The procedure returns the department number and location of the named department as a text string, which is `"10,New York"`.

```
status: symbol;  
code: integer;  
msg: text;  
return-value: text;
```

```
return-value,status,code,msg = call db-exec-stored-proc-return("Lup", "Accounting",  
myIO);
```

The following call in G2 executes the database stored function named `CtSal` across the interface object named `myIO`, with the single argument `"2000"`. The procedure returns the number of employees who have a monthly salary of the named salary or greater, as a text string, which is `"6"`.

```
status: symbol;  
code: integer;  
msg: text;  
return-value: text;
```

```
return-value,status,code,msg = call db-exec-stored-proc-return("CtSal", "2000",  
myIO);
```

## db-execute-immediate

Executes any SQL statement that does not contain bind variables or use an SQL object. Returns no data.

### Synopsis

db-execute-immediate

(*sql-stmt*:text, *auto-commit*:truth-value,  
*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*, *rows-processed*:integer

Argument	Description
<i>sql-stmt</i>	Specify any valid SQL statement.
<i>auto-commit</i>	Specify true to cause the transaction to be committed automatically if the operation completes successfully.  Specify false to disable automatic committing. In this case, you must commit or rollback the transaction manually.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of database rows effected by the transaction.

### Description

The procedure db-execute-immediate takes any SQL Data Definition Language (DDL) or SQL Data Manipulation Language (DML) statement as an argument, sends the statement to the bridge for processing, and returns status information.

`db-execute-immediate` is useful for executing SQL statements that you do not intend to execute repeatedly. This procedure does not declare database cursors. The SQL statement that it executes is not saved in the KB, and the database must devise an execution plan each time this procedure is executed.

Although `db-execute-immediate` does not return data, you can execute a `select` statement to test a database schema structure. For example, if you issue the SQL statement through a call to `db-execute-immediate`:

```
select invalid_column_name from emp
```

where *invalid\_column\_name* is in fact an invalid column name, the call to `db-execute-immediate` returns, indicating that the column name is invalid.

## Example

The following call to `db-execute-immediate` inserts the values 345.6, Gensym, and 1994 into three columns (`col_a`, `col_b`, and `col_c`) of a table named `table_b`, in an Oracle database:

```
status: symbol;
code: integer;
msg: text;
rows-processed: integer;
sql-stmt: text;
...
sql-stmt = "insert into table_b (col_a, col_b, col_c) values
(345.6, 'Gensym', 1994)";
status, code, msg, rows-processed = call db-execute-immediate
(sql-stmt, true, db1-interface);
```

## db-fetch-object

Performs a query and returns the results within either a user-defined G2 object or within a G2 item-list where each element in the item list is a user-defined G2 object containing one row of data.

### Synopsis

db-fetch-object

(*cursor-object*:class db-cursor-object, *user-object*:class item,  
*return-format*:symbol, *rcds-to-fetch*:integer,  
*interface*: class g2-database-interface)

-> *item*:class item, *status*, *code*, *message*, *num-rows*:integer,  
*cursor-position*: integer

Argument	Description
<i>cursor-object</i>	The cursor object that defines this query. For information about how to create cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>user-object</i>	An instance of a user-defined G2 class. The procedure <code>db-fetch-object</code> returns the result of the query in an object or objects of the same class as <i>user-object</i> . The G2-Database bridge generates the object or objects, populates them with the results of the query, and passes them to G2. The object specified by <i>user-object</i> is not modified by this procedure.
<i>return-format</i>	Specify <b>single</b> or <b>list</b> .  Specifying <b>single</b> causes the G2-Database bridge to return all query data in a single object, of the same class as the object that you specify for <i>user-object</i> . If you specify <b>single</b> , the attributes of <i>user-object</i> must be lists, arrays, or sequences.  Specifying <b>list</b> causes the G2-Database bridge to return query data in an item-list. Each element of the item-list is an object containing data from one row of the query. The objects in item-list are of the same class as the object that you specify for <i>user-object</i> .

Argument	Description
<i>rcds-to-fetch</i>	<p>The maximum number of records (rows) to fetch from the database cursor in this execution of <code>db-fetch-object</code>.</p> <p>If you specify 0 for <i>rcds-to-fetch</i>, the call to <code>db-fetch-object</code> returns all the rows in the database cursor.</p> <p><b>Note:</b> The <code>MAXROWS</code> bridge startup option specifies the default maximum number of database rows that a single transaction can return. If <i>rcds-to-fetch</i> specifies a value greater than the value of <code>MAXROWS</code>, the G2-Database bridge returns a warning and the transaction ends after the bridge returns the number of rows specified by <code>MAXROWS</code>. To get the remainder of rows in the cursor, execute <code>db-fetch-object</code> repeatedly until the bridge returns <code>ecursor</code>. For information about the <code>MAXROWS</code> option, see <a href="#">Initial Bridge Memory Requirements</a>.</p>
<i>interface</i>	<p>The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.</p>
Return Value	Description
<u><i>item</i></u>	<p>An item or an item-list of objects containing the result of the query:</p> <ul style="list-style-type: none"> <li>• If you specify <code>single</code> for <i>return-format</i>, <u><i>item</i></u> is a single object of the same class as <i>user-object</i>.</li> <li>• If you specify <code>list</code> for <i>return-format</i>, <u><i>item</i></u> is an item-list of objects of the same class as <i>user-object</i>.</li> </ul>
<u><i>status</i></u> , <u><i>code</i></u> , <u><i>message</i></u>	<p>For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a>.</p>
<u><i>num-rows</i></u>	<p>The number of database rows effected by the transaction.</p>
<u><i>cursor-position</i></u>	<p>The number of the last row returned from the database cursor.</p>

## Description

The procedure `db-fetch-object` performs the database query defined by a specified *cursor-object*. Depending on the value that you specify for *return-format*, the bridge returns the data in a G2 object or in an *item-list* of G2 objects. The object or objects all are of the same user-defined class as the object that you specify for *user-object*.

If your query contains bind variables, you must set values for the bind variables in the SQL statement defined in that *cursor-object* before you can call `db-fetch-object` to execute the query. You can set values for the bind variables by calling either `db-set-cursor` or `db-define-cursor`.

The attributes of the user-defined object must map to the column names and data types of the database table(s) from which the query data is being fetched. For information on mapping G2 data types to database data types see, [Appendix A, Bridge Data Types](#).

When retrieving INTEGER or DECIMAL types and “smart fetch” is enabled:

- When an attribute of *user-object* is an integer, this procedure returns the corresponding value as an integer, or it returns the maximum or minimum G2 integer (which have values 536,870,911 and -536,870,912, respectively) and generates an error if the actual value would have caused an overflow.
- When an attribute of *user-object* is a float, this procedure returns the corresponding value as a float.
- When an attribute of *user-object* is a quantity or value, this procedure returns the corresponding value as an integer, or as a float if returning it as an integer would have caused an overflow.

For more information, see [Using Smart Fetch](#).

When retrieving INTEGERS or DECIMALS, this procedure detects incompatible types and reports an error.

### User-Defined Object Data Types

When you define objects in G2, certain data types for attributes may not be fully supported for population with database data. The following table summarizes G2 data types for object attributes and the corresponding bridge support level:

Data Type	Support Level
Simple values (text, integer, float, etc.)	Full support.
Compound values (sequences, structures)	Support for sequences only.

Data Type	Support Level
Lists and arrays	Full support.
Variables and parameters	Full support.
User-defined objects	Not supported.

## Related Procedures

<a href="#">db-define-cursor</a>	Creates a re-usable cursor object that defines or redefines a query and provides a reference to the database cursor for that query.
<a href="#">db-set-cursor</a>	Sets or changes the values of bind variables in an existing cursor object.
<a href="#">db-fetch-records</a>	Fetches a specified number of rows (records) from a database cursor and returns them to the attributes of a G2 object.
<a href="#">db-refresh-cursor</a>	Refreshes the cursor data and repositions the pointer to the first record in the database cursor.

## Example

### Example 1: Execute a query and return results within a user-defined object.

The following call to `db-fetch-object` executes a query associated with a cursor-object named `my-cursor-obj` and returns the query data within a single object designated by `myOBJ`.

```

status: symbol;
code: integer;
msg: text;
rows-processed: integer;
cursor-pos: integer;
batch-size: integer = 0;
return-format: symbol = the symbol SINGLE;
cursor-obj: class db-cursor-object = my-cursor-obj;
    {created by db-define-cursor};
myItem: class item;

myItem,status,code,msg,rows-processed,cursor-pos =
    call db-fetch-object(cursor-obj, myOBJ, return-format, batch-size, myIO);

```

This call to `db-fetch-object` returns the result of the query within a single object of the same class as the user-defined G2 object `myOBJ`. Because `batch-size` is set to 0, all of the rows from the query are returned to G2.

If you call `db-fetch-object` as in the example above, changing the `return-format` value to `list`, the bridge returns an `item-list` of `myOBJ` objects. Each object in this `item-list` contains one row of data from the database.

**Example 2: Execute a query and return the results within a G2 item-list where the elements of the item-list are user-defined objects.**

The following call to `db-fetch-object` executes a query associated with a `cursor-object` named `my-cursor-obj` and returns the query data within a G2 `item-list`. Each element of the `item-list` contains a user-defined object as specified by `myOBJ`. Each user-defined object contains one row of data from the query.

```

status: symbol;
code: integer;
msg: text;
rows-processed: integer;
cursor-pos: integer;
batch-size: integer = 0;
return-format: symbol = the symbol LIST;
cursor-obj: class db-cursor-object = my-cursor-obj;
    {created by db-define-cursor};
myItem: class item;

myItem,status,code,msg,rows-processed,cursor-pos =
    call db-fetch-object(cursor-obj, myOBJ, return-format, batch-size, myIO);

```

# db-fetch-query-item

Returns data associated with a database cursor to G2 in a query item.

## Synopsis

db-fetch-query-item

(*cursor-object*:class db-cursor-object, *return-format*:symbol,  
*rcds-to-fetch*:integer, *interface*:class g2-database-interface)

-> *query-item*:class db-query-item, *status*, *code*, *message*,  
*num-rows*:integer, *cursor-position*:integer

Argument	Description
<i>cursor-object</i>	The cursor object that defines this query. For information about how to create and define cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>return-format</i>	Specify ARRAYS to cause the database bridge to return data in a query item of the class db-query-item-array. The data returned is contained in attributes of the db-query-item-array that are arrays.  Specify LISTS to cause the database bridge to return data in a query item of the class db-query-item-list. The data is contained in attributes of the db-query-item-list that are lists.
<i>rcds-to-fetch</i>	The maximum number of records (rows) to fetch from the database cursor in this execution of db-fetch-query-item.  If you specify 0 for <i>rcds-to-fetch</i> , the call to db-fetch-query-item returns all the rows in the database cursor.  <b>Note:</b> The MAXROWS bridge startup option specifies the default maximum number of database rows that a single transaction can return. If <i>rcds-to-fetch</i> specifies a value greater than the value of MAXROWS, the G2-Database bridge returns a warning and the transaction ends after the bridge returns the number of rows specified by MAXROWS. To get the remainder of rows in the cursor, execute db-fetch-query-item repeatedly until the bridge returns eocursor. For information about the MAXROWS option, see <a href="#">Initial Bridge Memory Requirements</a> .

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>query-item</i>	An instance of <code>db-query-item-array</code> or <code>db-query-item-list</code> returned by this procedure.
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>num-rows</i>	The number of database rows effected by the transaction.
<i>cursor-position</i>	The number of the last row returned from the database cursor.

## Description

The procedure `db-fetch-query-item` performs the database query defined by a specified *cursor-object*. Depending on the value that you specify for *data-format*, the bridge returns the data to a G2 object of the class `db-query-item-array` or `db-query-item-list`.

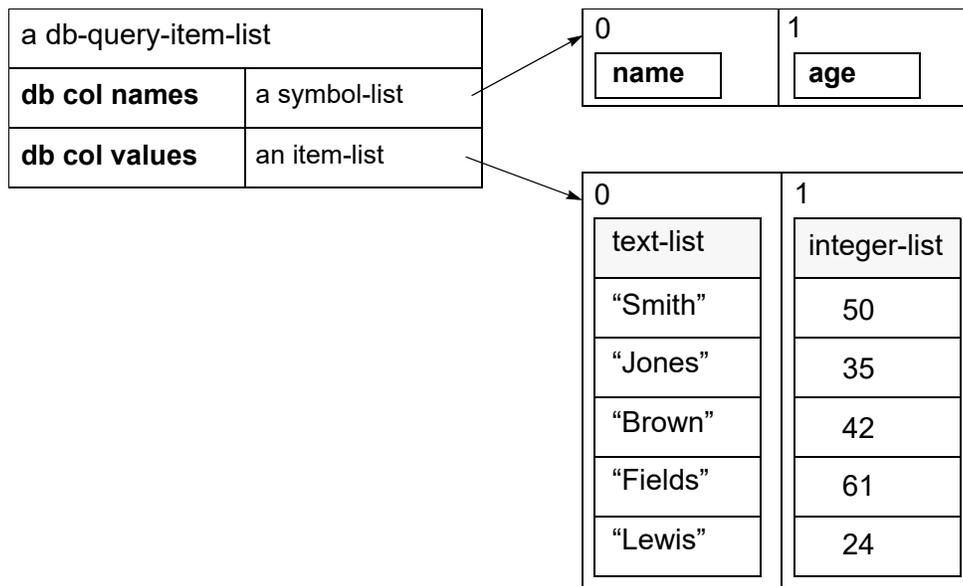
The bridge returns the results of the query in the `db-col-names` and `db-col-values` attributes of a `db-query-item-array` or `db-query-item-list` object:

- `db-col-names` lists the names of the columns in the database cursor. In a `db-query-item-array` object, this attribute is a `symbol-array`. In a `db-query-item-list` object, this attribute is a `symbol-list`.
- `db-col-values` contains the values in each column in the database cursor. In a `db-query-item-array` object, this attribute is an `item-array`. In a `db-query-item-list` object, this attribute is an `item-list`.

Each element of the `item-array` or `item-list` is an array or list of the values in one column in the database cursor.

Each element of the array or list in `db-col-names` corresponds to the same element in the array or list in `db-col-values`. Thus, the first element in `db-col-names` contains the name of a column, and the first element in `db-col-values` contains a list or array of the values in that column, and so on.

For example, the following figure illustrates the contents of the `db-col-names` and `db-col-values` attributes of a `db-query-item-list`:




---

**Caution** To delete a db-query-item-array or db-query-item-list, always use the `delete` user menu choice on that object. The user menu choice starts the method `db-delete`, which deletes not only the db-query-item-array or db-query-item-list itself, but also the lists or arrays in the `db-col-values` attribute. Other ways of deleting a db-query-item-array or db-query-item-list do not delete the lists or arrays in the `db-col-values` attribute and for this reason can cause a memory leak.

---

When retrieving INTEGER or DECIMAL types and “smart fetch” is enabled, this procedure returns the corresponding values as an array or list of quantities, where individual values are returned as integers, or as floats if returning them as integers would have caused an overflow. For more information, see [Using Smart Fetch](#).

## Related Procedures

- [db-define-cursor](#)      Creates a re-usable cursor object that defines or redefines a query and provides a reference to the database cursor for that query.
- [db-set-cursor](#)        Sets or changes the values of bind variables in an existing cursor object.
- [db-fetch-records](#)      Fetches a specified number of rows (records) from a database cursor and returns them to a specified G2 object.

## Example

The following call to `db-fetch-query-item` performs the query defined by a cursor object named `my-cursor-obj` and returns the results of the query to a `db-query-item-list` named `q-item`.

```
status: symbol;  
code: integer;  
msg: text;  
rows-processed: integer;  
cursor-pos: integer;  
batch-size: integer = 0;  
return-format: symbol = the symbol LISTS;  
cursor-obj: class db-cursor-object = my-cursor-obj;  
           {created by db-define-cursor};  
q-item: class db-query-item; {For stricter type-checking, you can declare  
           this local variable as either db-query-item-list or db-query-item-array,  
           depending on the value that you specify for return-format.}  
  
q-item,status,code,msg,rows-processed,cursor-pos  
  = call db-fetch-query-item(cursor-obj, return-format, batch-size, myIO);
```

## db-fetch-records

Fetches a specified number of rows (records) from a database cursor and returns them directly to the attributes of a specified object in G2.

### Synopsis

db-fetch-records

(*cursor-obj*:class db-cursor-object, *item*:class item, *rcds-to-fetch*:integer, *interface*: class g2-database-interface)

-> *status*, *code*, *message*, *rows-processed*:integer, *cursor-position*:integer

Argument	Description
<i>cursor-obj</i>	A cursor object, created by a call to db-define-cursor. For information about how to create and define cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>item</i>	An instance of a G2 class. The procedure db-fetch-records returns the result of the query directly to the attributes of <i>item</i> .
<i>rcds-to-fetch</i>	The number of records (rows) that you want to fetch from the database cursor into the Query Object in a single execution of db-fetch-records.  If you specify 0 for <i>rcds-to-fetch</i> , the call to db-fetch-records returns all the rows in the database cursor.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.

The following values are returned to the db-status, db-code, db-message, db-rows-processed, and db-cursor-position attributes in the Query Object.

Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of database rows processed by this transaction.
<i>cursor-position</i>	The last row returned from the database cursor.

## Description

This procedure fetches any number of rows (records) that you specify from a database cursor. The rows are inserted into column attributes of an object that correspond to database column or alias names. For information on mapping G2 data types to database data types see, [Appendix A, Bridge Data Types](#).

If you set *rcds-to-fetch* to a number other than 0, `db-fetch-records` returns that number of records to the specified Query Object. To fetch all the available records, call `db-fetch-records` repeatedly until the end of cursor (`ecursor`) is reached, or set *rcds-to-fetch* to 0.

If you want to retrieve more than one record at a time and store all records in one object, define your object column attributes as lists.

For information about how to perform queries using Query Objects, see [Query Objects](#).

When retrieving INTEGER or DECIMAL types and “smart fetch” is enabled:

- When an attribute of *user-object* is an integer, the corresponding value is returned as an integer, or, if returning it as an integer would cause G2 integer overflow, the maximum or minimum valid G2 integer (which have values 536,870,911 and -536,870,912, respectively) is stored in the object and an error is returned by the procedure call.
- When an attribute of *user-object* is a float, the corresponding value is stored as a float.
- When an attribute of *user-object* is a quantity, the value is stored as an integer or, if it will not fit in a G2 integer, as a float.

For more information, see [Using Smart Fetch](#).

When “smart fetch” is not enabled:

- When the attribute of *user-object* is an integer, an error is generated if the value would cause G2 integer overflow.
- When the attribute type of *user-object* is incompatible with the value being returned, an error is generated.

When retrieving INTEGERS or DECIMALS, this procedure detects incompatible types and reports an error.

## Related Procedure

<a href="#">db-define-cursor</a>	Creates a re-usable cursor object that defines or redefines a query and provides a reference to the database cursor for that query.
<a href="#">db-set-cursor</a>	Sets or changes the values of bind variables in an existing cursor object.
<a href="#">db-fetch-query-item</a>	Performs a query and returns the result to G2 in a query item.
<a href="#">db-refresh-cursor</a>	Refreshes the cursor data and repositions the pointer to the first record in the database cursor.

## Example

The following call to `db-fetch-records` executes the query associated with a cursor-object named `my-cursor-obj`. The bridge returns the results of the query to attributes of the G2 object `myObj`.

```
status: symbol;
code: integer;
msg: text;
rows-processed: integer;
cursor-pos: integer;
batch-size: integer = 0;
myObj: class object;
cursor-obj: class db-cursor-object
    = my-cursor-obj; {created by db-define-cursor}
...
create an instance of the class named by myObj;
change the gsi-interface-name of myObj to the name of myIO;
...
status,code,msg,rows-processed,cursor-pos
    = call db-fetch-records(cursor-obj, myObj, batch-size, myIO);
```

## db-fetch-structure

Performs a query and returns the results within a structure where each element of the structure is a name/sequence pair containing one column of database data.

### Synopsis

db-fetch-structure

(*cursor-object*:class db-cursor-object, *rcds-to-fetch*:integer,

*interface*: class g2-database-interface)

-> *struct*:structure, *status*, *code*, *message*, *num-rows*:integer,

*cursor-position*: integer

Argument	Description
<i>cursor-object</i>	The cursor object that defines this query. For information about how to create cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>rcds-to-fetch</i>	The maximum number of records (rows) to fetch from the database cursor in this execution of db-fetch-structure.  If you specify 0 for <i>rcds-to-fetch</i> , the call to db-fetch-structure returns all the rows in the database cursor.  <b>Note:</b> The MAXROWS bridge startup option specifies the default maximum number of database rows that a single transaction can return. If <i>rcds-to-fetch</i> specifies a value greater than the value of MAXROWS, the G2-Database bridge returns a warning and the transaction ends after the bridge returns the number of rows specified by MAXROWS. To get the remainder of rows in the cursor, execute db-fetch-structure repeatedly until the bridge returns eocursor. For information about the MAXROWS option, see <a href="#">Initial Bridge Memory Requirements</a> .
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<u><i>struct</i></u>	A structure that contains a sequence for each database column resulting from the query. The sequences contain all of the rows of data resulting from the query.
<u><i>status, code, message</i></u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<u><i>num-rows</i></u>	The number of database rows effected by the transaction.
<u><i>cursor-position</i></u>	The number of the last row returned from the database cursor.

## Description

The procedure `db-fetch-structure` performs the database query defined by a specified *cursor-object*. The bridge returns the data in a G2 structure.

- If the number of rows fetched by a query equals 0, a structure containing no elements will be returned to G2.
- If the number of rows fetched by a query is greater than 0, a structure will be returned to G2 where the elements of the structure consist of name/sequence pairs. Each of the name/sequence pairs represents a database column. The sequences contain all of the rows of data generated by the query.

If your query contains bind variables, you must set values for the bind variables in the SQL statement defined in that *cursor-object* before you can call `db-fetch-structure` to execute the query. You can set values for the bind variables by calling either `db-set-cursor` or `db-define-cursor`.

Because `db-fetch-structure` uses sequences and structures to return data to G2, there are several advantages to using this approach to fetch data.

- While structures and sequences offer similar functionality to objects and lists, respectively, they consume significantly less memory.
- The memory used by structures and sequences is managed automatically by G2. This means that you do not have to worry about deleting them.
- Structures and sequences can be transmitted much faster between G2 and a bridge process (under most conditions). This means that you should see better performance when fetching structures than when fetching G2 objects.

When retrieving INTEGER or DECIMAL types from the database and “smart fetch” is enabled, this procedure returns values as integers, or as floats if

returning them as integers would cause G2 integer overflow. For details, see [Using Smart Fetch](#).

## Related Procedures

<a href="#">db-define-cursor</a>	Creates a re-usable cursor object that defines or redefines a query and provides a reference to the database cursor for that query.
<a href="#">db-set-cursor</a>	Sets or changes the values of bind variables in an existing cursor object.
<a href="#">db-fetch-query-item</a>	Performs a query and returns the result to G2 in a query item.
<a href="#">db-refresh-cursor</a>	Refreshes the cursor data and repositions the pointer to the first record in the database cursor.

## Example

The following call to `db-fetch-structure` executes a query associated with a cursor-object named `my-cursor-obj` and returns the query data within a G2 structure:

```
status: symbol;  
code: integer;  
msg: text;  
rows-processed: integer;  
cursor-pos: integer;  
batch-size: integer = 0;  
struct: structure;  
  
struct,status,code,msg,rows-processed,cursor-pos =  
  call db-fetch-structure(cursor-obj, batch-size, myIO);
```

# db-get-triggers

Returns a list of currently active trigger watches.

## Synopsis

db-get-triggers

(*interface*: class g2-database-interface)

-> *trigger-list*: symbol-list, *trigger-count*:integer, *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<i>trigger-list</i>	A list of the names of the currently active trigger watches.
<i>trigger-count</i>	The number of currently active trigger watches.
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

Returns a list of currently active trigger watches for a bridge process and the total number of all such trigger watches.

## Related Procedures

[db-set-trigger](#) Enables or disables a trigger watch.

## Example

The following call to `db-get-triggers` returns the names and total number of the currently active trigger watches:

```
status: symbol;  
code: integer;  
msg: text;  
trigger-names: class symbol-list;  
trigger-count: integer;
```

```
trigger-names, trigger-count, status,code,msg = call db-get-triggers(myIO);
```

# db-io-status

Changes display (colors) of the G2-Database Interface icon to reflect changes to the status of the connection between the G2-Database bridge and the database.

## Synopsis

db-io-status  
(*interface*: class g2-database-interface)

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

## Description

The procedure `db-io-status` set regions of the G2-Database Interface icon to different colors whenever the value of the `gsi-interface-status` attribute of the `g2-database` interface object changes.

You can also use `db-io-status` to set the color of the status region of SQL objects, cursor objects, trigger objects, and query objects.

This procedure is automatically called when `db-colors` is set to `true`.

For information about how to use `db-io-status` to provide an ongoing visual indication of changes to the status of a connection between G2 and a G2-Database bridge, see [Displaying the Connection Status](#).

---

**Note** This procedure is not intended for use by end users.

---

## db-kill-bridge

Kills a specified G2-Database bridge process.

### Synopsis

db-kill-bridge  
*(interface: class g2-database-interface)*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

### Description

The procedure db-kill-bridge kills a specified G2-Database bridge process. All connections from the bridge to the database(s) are closed for all contexts, and all resources that were used by the connection(s) are freed.

This procedure affects all connections and contexts associated with the specified G2-Database bridge process.

---

**Note** Because this procedure returns no values, you must invoke it by using a **start** action, rather than a **call** procedure statement. See [Invoking a Bridge Procedure from a Rule, Action-Button, or User-Menu-Choice](#).

---

To start the bridge on the local machine, use the G2 system procedure `g2-spawn-process-with-arguments`, or to start the bridge on a remote machine with a Telewindows client, use `g2-spawn-remote-process-with-arguments`. You can also use telnet and batch file or script to start the bridge remotely.

### Example

The following call to db-kill-bridge kills the bridge process that is running on a connection configured by the G2-Database interface object named db1-interface.

```
start db-kill-bridge(db1-interface);
```

# db-logfile

Creates, opens, and closes log files and modifies log file filter settings.

## Synopsis

db-logfile

(*mode*:symbol, *filter*:text, *interface*:class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>mode</i>	<p>Specify <b>open</b> or <b>close</b>, to open or close the log file. If the logfile does not exist, the procedure creates one.</p> <p>This argument must be specified as a symbol.</p> <p>Example: the symbol <b>CLOSE</b></p>
<i>filter</i>	<p>Specifies the category or categories of messages that can be written to the log file. Possible values are: <b>ALL</b>, <b>FATAL</b>, <b>ERROR</b>, <b>WARN</b> or <b>WARNING</b>, and <b>INFO</b>. Only messages of the category specified by <i>filter</i> are written to the log file. If you enter more than one value for <i>filter</i>, separate the values by commas or spaces.</p> <p>For more information about how to filter messages written to the log file, see <a href="#">Saving Messages in Log Files</a>.</p>
<i>interface</i>	<p>The G2-Database interface object that configures the connection for which you are logging messages. Specify the name of the G2-Database interface object.</p> <p><b>db-logfile</b> uses the log file specified in the <b>log-file</b> attribute of this G2-Database interface object. If no log file is specified in the <b>log-file</b> attribute, logging of messages is disabled.</p>

Return Value	Description
<i>status</i>	One of the following symbols: SUCCESS, ERROR, WARNING, INFO, or DISCONNECTED.
<i>code</i>	The error or information code as an integer.
<i>message</i>	The associated message text.  The log message has the same fields as messages returned by the procedure <code>db-context-event-msg</code> , with a timestamp prefixed

## Description

The procedure `db-logfile` opens a new log file, closes an open log file, or reopens a closed log file.

Messages logged to a reopened log file are appended to any existing messages in that log file.

Each context can have one logfile. For information about contexts, see [Running a Bridge with Multiple Connections to G2](#).

## Examples

**Example 1:** The following call to `db-logfile` opens the log file specified in the `logfile` attribute of an G2-Database interface object named `db1-interface`. The call specifies that only error and warning messages are to be logged.

```
status:symbol;
code: integer;
msg: text;
...
status, code, msg =
  call db-logfile(the symbol OPEN, "ERROR, WARNING", DB1-INTERFACE);
```

**Example 2:** The following call to `db-logfile` manages the logfile associated with a G2-Database interface object. This procedure allows you to open, close or change the logfile filters for a logfile.

```
status: symbol;
code: integer;
msg: text;
mode: symbol = the symbol OPEN; {or CLOSE}
filter: text = "ERROR,WARN"; {or INFO,FATAL,ALL}

status,code,msg = call db-logfile( mode, filter, myIO );
```

**Example 3:** This procedure opens the logfile associated with myIO (if it is not already open) and set the logfile filter to **error** and **warning** messages. All other messages will not be logged. Multiple filter arguments may be specified by separating arguments with commas. If full logging is required then specify ALL. Note, the logfile is closed when the KB is paused and re-opened upon a resume.

```
mode: symbol = the symbol CLOSE;  
filter: text = "";
```

```
status,code,msg = call db-logfile( mode, filter, myIO );
```

## db-ping

Returns the status of the connection between the G2-Database bridge and the database.

### Synopsis

db-ping  
 (*interface*: class g2-database-interface)  
 -> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

### Description

This procedure determines the status of the connection between the G2-Database bridge and the database and returns this information in the *status*, *code*, and *message* values.

If the connection exists, `db-ping` returns **success** to *status*. If a connection does not exist, it returns **error** to *status*, as well as any error information provided by the database to *code* and *message*.

### Example

The following call to `db-ping` returns the status of the connection configured by the G2-Database interface object named `myIO`.

```
status: symbol;
code: integer;
msg: text;
...
status, code, msg = call db-ping(myIO);
```

# db-redirect-callback

Redirects messages initially intended for either db-context-event-msg or db-trigger-event to any user defined G2 procedure.

## Synopsis

db-redirect-callback

(*old-callback*: symbol, *new-callback*: symbol)

-> *status*, *code*, *message*

Argument	Description
<i>old-callback</i>	<p>Specify db-context-event-msg or db-trigger-event.</p> <p>Specifying db-context-event-msg causes the G2-Database bridge to return messages intended for the db-context-event-msg procedure to the procedure designated by <i>new-callback</i>.</p> <p>Specifying db-trigger-event causes the G2-Database bridge to return trigger messages intended for the db-trigger-event procedure to the procedure designated by <i>new-callback</i>.</p>
<i>new-callback</i>	<p>The name of the G2 procedure that will receive messages that were initially intended for the procedure named by <i>old-callback</i>.</p>
<i>interface</i>	<p>The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.</p> <p>This procedure references no attributes in the G2-Database interface object.</p>
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	<p>For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a>.</p>

## Description

This procedure instructs the bridge to return messages that were initially intended for `db-context-event-msg` or `db-trigger-event`, to some other G2 procedure.

The signature of the user defined procedure that will be receiving messages must match the signature of the procedure that it is replacing. If the signature does not match, then a G2 error will occur when the bridge attempts to return a message to the procedure.

---

**Note** When you call `db-redirect-callback`, messages are only redirected for the context (g2-database-interface object) across which the `db-redirect-callback` procedure was called. If you want to redirect messages for every context, then you must call `db-redirect-callback` across each g2-database-interface object.

---

If you want to restore messaging to the original procedures, then you would simply call `db-redirect-callback` with the name of the G2-Database procedure in both the *old-callback* and *new-callback* arguments.

If you want to redirect messages from `db-context-event-msg` to your own procedure, then your procedure must have the same arguments as `db-context-event-msg`. The arguments must match both in order and in data type.

The procedure allows messages of up to approximately 4000 bytes to be transferred from Oracle to G2.

## Related Procedures

<a href="#">db-context-event-msg</a>	Reports the occurrence of significant events during the execution of the G2-Database bridge
<a href="#">db-trigger-event</a>	Reports the occurrence of trigger messages to G2

## Example

### Example 1: Redirecting messages to a user defined procedure.

The following example illustrates how you might redirect messages from `db-context-event-msg` to a procedure named `my-event-msgs`.

```

status: symbol;
code: integer;
msg: text;
. . .
status, code, msg = call db-redirect-callback(the symbol db-context-event-msg,
the symbol my-event-msgs);

```

This assumes that your procedure has a signature similar to the following:

```
my-event-msgs (  
  status: symbol,  
  code: integer,  
  message: text,  
  facility: symbol,  
  timestamp: text ) = ()  
begin  
  
  { Code your message handler here}  
  
end
```

**Example 2: Restoring messaging to db-context-event-msg.**

The following example illustrates how you would restore messaging from a user defined procedure back to db-context-event-msg.

```
status: symbol;  
code: integer;  
msg: text;  
...  
status, code, msg = call db-redirect-callback(the symbol db-context-event-msg,  
  the symbol db-context-event-msg);
```

## db-refresh-cursor

Refreshes the cursor data and repositions the cursor pointer to the first record in the database cursor.

### Synopsis

db-refresh-cursor

(*cursor-obj*:class db-cursor-object, *interface*: class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>cursor-obj</i>	The name of a Cursor Object that has been defined by db-define-cursor and associated with an SQL Select statement.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.

The following values are returned to the db-status, db-code, and db-messages attributes in the Query Object.

Return Value	Description
<u><i>status</i></u> , <u><i>code</i></u> , <u><i>message</i></u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

### Description

Refreshes the cursor and repositions the cursor pointer to the first record. You must call this procedure if you want to fetch from a cursor again after fetching the last record in the cursor (status = eocursor), or if you want to begin fetching from the beginning of the cursor.

This procedure does not cause any data to be fetched.

## Related Procedures

<a href="#"><u>db-define-cursor</u></a>	Creates a re-usable cursor object that defines a query and provides a reference to the database cursor for that query.
<a href="#"><u>db-set-cursor</u></a>	Sets or changes the values of bind variables in an existing cursor object
<a href="#"><u>db-fetch-object</u></a>	Performs a query and returns the result to a user-defined G2 object.
<a href="#"><u>db-fetch-query-item</u></a>	Returns data associated with a database cursor to G2 in a query item.
<a href="#"><u>db-fetch-records</u></a>	Fetches a specified number of rows (records) from a database cursor and returns them to the attributes of a G2 object.

## Example

The following call to `db-refresh-cursor` resets the cursor pointer to the beginning of the database cursor (first row of query).

```
status: symbol;  
code: integer;  
msg: text;  
cursor-obj: class db-cursor-object;  
...  
status,code,msg = call db-refresh-cursor(cursor-obj, myIO);
```

## db-rollback

Cancels (undoes) all changes made in a specified context since the last commit (save) to your database.

### Synopsis

db-rollback

(*interface*: class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

### Description

The procedure db-rollback cancels (undoes) all changes since the last commit (save) to your database. This procedure:

- Undoes all changes made to the database during the current transaction.
- Ends the transaction.
- Releases all row and table locks.

You cannot roll back a transaction after it has been committed.

### Related Procedures

[db-execute-immediate](#) Executes a non-query SQL statement that does not contain bind variables or use an SQL object.

[db-commit](#) Commits all changes made since your last commit (save) to your database.

---

**Note** The actual behavior of `db-rollback` may vary depending upon the database being used. Consult the Release Notes and the database specific documentation for details on using `db-rollback`.

---

## Example

The following call to `db-rollback` undoes all changes made since the last commit on the connection configured by the G2-Database interface object named `db1-interface`.

```
status: symbol;  
code: integer;  
msg: text;  
...  
status, code, msg = call db-rollback(db1-interface);
```

The following call to `db-rollback` aborts the transaction submitted to the database by `db-execute-immediate`:

```
status: symbol;  
code: integer;  
msg: text;  
rows-processed: integer;  
sql-stmt: text = "insert into emp (ename) values ('Smith')";  
auto-commit: truth-value = FALSE;  
  
status,code,msg,rows-processed = call db-execute-immediate(sql-stmt,  
    auto-commit, myIO);  
status,code,msg = call db-rollback(myIO);
```

## db-set-connection-status

Called by G2-Database procedures to set the `database-connection-status` attribute of a G2-Database interface object and to set the icon color to reflect the status of the connection between the bridge and the database.

### Synopsis

`db-set-connection-status`

(*status*: symbol, *code*: integer *interface*: class g2-database-interface)

Argument	Description
<i>status</i>	The return status from a call to a database.
<i>code</i>	The error code.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge.

### Description

The procedure `db-set-connection-status` is used by system procedures to set the `database-connection-status` attribute of G2-Database interface objects. This procedure is called by g2-Database procedures following a transaction that does not result in a *status* of *success*. This procedure also sets the icon colors for the database portion of the g2-Database interface object icon.

---

**Note** This procedure is not intended for use by end users.

---

# db-set-cursor

Sets or changes the values of bind variables in an existing cursor object.

## Synopsis

db-set-cursor

(*cursor-object*:class db-cursor-object, *bind-variables*:text-list or text,  
*interface*: class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>cursor-object</i>	An existing cursor object.
<i>bind-variables</i>	The bind variable(s) in the SQL statement that this call to <b>db-set-cursor</b> sets or changes. Bind variables must be supplied as a text-list. If you prefer, you can specify bind values as a text string with each value separated by a comma. Note: This approach is not as efficient as providing a text-list.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<u><i>status</i></u> , <u><i>code</i></u> , <u><i>message</i></u>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

This procedure sets the values of bind variables in the SQL statement in a specified cursor object.

If you call **db-define-cursor** to create a cursor object without specifying values for bind variables, you must call **db-set-cursor** to set the values of the bind variables in the cursor object before you can perform a query using that cursor object.

**db-set-cursor** changes the values of bind variables without reprocessing the SQL statement or requiring the database to generate a new database cursor. Thus, this procedure enables you to change the values of the bind variables without significant processing overhead either in the bridge or in the database.

## Related Procedures

### [db-define-cursor](#)

Creates a re-usable cursor object that defines a query and provides a reference to the database cursor for that query.

## Example

The following call to `db-set-cursor` modifies the bind variables of the SQL statement in the cursor object `my-cursor-obj`. The procedure modifies the bind variable that was initially set to `557` to `778`.

```
status: symbol;  
code: integer;  
msg: text;  
cursor-obj-name: symbol = the symbol MY-CURSOR-OBJ;  
cursor-obj: class db-cursor-object;  
sql-stmt: text = "select ename from emp where empid = :n";  
bind-vars1, bind-vars2: text-list;
```

```
insert "557" at the end of bind-vars1;  
sql-obj,status,code,msg, = call db-define-cursor(cursor-obj-name, sql-stmt,  
    bind-vars1, myIO);  
insert "778" at the end of bind-vars2;  
status,code,msg, = call db-set-cursor(cursor-obj, bind-vars2, myIO);
```

# db-set-sql

Sets or changes the values of bind variables in an SQL object where bind variables are represented as simple values.

## Synopsis

db-set-sql

(*sql-object*:class db-sql-object, *bind-variables*:text-list or text,  
*interface*: class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>sql-object</i>	An existing SQL object.
<i>bind-variables</i>	The values to which the bind variables in the SQL statement in SQL object will be set. These values should be provided as either a text-list or as a text string with values separated by commas. It is more efficient to use a text-list.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure **db-set-sql** modifies the bind variables of a non-query SQL statement declared by **db-define-sql**.

If you call **db-define-sql** to create an SQL object without specifying values for bind variables, you must call **db-set-sql** to set the values of the bind variables in the SQL object before you can perform a database operation using that SQL object.

The procedure **db-set-sql** changes the values of bind variables without reprocessing the SQL statement or requiring the database to generate a new database cursor. Thus, this procedure enables you to change the values of the bind variables without significant processing overhead either in the bridge or in the database.

## Related Procedures

[db-define-sql](#)

Creates an SQL object for use in non-query database operations on individual values.

## Example

The following call to `db-set-sql` changes the values of the bind variables in a cursor object named `my-sql-obj` from “Smith, 50” to “Jones, 30”.

```
status: symbol;  
code: integer;  
msg: text;  
sql-obj-name: symbol = the symbol my-sql-obj;  
sql-obj: class sql-object;  
sql-stmt: text = "insert into emp (ename,age) values (:n,:2)";  
bind-vars1, bind-vars2: class text-list;  
  
create a text-list bind-vars1;  
create a text-list bind-vars2;  
insert "Smith" at the end of bind-vars1;  
insert "50" at the end of bind-vars1;  
sql-obj,status,code,msg, = call db-define-sql(sql-obj-name, sql-stmt,  
    bind-vars1, myIO);  
insert "Jones" at the end of bind-vars2;  
insert "30" at the end of bind-vars2;  
status,code,msg, = call db-set-sql(sql-obj, bind-vars2, myIO);
```

# db-set-sql-obj

Sets or changes the values of bind variables in an SQL statement with the values that are stored in attributes of a user-defined G2 object.

## Synopsis

db-set-sql-obj

(*sql-object*:class db-sql-object, *bind-variable-names*:text,  
*user-object*:class object, *interface*:class g2-database-interface)  
-> *status*, *code*, *message*

Argument	Description
<i>sql-object</i>	An existing SQL object.
<i>bind-variable-names</i>	Represents the <i>names</i> of the attributes of <i>user-object</i> that correspond to the bind variables in the SQL statement defined by db-define-sql-obj. The values of the attributes of <i>user-object</i> will be associated with the SQL statement defined in db-define-sql-obj in the order in which they appear.  If you specify two or more bind variable names, separate them using commas.  Specify values for <i>bind-variable-names</i> in an order that corresponds to the order of the bind variables in the SQL object <i>sql-object</i> .
<i>user-object</i>	A G2 object of a user-defined class. The bind variable names that you specify for <i>bind-variable-names</i> must match the names of attributes of this object.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.

Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure `db-set-sql-object` sets the values of bind variables in an SQL object, using values stored in attributes of a user-defined G2 object. The bind variable names specified in the *bind-variable-names* argument must match the names of attributes of the user-defined G2 object specified in the *user-object* argument.

The procedure `db-define-sql-obj`, that creates SQL objects, cannot set the values of bind variables in the SQL objects. Thus, you must call `db-set-sql-object` to set the values of bind variables in an SQL object before you can use that SQL object to perform a database operation.

---

**Note** The order in which you specify the bind variable names is important as the bind variable names are paired to the bind variables within the SQL statement in the order in which they appear. For example, the first bind variable name is paired with the first occurrence of a bind variable in the SQL statement.

---

## Related Procedures

### [db-define-sql-obj](#)

Creates an SQL object for use in non-query database operations where the bind variable values will be supplied in a G2 object.

## Example

The following example illustrates:

- How `db-define-sql-obj` creates an SQL object.
- How `db-set-sql-obj` set values of bind variables in that SQL object.

The following call to `db-define-sql-obj` creates an SQL object named `my-sql-obj`, which contains two bind variables, `:n` and `:a`.

```
status: symbol;
code: integer;
msg: text;
sql-obj-name: symbol = the symbol MY-SQL-OBJ;
my-sql-obj: class db-sql-object;
sql-stmt: text = "insert into emp (ename,age) values (:n, :a)";

my-sql-obj,status,code,msg = call db-define-sql-obj(sql-obj-name,
    sql-stmt, myIO);
```

Assume that a user defined G2 object, `myObj`, has the following attributes:

`ename` is an instance of a `text-list`;

`age` is an instance of an `integer-list`;

The following call to `db-set-sql-obj` uses the values of the attributes in `myObj` to set the values of the two bind variables `:n` and `:a`.

```
bind-var-names: text = "ename,age";  
status,code,msg = call db-set-sql-obj(sql-obj, bind-var-names, myObj, myIO);
```

`db-set-sql-object` sends the object associated with `myOBJ` to the bridge. The bridge then:

- 1** Matches a bind variable name specified in `bind-var-names` to the name of an attribute within the object `myObj` (e.g., `ename` must exist in `myObj`).
- 2** Matches the bind variable value(s) of this attribute with the bind variable name specified during `db-define-sql-obj` (e.g., the values located within the `ename` attribute of `myObj` are associated with bind variable `:n`, etc.).
- 3** Repeats this procedure for each bind variable.

## db-set-trigger

Enables or disables a trigger watch of a specified trigger name.

### Synopsis

db-set-trigger

(*trigger*:class db-trigger-object or symbol, *trigger-state*:truth-value,  
*interface*: class g2-database-interface)

-> *status*, *code*, *message*

Argument	Description
<i>trigger</i>	Specify the name of a database trigger (a symbol), or a named trigger object (db-trigger-object).
<i>trigger-state</i>	Set to <b>true</b> to enable the trigger watch for this bridge, or to <b>false</b> to disable the trigger.  The value of <i>trigger-state</i> does not affect the trigger in the database.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

### Description

This procedure sets a trigger watch, with the result that when a trigger event occurs in the database, the G2-Database bridge does one of two things:

- If you specify a G2 trigger object for the *trigger* argument, it sends information about the trigger directly to the **last-recorded-value** attribute of that trigger object, and it sends a time stamp to the **time-stamp** attribute.
- If you specify the name of a trigger for the *trigger* argument, it sends information about the trigger event to the G2 procedure **db-trigger-event**.

You can set a maximum of 50 trigger watches for each G2-Database bridge process.

The procedure allows messages of up to approximately 4000 bytes to be transferred from Oracle to G2.

## Related Procedures

<a href="#">db-get-triggers</a>	Returns a list of currently active trigger watches.
<a href="#">db-trigger-event</a>	Called when a trigger event specified by <code>db-set-trigger</code> occurs in the database.
<a href="#">db-redirect-callback</a>	Designate an alternate G2 procedure that will receive trigger event messages.

## Examples

**Example 1:** The following call to `db-set-trigger` enables a trigger watch on a trigger named `my-trigger-object` (which must also be the name of the `db-trigger-object` in G2). If a trigger event occurs in the database for this trigger, the bridge sends a text message directly to the trigger object.

```
status: symbol;  
code: integer;  
msg: text;  
activate-trigger: truth-value = true;  
trigger-obj: class db-trigger-object = my-trigger-object;  
...  
status,code,msg = call db-set-trigger(trigger-obj, activate-trigger, myIO);
```

Since a `trigger-object` was supplied as the first argument to this procedure, a trigger watch is established to watch for a trigger event in the database named `my-trigger-object`. If the trigger occurs, then a message is sent directly to the trigger object named `my-trigger-object`.

**Example 2:** In the following call to `db-set-trigger`, the name of a trigger (rather than the name of a trigger object) is the first argument. The call enables a trigger watch on a trigger named `my-trigger`. If a trigger event occurs in the database for this trigger, the bridge sends a text message to the G2-Database procedure `db-trigger-event`.

```
trigger-name: symbol = the symbol MY-TRIGGER-OBJECT;  
activate-trigger: truth-value = true;  
...  
status,code,msg = call db-set-trigger(trigger-name, activate-trigger, myIO);
```

By setting the `activate-trigger` argument to `false`, the trigger watch is disabled. This trigger watch can be disabled by calling `db-set-trigger` with `activate-trigger` set to `FALSE` instead of `TRUE`. When a trigger watch is disabled, the bridge does not report database trigger events for that trigger to G2.

# db-sql-function

Returns a single text or float value resulting from a query.

## Synopsis

db-sql-function

(*sql-stmt*:text, *interface*: class g2-database-interface)

-> *num-val*:float, *text-val*:text, *status*, *code*, *message*

Argument	Description
<i>sql-stmt</i>	Any valid SQL select statement.
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.
Return Value	Description
<i>num-val</i>	The single value result (numeric) of the SQL function. Set to a null value if a text value is returned to the text-val argument.
<i>text-val</i>	The single value result (text) of the SQL function. Set to a null value if a float value is returned to the num-val argument.
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure `db-sql-function` can use any SQL select statement to return a single value. If the select statement returns more than one value, `db-sql-function` returns only the first of these values. The select statement can invoke SQL functions such as `SUM`, `MAX`, `MIN`, `AVG`, and `COUNT`, which return single values. Each call to `db-sql-function` can execute only one function.

If the result data is numeric, it is converted to a float and returned to *num-val*. The *text-val* argument is set to the null value that you specified in the null-string attribute of the G2-Database interface object.

If the result data is text, it is returned to the *text-val* argument, and the *num-val* argument is set to the null value that you specified in the null-number attribute of the G2-Database interface object.

If the **select** statement returns 0 rows, both *text-val* and *num-val* are set to null values.

---

**Note** db-sql-function does not return values to objects.

---

## Examples

**Example 1:** The following call to db-sql-function calls **select** to select names from a database table named **emp**:

```
status: symbol;
code: integer;
msg: text;
sql-stmt: text;
num-result: float;
txt-result:text;
...
sql-stmt = "select ename from emp";

num-result, txt-result, status, code, msg =
  call db-sql-function(sql-stmt, db1-interface);

if status is SUCCESS then
  inform the operator that "The name = [txt-result]";
```

Assume the following database table **emp**:

<b>ename</b>	<b>age</b>
John	10
Bill	12
Judy	20
Tony	55

In this example, the values returned by db-sql-function include:

**num-result** = 0 or the value of null-number in your G2-Database interface object.

**txt-result** = "John"

**Example 2:** The following call to `db-sql-function` executes a query and returns row #1, column #1 of the query. This procedure is useful for performing queries where 1 value is desired (such as `count`, `avg`, `sum` and other functions). This procedure returns either a text or float value, depending upon the result.

```
status: symbol;  
code: integer;  
msg: text;  
sql-stmt: text = "select count(*) from emp";  
float-val: float;  
text-val: text;
```

```
float-val,text-val,status,code,msg = call db-sql-function(sql-stmt, myIO);
```

In this example, since the value for `count` is an integer, the value will be returned to the local `float-val` variable. If the result of the operation generated a text value, then the value would have been returned in the `text-val` variable. If there is no value for `float-val` or `text-val`, `float-val` or `text-val` is set to the null value specified in the G2-Database interface object.

# db-startup

A procedure that calls db-configuration and db-connect to configure and establish a connection between G2 and a G2-Database bridge.

## Synopsis

db-startup  
(*interface*: class g2-database-interface)  
-> *status*, *code*, *message*

Argument	Description
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.  This procedure references no attributes in the G2-Database interface object.
Return Value	Description
<i>status</i> , <i>code</i> , <i>message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .

## Description

The procedure db-startup is called automatically by a rule on the subworkspace of the g2-database-interface class definition when a connection is established between G2 and a bridge to configure and establish a database connection. The recommended practice is to invoke db-startup only through this rule. For information about this rule, see [Sending Connection Configuration Information to the Bridge](#).

The procedure db-startup calls db-configuration, which sends configuration information provided by a G2-Database interface object to the bridge after G2 has established an active connection to the bridge.

When db-configuration successfully completes execution, db-startup then attempts to establish a connection between the bridge and the database by calling db-connect.

---

**Note** This procedure is not intended for use by end users.

---

## db-text-to-text-list

Converts a text string that contains values separated by commas into a text-list.

### Synopsis

```
db-text-to-text-list
  (text-string:text)
  -> text-list
```

Argument	Description
<i>text-string</i>	Comma delimited text string of bind values.

Return Value	Description
<u><i>text-list</i></u>	A text-list where the elements of the text list represent the values from the comma delimited text string <i>text-string</i> .

### Description

The procedure `db-text-to-text-list` converts a text string that contains values separated by commas into a text-list. This procedure is called by G2-Database procedures that need to convert their bind variable arguments into a text-list. Many G2-Database procedures accept either a text-list or a text as a value for their bind variable argument. However, in order to use the bind variables, they must be converted into a text list.

#### Example

The following example shows how to convert a text string into a text-list.

```
text-string: text = "Jones,Smith,Reynolds,Jackson";
text-list: class text-list;

text-list = call db-text-to-text-list(text-string);
```

---

**Note** This procedure is not intended for use by end users.

---

# db-trigger-event

A user-modifiable procedure that the bridge calls to send trigger messages to G2. This procedure may be used as a foundation procedure for developing a customer event handler.

## Synopsis

db-trigger-event

(*trigger-name*:symbol, *message*:text, *timestamp*:text)

Argument	Description
<i>trigger-name</i>	The name of a trigger event specified by a call to db-set-trigger.
<i>message</i>	The message associated with the database trigger event.
<i>timestamp</i>	The timestamp provided with the trigger event message.

## Description

The procedure db-trigger-event is a user-modifiable procedure that G2 calls when a trigger event specified by db-set-trigger occurs in the database.

You complete the code of db-trigger-event to specify how your G2 application responds to the trigger event.

## Recommended Approach

Alternatively, you can instruct the bridge to send messages to any user defined G2 procedure by calling db-redirect-callback. This is the recommended approach as it does not require you to modify db-trigger-event. For information about db-redirect-callback see [db-redirect-callback](#).

## Related Procedures

[db-get-triggers](#)

Returns a list of currently active trigger watches.

[db-set-trigger](#)

Enables or disables a trigger watch on a specified trigger name.

[db-redirect-callback](#)

Allows you to redirect event messages, that would normally be sent to db-trigger-event, to some other G2 procedure.

# db-update-object

Returns data associated with a database cursor to an existing object in G2. The data within the object may be either replaced or appended.

## Synopsis

db-update-object

(*cursor-object*:class db-cursor-object, *user-object*:class object,  
*update-action*:symbol, *rcds-to-fetch*:integer,  
*interface*: class g2-database-interface)  
-> *status*, *code*, *message*, *rows-processed*:integer,  
*cursor-position*: integer

Argument	Description
<i>cursor-object</i>	The cursor object that defines this query. For information about how to create cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>user-object</i>	A user-defined object that currently exists in G2.
<i>update-action</i>	Specify REPLACE to cause the database bridge to replace the data in the specified object with the data resulting from the query.  Specify APPEND to cause the database bridge to append to the attribute in the specified object, the data resulting from the query. If APPEND is used, the attribute being appended must be a list or an array.

Argument	Description
<i>rcds-to-fetch</i>	<p>The maximum number of records (rows) to fetch from the database cursor in this execution of <code>db-update-object</code>.</p> <p>If you specify 0 for <i>rcds-to-fetch</i>, the call to <code>db-update-object</code> returns all the rows in the database cursor.</p> <p><b>Note:</b> The <code>MAXROWS</code> bridge startup option specifies the default maximum number of database rows that a single transaction can return. If <i>rcds-to-fetch</i> specifies a value greater than the value of <code>MAXROWS</code>, the G2-Database bridge returns a warning and the transaction ends after the bridge returns the number of rows specified by <code>MAXROWS</code>. To get the remainder of rows in the cursor, execute <code>db-fetch-object</code> repeatedly until the bridge returns <code>ecursor</code>. For information about the <code>MAXROWS</code> option, see <a href="#">Initial Bridge Memory Requirements</a>.</p>
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status, code, message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of database rows processed by the transaction.
<i>cursor-position</i>	The number of the last row returned from the database cursor.

## Description

The procedure `db-update-object` performs the database query defined by a specified *cursor-object*. Depending on the value that you specify for *update-action*, the bridge will either replace or append the data within an existing object. For example, if the attributes of the object are lists and you specify *replace* as the *update-action*, then the values within the list attributes in the object are replaced with new values from the query. If you specify *append* as the *update-action*, then the values from the query will be appended to the end of the list attributes of the object. The original values will not be modified.

---

**Note** If you change your query and the query references new database columns, only the values from the columns of the query will be updated in the object. Values from any other columns will not be modified.

---

Before you can call `db-update-object` to execute the query defined by `cursor-object`, you must set values for the bind variables in the SQL statement defined in that `cursor-object`. You can set values for the bind variables by calling either `db-set-cursor` or `db-define-cursor`.

When retrieving INTEGER or DECIMAL types and “smart fetch” is enabled:

- When an attribute of `user-object` is an integer, the corresponding values are returned as integers. If any value is too large or small to fit in a G2 integer (the upper and lower limits are 536,870,911 and -536,870,912, respectively), the procedure call returns an error.
- When an attribute of `user-object` is a float, the value is returned as a float.
- When an attribute of `user-object` is a quantity, the corresponding values that will fit in a G2 integer are returned as integers; those that will not are returned as floats.

For more information, see [Using Smart Fetch](#).

When “smart fetch” is not enabled, when attribute of `user-object` is an integer, an error is generated if any value would cause G2 integer overflow.

When no records are added to the `user-object` due to an error, the `rows-processed` return value is 0 and the `cursor-position` return value is not updated.

## User-Defined Object Data Types

When you define objects in G2, certain data types for attributes may not be fully supported for population with database data. The following table summarizes G2 data types for object attributes and the corresponding bridge support level.

Data Type	Support Level
Simple values (text, integer, float, etc.)	Full support.
Compound values (sequences, structures)	Support for sequences only. In addition, only the <code>replace update-action</code> is supported.
Lists and arrays	Full support.
Variables and parameters	Full support.
User-defined objects	No support.

## Related Procedures

[db-define-cursor](#)

Creates a re-usable cursor object that defines or redefines a query and provides a reference to the database cursor for that query.

[db-set-cursor](#)

Sets or changes the values of bind variables in an existing cursor object.

[db-fetch-object](#)

Performs a query and returns the results to G2 within a user specified object.

# db-update-query-item

Returns data associated with a database cursor to an existing query item in G2. The data within the query item may be either replaced or appended.

## Synopsis

db-update-query-item

(*cursor-object*:class db-cursor-object,  
*query-item*:class db-query-item, *update-action*:symbol,  
*rcds-to-fetch*:integer, *interface*:class g2-database-interface)  
-> *status*, *code*, *message*,  
*rows-processed*:integer, *cursor-position*:integer

Argument	Description
<i>cursor-object</i>	The cursor object that defines this query. For information about how to create and define cursor objects, see <a href="#">Creating a Cursor Object</a> .
<i>query-item</i>	A db-query-item that currently exists in G2.
<i>update-action</i>	Specify REPLACE to cause the database bridge to replace the data in the specified query item with the data resulting from the query.  Specify APPEND to cause the database bridge to append to the attribute in the specified object, the data resulting from the query. If APPEND is used, the attribute being appended must be a list or an array.

Argument	Description
<i>rcds-to-fetch</i>	<p>The maximum number of records (rows) to fetch from the database cursor in this execution of <code>db-fetch-query-item</code>.</p> <p>If you specify 0 for <i>rcds-to-fetch</i>, the call to <code>db-fetch-query-item</code> returns all the rows in the database cursor.</p> <p><b>Note:</b> The <code>MAXROWS</code> bridge startup option specifies the default maximum number of database rows that a single transaction can return. If <i>rcds-to-fetch</i> specifies a value greater than the value of <code>MAXROWS</code>, the G2-Database bridge returns a warning and the transaction ends after the bridge returns the number of rows specified by <code>MAXROWS</code>. To get the remainder of rows in the cursor, execute <code>db-fetch-query-item</code> repeatedly until the bridge returns <code>ecursor</code>. For information about the <code>MAXROWS</code> option, see <a href="#">Initial Bridge Memory Requirements</a>.</p>
<i>interface</i>	The G2-Database interface object that configures this connection to the database bridge. Specify the name of an existing G2-Database interface object.
Return Value	Description
<i>status, code, message</i>	For information about these return arguments, see <a href="#">Invoking G2-Database Bridge Procedures</a> .
<i>rows-processed</i>	The number of database rows processed by the transaction.
<i>cursor-position</i>	The number of the last row returned from the database cursor.

## Description

The procedure `db-update-query-item` performs the database query defined by a specified *cursor-object*. Depending on the value that you specify for *update-action*, the bridge will either replace or append the data within an existing query item. For example, if the query item is a `db-query-item-list` and you specify `replace` as the update-action, then the values within the list attributes in the query item are replaced with new values from the query. If you specify `append` as the update-action, then the values from the query will be appended to the end of the list attributes of the query item. The original values will not be modified.

---

**Note** If you change your query and the query references new database columns, then only the values from the columns of the query will be updated in the query item. Values from any other columns will not be modified.

---

When retrieving INTEGER or DECIMAL types:

- When *update-action* = **replace**, the values are returned as an array or list of quantities, where individual values are returned as integers, or as floats if they would cause G2 integer overflow.
- When *update-action* = **append**
  - When an attribute of the *query-item* is an integer, the corresponding values are returned as integers. If any value is too large or small to fit in a G2 integer, that value is returned as the maximum or minimum valid G2 integer (which have values 536,870,911 and -536,870,912, respectively) and an error is returned by the procedure call.
  - When an attribute of the *query-item* is a float, the values are returned as floats.
  - When attribute of *query-item* is a quantity, values that will fit in a G2 integer are returned as integers; those that will not are returned as floats.

For more information, see [Using Smart Fetch](#).

This procedure:

- Requires the *query-item* to have the same number of columns as the query in the cursor and the column names to match and be in the same order.
- Generates an error if there is an incompatibility between a value returned by the query and the type of the array or list that should hold it.

## Related Procedures

[db-fetch-query-item](#) Performs a database query and returns the results to G2 within a query item.

# Message Handling

---

*Describes the G2-Database bridge error and message handling facility.*

Introduction	173
Handling Messages	174
Trigger Events	176
Redirecting Messages	178
Saving Messages in Log Files	178



## Introduction

All G2-Database bridges return information about errors and other events that result from the execution of SQL statements or bridge operations. G2-Database bridges also provide an error and message handling facility to help you identify and respond to the reported events.

G2-Database bridges can report messages in the following ways:

- All bridge procedures pass information to return values (G2 variables) of the procedure call. This information includes the status value, code, and message text. Some procedures return the number of rows processed by the database, the cursor position, and other values.
- SQL Select statements that use Query Objects return information to the following attributes of the Query Objects associated with the event for which the error is reported: `db-cursor-position`, `db-status`, `db-code`, `db-message`, and `db-rows-processed`. The information in these attributes duplicates the information returned by the procedure called.

- Bridge messages are sent to a log file if you specify a file name in the `log-file` attribute of your G2-Database interface object before the bridge is connected or before you call `db-configuration`. You can open a log file at any time while your bridge is connected by calling `db-logfile`. You can specify that only certain kinds of messages be reported to the log file. You can add time stamps to messages reported to the log file.
- The bridge calls the procedure `db-context-event-msg` to report messages to G2 if the `enable-messaging` attribute of the G2-Database interface object is set to `true`. You can edit the definition of this procedure to modify how messages are handled within your G2 knowledge base or you can specify a user-defined procedure that will receive messages.
- The bridge calls the procedure `db-trigger-event` to report events that occur within the database if a previous trigger watch has been defined. You can edit the definition of this procedure to modify how events are handled within your G2 knowledge base or you can specify a user-defined procedure that will receive trigger event messages.

## Handling Messages

G2 receives unsolicited reports of certain events that occur in the bridge or database. The reports include status information and messages that describe the events. Each report refers to an event within a particular context, which is identified in the report.

Within G2, reports of events are received by the procedure `db-context-event-msg`. The elements of each report — such as the status value, code, and message text — are stored in separate arguments of `db-context-event-msg`. These elements of the report can be accessed by your G2 application.

You must edit the procedure `db-context-event-msg` to specify how the procedure handles events. For more information see the description of [db-context-event-msg](#).

## Enabling and Disabling Message Reporting

Message reporting by `db-context-event-msg` is initially enabled or disabled for a connection when the connection becomes active and the configuration information in the G2-Database interface object for that connection is sent to the bridge.

After the connection becomes active, you can enable or disable message reporting through `db-context-event-msg`.

**To enable or disable messages on an active connection:**

- 1 Disable the G2-Database interface object.
- 2 Change the `enable-messaging` attribute of the G2-Database interface object to `true` (enable) or `false` (disable).
- 3 Re-enable the G2-Database interface object or reset your G2 KB.

If you disable `db-context-event-msg` or modify it, the bridge still sets the four status attributes of Query Objects and still returns the results of procedure calls.

---

**Note** Do not attempt to use the `db-configuration` procedure during a query to enable or disable message reporting by `db-context-event-msg`. The G2-Database bridge will reject your attempt to reconfigure the context and return a warning message.

---

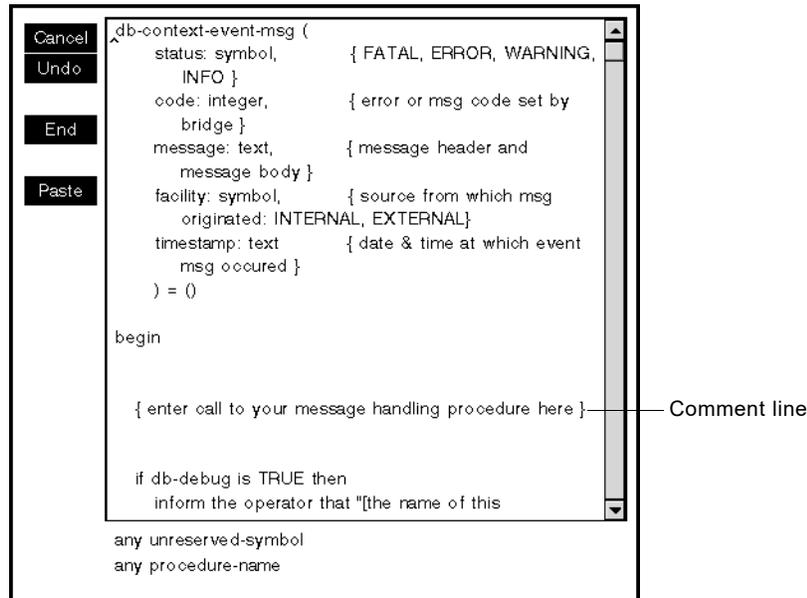
## Editing Messages

You must edit the default definition of `db-context-event-msg` to specify how messages are handled. If you do not edit the default definition, the procedure is called but the message is ignored.

**To edit messages:**

- 1 Click the Procedures button on the `g2-database` workspace.
- 2 Click the Error & Message Handling button on the G2-Database Procedures workspace.
- 3 Ctrl-click the `db-context-event-msg` procedure icon to display the definition.

For example:



```
db-context-event-msg (  
  status: symbol,      { FATAL, ERROR, WARNING,  
                       INFO }  
  code: integer,       { error or msg code set by  
                       bridge }  
  message: text,       { message header and  
                       message body }  
  facility: symbol,    { source from which msg  
                       originated: INTERNAL, EXTERNAL}  
  timestamp: text      { date & time at which event  
                       msg occurred }  
  ) = ()  
  
begin  
  
  { enter call to your message handling procedure here }  
  
  if db-debug is TRUE then  
    inform the operator that "[the name of this  
any unreserved-symbol  
any procedure-name
```

- 4 Replace the following comment line with code that specifies how you want `db-context-event-msg` to handle messages.

{ enter call to your message handling procedure here }

For example, to display on the Message Board the values sent by the bridge to the *status*, *facility*, and *code* arguments, replace the comment line with the following line:

```
inform the operator that "db-context-event-msg status = [status],  
facility = [facility], code = [code]";
```

If you have a message management facility, you can call it here. You can reference the G2 date-time string in `db-context-event-msg` to add a time stamp to messages.

## Trigger Events

Through a G2-Database Bridge, G2 can receive messages from database triggers. The database administrator can define events that affect specific objects, such as tables, as database triggers.

A text message and timestamp are associated with each trigger. You can cause the trigger messages and timestamps to be returned to G2 procedures or to G2 trigger objects.

## Returning Trigger Messages to a G2 Procedure

To return trigger messages to a G2 procedure:

- 1 Specify the name of a database-defined trigger that G2-Database Bridge will watch for trigger messages, using the procedure `db-set-trigger`.
- 2 Edit the `db-trigger-event` procedure to specify how G2 responds to the trigger message.

This procedure can pass the text and timestamp of trigger messages to G2 procedures that process the information in the messages as required by your application. For example:

```
db-trigger-event(trigger-name, trigger-msg, trigger-time)
```

## Returning Trigger Messages to a Trigger Object

To return trigger messages to a G2 trigger object you must:

- Create a G2 trigger object that will receive the message of a specified trigger.
- Specify the trigger object to which you want the G2-Database Bridge to return the trigger message.

To create a trigger object:

- 1 Click the SQL Object Classes button on the G2-DATABASE workspace to display the G2-Database SQL Object Classes workspace.
- 2 Choose `create instance` from the object definition menu of the `db-trigger-object`, and place it on your workspace.
- 3 Choose `table` from the trigger object menu to display the attributes.
- 4 Specify a name for the trigger object in the `names` attribute.
- 5 Specify the name of a G2-Database interface object in the `gsi-interface-names` attribute.

The interface object that you specify must be the one that configures a connection between the G2-Database Bridge and G2.

To specify the trigger object to which the G2-Database Bridge returns the trigger message:

- Call `db-set-trigger`.

For example:

```
db-set-trigger(my-trigger-object, true, my-db-interface)
= status, code, message
```

where `my-trigger-object` is the name of the trigger object.

For information about `db-set-trigger`, `db-get-triggers`, and `db-trigger-event`, see [Bridge Procedures](#).

## Redirecting Messages

Messages or events intended for either `db-context-event-msg` or `db-trigger-event` can be redirected to user-defined procedures in G2 by calling `db-redirect-callback`. `db-redirect-callback` allows you to indicate which G2 procedure will receive messages initially intended for `db-context-event-msg` and which procedure will receive messages initially intended for `db-trigger-event`. Redirecting messages to your own procedures is the recommended approach since you will not have to modify the procedures provided with `g2-database.kb` in order to develop your own message handler routines. For information about `db-redirect-callback`, see [db-redirect-callback](#).

## Saving Messages in Log Files

You can create a log file to store messages reported during execution of the G2-Database bridge. The log file stores messages reported for the connection that is configured by this G2-Database interface object.

You can create, open, and close log files automatically, or through calls to the bridge procedure [db-logfile](#). For information opening log files automatically, see the discussion of the `log-file` attribute in [Attributes of G2-Database-Interface Objects](#).

Each time a log file opens, the bridge writes a header to it. The header contains the name of the bridge from which the messages originate and a timestamp. When a logfile closes, the bridge writes a footer to the logfile with a timestamp.

Logfile entries are of the following form:

```
date time | bridge-name: context-name: status: message
```

For example:

```
12-dec-96 16:00:00 | g2-oracle: my-interface: ERROR:  
Could not allocate memory
```

## Opening and Closing Log Files

You can cause a log file to be opened automatically when G2 establishes a connection to the G2-Database bridge.

**To open a log file automatically:**

- ➔ Specify a full pathname for the file in the `log-file` attribute of your G2-Database interface object.

The bridge opens a log file by this name when the connection is established. If a log file by this name does not exist, the bridge creates one. For information about the `log-file` attribute, see [Attributes of G2-Database-Interface Objects](#).

When you call `db-logfile` to open a log file that is already open, the procedure first closes the log file and then reopens it. If messaging is enabled, an information message is sent to G2 and to the log file, noting that you reopened an already open log file.

Only one log file can be open in a context at a given time.

A log file closes automatically when you:

- Reset the knowledge base.
- Disable the interface object that uses the log file.
- Kill the bridge process by calling the `db-kill-bridge` procedure.

The log file also closes when you pause the knowledge base. Resuming the knowledge base reopens the log file.

## Accessing the Log File

You must close a logfile during a bridge operation before you can read, edit, or print it. You can do this by pausing the knowledge base, or by calling the `db-logfile` procedure with the `close` option.

## Filtering Log File Entries

You can specify default filters for log file entries as part of the `log-file` attribute of the G2-Database interface object. See [Attributes of G2-Database-Interface Objects](#).

If you need to override the default filters at any time during execution of the bridge, you can call the `db-logfile` procedure to open the log file with new filters. When `db-logfile` is called to reopen a log file, the filter specified in the call to `db-logfile` supersedes any filter specified in the `log-file` attribute of the G2-Database interface object or in any previous call to `db-logfile`.

If you call `db-logfile` without specifying filters in the call, the default filters specified by the G2-Database interface object are used.

If you do not specify filters either in the `log-file` attribute of the G2-Database interface object or in a call to `db-logfile`, messages in all categories are logged.



# Troubleshooting

---

*Describes common problems that you may encounter when you run a G2-Database bridge, and describes solutions for each problem.*

Introduction **181**

You Cannot Make Connections **182**

Query Does Not Return Expected Values **183**

Deadlocks - Hung or Not Responding Bridge **185**

Other Unexpected Behaviors **187**

Debugging Facility **187**



## Introduction

The following sections describe categories of common problems and the recommended solutions or workarounds for each problem.

# You Cannot Make Connections

**Problem** You cannot make a successful connection to the database.

---

- Solution**
- Make sure that your bridge is connected to G2. If the bridge is connected, the `gsi-interface-status` attribute of the G2-Database interface object is set to 2.
  - Make sure that you have set the `db-user`, `db-password`, and `db-connect-string` attributes of the G2-Database interface object to valid values. See the Release Notes for your particular G2-Database bridge.
  - Make sure that you have not exceeded the maximum number of connections allowed to a single copy of a bridge (50).
  - Your database may require additional time to establish the connection.

–

**Problem** You cannot make a successful connection to your bridge from G2.

---

- Solution**
- You enabled your G2-Database interface object before the bridge was running. Start your bridge before you enable your G2-Database interface object.
  - Your request to make a connection is taking longer than the specified timeout period. Increase the value specified for the `interface-timeout-period` attribute of your G2-Database interface object.
  - The `gsi-connection-configuration` attribute of your G2-Database interface object does not specify the same TCP/IP port number that you used in the command to start the bridge. Note that if you failed to specify either a TCP/IP port number in the start command, the bridge uses default values for these parameters.

## Query Does Not Return Expected Values

<b>Problem</b>	Query does not return data for all columns selected.
<b>Solution</b>	<ul style="list-style-type: none"> <li>• The data type of a column attribute may not correspond to the column attribute type. For example, you query a float value and the corresponding column attribute is defined as a text-parameter.</li> <li>• A column attribute does not exist in the query object.</li> <li>• A column attribute is incorrectly named in the query object.</li> <li>• The data value was NULL.</li> </ul>
<b>Problem</b>	The column attribute is defined as an integer, but the database returned a float value.
<b>Solution</b>	Try redefining the column attribute as a quantity.
<b>Problem</b>	db-fetch-records receives EOCURSОР status but no data.
<b>Solution</b>	Your query resulted in 0 records or you have already fetched all records in the cursor. Call db-refresh-cursor before you call db-fetch-records again.
<b>Problem</b>	db-sql-function does not receive a value.
<b>Solution</b>	Check both return values of db-sql-function, num-val and text-val. The num-val argument receives a value if the result is numeric. The text-val argument receives a value if the result is text.
<b>Problem</b>	db-sql-function does not receive all columns selected or all rows.
<b>Solution</b>	This procedure is designed to use SQL functions to return a single value.

**Problem** When querying data that you think is of type integer, you receive the data as float or quantity type data.

---

**Solution** Your database may support scaled integers or number data types with precision and scale. Neither the bridge nor G2 supports scaled integers, and they convert scaled integers to float.

**Problem** Blank, padded textual data is not being returned properly.

---

**Solution** Blanks will only be returned as a part of textual data for certain database types. This rule varies from one database to another. Refer to your database reference manual for detailed information about text data types.

**Problem** After querying from your database while doing a join with multiple tables, certain column attributes have twice or more than twice the number of values in other column attributes that were fetched by the same query.

---

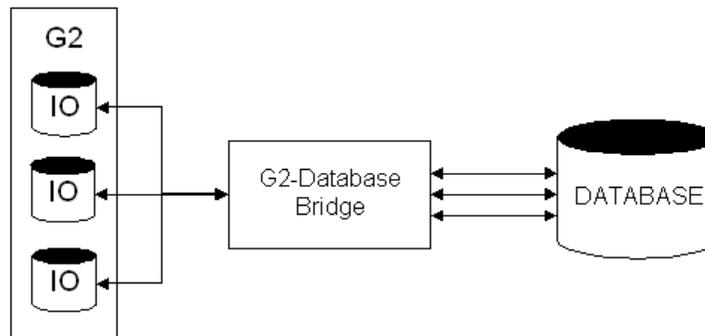
**Solution** You may have queried two columns from different tables which have the same name. Since G2 places data into column attributes by column name, more than one column of data with the same name may be appended to your column attribute list.

You can avoid this problem by carefully constructing join queries, using column aliases in your SQL Select statement. For example: `select proc1.parts, proc2.parts from proc1, proc2;` will produce two columns in the cursor named `parts`. Both columns of data will be appended to the column attribute named `parts`. If you use column aliases, for example: `select proc1.parts 'parts1', proc2.parts 'parts2' from proc1, proc2;` the resulting data will be in two columns, `parts1` and `parts2`, and column attributes named `parts1` and `parts2` will each receive the corresponding column of data.

## Deadlocks - Hung or Not Responding Bridge

An attempt to access database data which is locked (i.e. via row locks, table locks, etc.) may result in a deadlock condition. A symptom of this condition is a *hanging* of the bridge process, expressed as an inability to perform any transaction across any database connection. Due to the **single-threadedness** of GSI, transactions across all contexts (i.e. database connections) are processed sequentially.

If your application supports multiple contexts, that attempt to access the same data simultaneously, it is at risk of generating a deadlock because access to locked data may be attempted before the lock is released via a commit or rollback. To avoid deadlocks use one of the following approaches when using multiple contexts (multiple interface objects connecting to the same bridge).



- **Use the *for update nowait* SQL syntax.**

It is recommended that all *for update* query operations contain the SQL syntax *for update nowait* or similar syntax that prevents a transaction from waiting for a locked resource to become available. This will result in the query being rejected and an error message being returned to G2 if access to locked data is attempted. Users may then be notified and the transaction may be resubmitted at some predetermined time interval.

The following example illustrates a query that will abort if the data being accessed is locked:

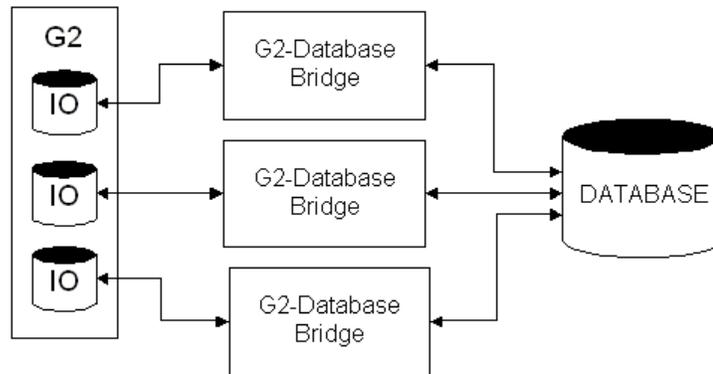
```
select * from emp for update nowait;
```

See "SELECT" in the Oracle7 Server SQL Language Reference Manual.

- **Provide a separate bridge process for each interface object.**

Another, more costly approach is to use a separate bridge process for each database connection (i.e. a 1-to-1 correspondence between interface object and bridge process). This approach implies that each bridge/interface object is associated with only 1 database connection. It helps to have G2 start-up the

bridge processes automatically as users login to G2 or as additional interface objects are needed.



- **Provide transaction management from within the KB.**

The KB may be designed to keep track of certain operations that may result in database data locks. You can manage user transactions from within the KB so that transactions that could result in a deadlock will either be rejected or placed on hold until the lock potential no longer exists. Do this by setting flags in the interface object that are checked before allowing certain transactions.

Warning: If a *bridge* deadlock occurs, there is no way to release the deadlock from G2 or the bridge. The bridge process must be stopped and the bridge must be restarted.

- **Utilize *auto-commits* with DML SQL transactions.**

If you take advantage of the auto-commit feature provided with various G2-database KB procedures, you can significantly reduce the incidence of bridge deadlocks. The auto-commit feature will automatically commit after it executes. This removes the requirement to manually invoke the db-commit procedure to commit the transaction, and has the added advantage of reducing the number of database transactions by ½.

- **Embed commit within SQL statement.**

You can embed a commit statement within your SQL transaction. This will have the same effect as if you utilized the auto-commit feature described above. By embedding the commit within your SQL transaction, the transaction will automatically commit following successful execution. This removes the requirement to manually invoke the db-commit procedure to commit the transaction, and has the added advantage of reducing the number of database transactions by ½.

## Other Unexpected Behaviors

<b>Problem</b>	The db-fetch-record procedure returns a status value (such as SUCCESS) before it returns all of your data records.
<b>Solution</b>	<p>You have a G2 priority problem. In general, to ensure that all records are returned to your query object before the completion status is returned from db-fetch-records, set the priority of the data server parameter higher (that is, numerically lower) than the priority of db-fetch-records and any procedure that calls it.</p> <p>To set the priority of the data server, choose Main Menu &gt; System Tables &gt; Data Server Parameters and specify a priority value in the priority-of-data-service field of the Data Server Parameters table.</p> <p>To set the priority of a procedure, open the table for the procedure and specify a priority value in the default-procedure-priority attribute.</p>
<b>Problem</b>	When querying either by calling db-fetch-records or using automatic updates by G2, you receive the following error: 'SQL statement has not been processed.'
<b>Solution</b>	You may be trying to fetch data into a query object or query object attributes that have been disabled.
<b>Problem</b>	G2 aborts after a call to db-fetch-records.
<b>Solution</b>	You may be trying to fetch data into a query object or query object attributes that have been disabled.

## Debugging Facility

It's possible to set -d option in **remote-process-initialization-string** attribute of **g2-database-interface** objects to make all G2-Database bridges printing debug informations. Following flags are supported:

<b>Flag</b>	<b>Debug output</b>
DBG0	no debug
DBG1	function names.
DBG2	info
DBG3	errors
DBG4	warnings
DBG5	results
DBG6	info detail (SQL statements)
DBG7	debug tools & handles
DBG8	config & setup
DBG9	memory
DBGM	fatal
DBGC	gsi callback functions
DBGALL	DBG1->DBGC
DBGX	disable GSI error handler
DBG10	descriptor dump
DBG11	memory leak
DBG12	reserved
DBG13	reserved

---

**Note** Use of these debugging flags may slow down the performance of bridges.

---

# Performance

---

*Describes several ways to improve the performance of your G2-Database Bridge application.*

- Introduction **189**
- Complicated Queries **190**
- Data Service Priority **190**
- Distribution of Bridge Load **190**
- Network Considerations **191**
- Object Passing **191**
- Bind Variables **191**



## Introduction

You can improve the performance of the bridge and your G2 knowledge base by tuning your database for best performance. See you database administrator and database reference manual for information about how to tune your database.

## Complicated Queries

For complicated or cross-join queries, especially those you use repeatedly:

- Try a database view.

This will greatly reduce database processing at run-time.

- Reuse existing query objects.

The database caches cursors and reuses existing cursors before it prepares new queries. Use `db-refresh-cursor` to reuse an existing cursor.

Strings may use more resources and therefore take more time to query.

- Select only the columns and data that you need.

## Data Service Priority

You may increase the G2 system parameter for priority of data service in order to provide a higher (lower-numbered) priority. (The default priority is 4.) However, doing this may affect the overall performance of your G2 application. Experiment to find out which priority is best for your application.

## Distribution of Bridge Load

A single running copy of the G2-Database bridge can process only one G2 request at a time. Thus, if all database operations are using one bridge, a large query can cause delays in the execution of other operations.

To increase processing speed, you can run multiple copies of a G2-Database bridge and distribute requests from G2 among the different copies. When you run several copies of a bridge, one copy can process one request while another copy is processing another request.

For example, you can distribute the load on the bridges by performing large, slow queries on one bridge and performing quicker, more urgent queries on another bridge. Or, you can use one bridge for querying and reading, and another bridge for inserting, modifying, and writing.

Running multiple copies of a bridge can increase efficiency in cases where the G2 knowledge base is processing a large volume of transactions, or where several G2 knowledge bases are using the bridge for connections to the database. However, the performance of G2 may be affected if it cannot keep up with the data being returned by several different bridges.

For information on starting multiple copies of a G2-Database bridge, see [Running Multiple Copies of a Bridge](#).

## Network Considerations

A system running G2, the database bridge, and a database can become overloaded because of inadequate memory or speed. In this case, you can improve performance by running the database and the bridge remotely from G2. The performance loss that results from remote operation may be significantly offset by the performance gain of distributing your processes.

## Object Passing

G2-Database bridges can use G2's object-passing ability to increase the efficiency and throughput of your G2 application's database transactions. Your G2 application can pass multiple values to the G2-Database bridge in a single transaction, by passing the values to the bridge in attributes of a single G2 object.

Your G2 can use object-passing in both non-query and query SQL operations:

- Your G2 application can perform a non-query SQL operation on multiple values by passing to the bridge a single G2 object that contains these values in a list or an array. For information about how to do this, see [Database Operations Using Objects](#).
- Your G2 application can perform a query for multiple values and receive the results in a single G2 object that contains the queried values in lists or arrays. For information about how to perform queries, see [Querying the Database](#).

## Bind Variables

To increase the flexibility and efficiency of the database operations, you can use bind variables in SQL statements. You assign values to the bind variables to specify the data that you want to insert, delete, or update.

You can change the values of bind variables in a previously-executed SQL statement without forcing the database to generate a new execution plan for the SQL statement when you execute it again. However, if you make changes to a SQL statement itself (that is, if you change the sequence of characters, case, spaces, or punctuation in the statement) before you execute it again, you force the database to generate a new execution plan for the SQL statement and incur other overhead.



# Bridge Data Types

---

*Lists common database data types and the G2 data types to which they correspond.*

Introduction 193



## Introduction

The following table lists common database data types and the types of the corresponding values that the G2-Database bridge returns to your G2 application.

Use the bridge data types, rather than the database data types, when you define column attributes in the `attributes-specific-to-class` attribute of an object definition for query objects.

Database Data Type	G2 Data Type
<code>asciz</code>	text
<code>bigint (64 bits)</code>	float or quantity
<code>binary (n)</code>	Not supported.
<code>bit</code>	integer or quantity
<code>bit (n)</code>	Not supported.
<code>bit varying (n)</code>	Not supported.
<code>byte</code>	Not supported.
<code>char</code>	text

<b>Database Data Type</b>	<b>G2 Data Type</b>
date	text
date ansi	text
datetime	text
decimal	integer or quantity
decimal (p)	integer or float
decimal (p,s)	integer or float
double precision	float or quantity
float (p)	float or quantity
integer	integer or quantity
interval	text
money	quantity or float
nationalchar, nchar	text
national varchar	text
number	integer or quantity
number (p)	integer or float
number (p,s)	integer or float
number (0,0)	float or quantity
numeric (p)	integer or quantity
numeric (p,s)	float or quantity
quadword	integer or quantity
real	float or quantity
serial	integer
smallfloat	float or quantity
smalldatetime	text
smallint	integer or quantity

Database Data Type	G2 Data Type
small-money	quantity or float
string	text
text	text
time	text
timestamp	text
tinyint	integer or quantity
varbinary (n)	Not supported.
varchar	text
varchar2	text

Your database may not support all of these data types. See your database documentation for a list of supported data types.

If a database function (such as max, min, sum) is performed on a number (p, s) or decimal (p, s) column, the result is a G2 float if either the scale is non-zero or the precision is 9 or greater. The query object attribute must then be of type float or quantity or, if the scale is 0, "smart fetch" must be enabled. If the precision is 8 or less and the scale is 0, the bridge returns a G2 integer.

For Oracle, the default precision and scale for the INTEGER data type is (38,0); therefore, the result is a float.

Any integer greater than 536,870,911 or less than -536,870,912 (that is, 29 bits) will be converted to a float by the bridge and thus must have its column attribute defined as quantity.

All 64-bit integers will be converted to floats by the bridge and thus must have their column attributes defined as float or quantity.



# Status Values

---

*Lists the status values returned by G2-Database Bridges.*

Introduction 197



## Introduction

The following table summarizes the status values that can be returned by G2-Database bridges. These status values may be returned following calls to G2-Database KB API procedures.

Status Value	Returned To	Code/Description
SUCCESS	Procedures, query objects	Code = 0 The database transaction completed successfully.
WARNING	Procedures, query objects	Code for bridge errors: 9000 to 9999. A database-specific SQL warning code may also be reported. All codes are positive.  A transaction completed successfully, but there was a database-specific warning message.

Status Value	Returned To	Code/Description
INFO	Procedures, query objects	Code: 9000 to 9999 (positive values)  For bridge errors only  The transaction completed successfully, but the bridge reports additional, useful bridge-specific information.
EOCURSOR	Procedures, query objects	Code = 9001  Code is specific to the database.  All records (rows) have been fetched, or the query returned no records.
CONNECTED	Procedure/ g2-database- interface object	Indicates that there is a connection between the bridge and database. In addition, this value is set into the database-connection-status attribute of g2-database-interface objects.
DISCONNECTED	Procedures/ g2-database- interface object	Code is specific to the database.  An attempt to establish a connection failed. Reconnect to the database before you perform any more database transactions.  In addition, this value is set into the database-connection-status attribute of g2-database-interface objects.  <b>Note:</b> You can instruct the bridge to automatically attempt to reconnect when this status is reported. See the description of Auto-database-reconnect in <a href="#">Attributes of G2-Database-Interface Objects</a> .

<b>Status Value</b>	<b>Returned To</b>	<b>Code/Description</b>
ERROR	Procedures, query objects	Code for bridge errors: -9000 to -9999. Codes for database errors are specific to the database. All codes are negative.  A transaction did not complete successfully.
FATAL	Procedures, query objects	A non-recoverable error occurred. It may not be possible to continue to run the bridge.



A B C D E F G H I J K L M  
N O P Q R S T U V W X Y Z

---

## A

**auto-database-reconnect:** An attribute of Interface Objects. Specifies whether the bridge automatically attempts to reestablish a connection to the database if the connection is broken.

## B

**bind variable:** A variable in a SQL statement that identifies an item of data on which the transaction defined by the SQL statement is to be performed. You can change the values of bind variables in a previously-executed SQL statement without forcing the database to generate a new execution plan for the SQL statement when you re-execute it. Bind variables are required in SQL statements that define database operations on lists or arrays of values. The syntax that you must use to identify bind variables in SQL statements is specific to each database.

## C

**class-name attribute:** An attribute of object definitions. For Query Objects, this attribute should describe the kind of data for which Query Objects created from this object definition will query.

**class-specific-attributes:** An attribute of object definitions. For Query Objects, this attribute specifies the database columns that can be queried for data by Query Objects created from this object definition.

**column alias:** An alternate name for a database column. Column aliases can be used in the **class-specific-attributes** of object definitions for Query Objects. See your bridge-specific Release Notes for details.

**column attributes:** Attributes of Query Objects. The column attributes represent the names (or aliases) of the database columns that are queried by the Query Objects. Column attributes are defined in the **class-specific-attributes** attribute of the object definition of the Query Objects.

**context:** A connection between the bridge and a G2 application. A single bridge instance (process) can support more than one context. All requests that G2 makes of the bridge are processed within a context. You specify a name for each context in the **context name** parameter of the Interface Object that configures the connection. The **db-context-event-msg** procedure includes the context name in messages that it sends to G2 or to a log file.

**cursor:** A temporary relation or result table from a database query that can be accessed by a program. Since SQL operates on relations (sets of records or tables) and programs operate on records, this mechanism provides access to the result data from a query, one record at a time. A cursor must be opened before it can be dynamically accessed. The bridge creates and manages cursors.

**cursor object:** A G2 object that contains a SQL statement defining a database query. The cursor object provides a reference to the database cursor that the database generates when you execute the query. You create cursor objects by using the procedure `db-define-cursor`. Cursor objects are referenced by the procedures `db-fetch-object`, `db-fetch-query-item`, `db-fetch-records`, `db-set-cursor`, `db-refresh-cursor`, `db-fetch-structure`, `db-update-query-item`, and `db-update-object`.

## D

**Data Definition Language (DDL):** SQL statements used to define the data dictionary or schema. Included in ANSI standard SQL.

**Data Manipulation Language (DML):** SQL statements used to manage data, such as Select, Insert, Update, and Delete. Included in ANSI standard SQL.

**database-connect-string:** An attribute of Interface Objects. This attribute specifies the connect string for a particular database. See your bridge-specific Release Notes for details.

**database cursor:** An internal table generated by a database that lists the rows and columns included in a query or other SQL operation. G2 references database cursors using cursor objects (for queries) and SQL objects (for non-query database operations).

**database-password:** An attribute of Interface Objects. The database user password that the bridge uses to build the complete database connection string. The bridge uses this string to establish a connection to the database.

**database-user:** An attribute of Interface Objects. The database user name that the bridge uses to build the complete database connection string. The bridge uses this string to establish a connection to the database.

**db-code:** An attribute of Query Objects. The bridge uses this attribute to return the result code generated by the database from the SQL query you write, or to display a bridge error code.

**db-message:** An attribute of Query Objects. The bridge uses this attribute to display messages associated with the value in `db-code`.

**db-record-count:** An attribute of Query Objects. The bridge uses this attribute to display the row number of the records fetched into the Query Object. If the Query Object contains only 1 record (row), this attribute represents the row ID. If the Query Object contains more than one record (in lists), this attribute represents the total number of rows fetched and placed in the Query Object.

**db-rows-processed:** An attribute of Query Objects. The bridge uses this attribute to display the number of rows processed by the query.

**db-status:** An attribute of Query Objects. The bridge uses this attribute to display the status of the result of a query or SQL statement execution.

**default-update-interval:** An attribute of Query Objects. This attribute specifies an update interval the bridge can use in conjunction with the value in **external-system-has-a-scheduler** (in the Interface Object) to determine how often data is fetched into the Query Object.

**direct-superior-classes:** An attribute of object definitions. For Query Objects, use this attribute to specify the superior class of the Query Object you are defining.

## E

**enable-messaging:** An attribute of Interface Objects. Enables or disables the reporting of error, information, and warning messages by the bridge to G2.

**external-system-has-a-scheduler:** An attribute of Interface Objects. Enter an attribute value to use in conjunction with the value in **default-update-interval** attribute (in the Query Object) to specify how often you will be fetching data into the Query Object.

## G

**grouping-specification:** An attribute of Interface Objects. This attribute is not used.

**gsi-connection-configuration:** An attribute of Interface Objects. This attribute specifies: (1) which network protocol will be used to communicate with the bridge, (2) the host name of the machine running the bridge, and (3) which TCP/IP port can be used to identify the bridge process.

**gsi-interface-status:** An attribute of Interface Objects. The bridge uses this attribute to display a code that represents the status of the connection between G2 and the bridge.

**gsi-interface-object:** An attribute of Query Objects. This attribute specifies the name of the Interface Object used by this Query Object to communicate with the bridge.

## I

**identifying-attributes:** An attribute of Interface Objects. You must always set this attribute to names.

**interface object:** A GSI interface object that you create and define to supply network routing and miscellaneous information needed by the bridge to control data exchanges between G2, the bridge, and the database over one connection.

**interface-timeout-period:** An attribute of Interface Objects. If a connection is not established within the amount of time specified by this attribute, G2 assumes that the connection is timed-out and stops sending requests to the bridge over the connection configured by the Interface Object.

## M

**maximum-definable-cursors:** An attribute of Interface Objects. Defines the maximum number of open cursors allowed for the context defined by this Interface Object. See your bridge-specific Release Notes for details.

## N

**names:** An attribute of most objects. For Interface Objects, this attribute specifies an identifying attribute for the Interface Object. You use this name as the `gsi-interface-name` attribute of Query Objects to indicate that the Query Objects will exchange data with the database through this Interface Object.

## O

**object-handle:** An attribute of all objects. For Query Objects, this attribute is used internally by the bridge to associate a cursor and its data with a Query Object.

## P

**poll-external-system-for-data:** An attribute of Interface Objects. This attribute specifies whether or not G2 should poll the bridge once per cycle to check for a successful connection.

## Q

**query:** An SQL Select statement.

**query object:** A G2-Database bridge object that is mapped to a database cursor, and ultimately receives the result record(s) of data from a database cursor.

## R

**record:** One row of data (in one or more columns). A cursor contains records that result from a query. Records are fetched into one or more column attributes of a Query Object.

**remote-process-initialization-string:** An attribute of Interface Objects. Turns on or off the reporting of bridge initialization messages on the console window where you start the bridge process.

**result table:** A set of records generated as a result of a database query. A cursor is a result table stored in memory and accessible by programs.

## S

**SQL:** Structured Query Language. A computer language that is used by most database vendors to define, manipulate, and extract data from databases. Commands are usually classified as being either Data Definition Language (DDL) commands or Data Manipulation Language (DML) commands.

**sql-query:** An attribute of Query Objects.

**SQL SELECT:** An SQL statement that returns a table of records. The statement performs a query.

## T

**trigger:** A procedure that is stored in a database and executed automatically when specific events as defined by the database administrator occur. A text message and timestamp are associated with each trigger. You can cause the trigger messages and timestamps to be returned to G2 procedures or to G2 trigger objects.

**trigger object:** A G2 object that receives a message from a G2-Database bridge when a database trigger occurs.



@	A	B	C	D	E	F	G	H	I	J	K	L	M
#	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

---

**A**

- attribute definition
  - list
  - parameter
  - simple
- attributes
  - column
    - specifying
  - query item
    - copying
  - query object class definition

**B**

- bind variables
  - defined
  - effect on performance
  - in non-query SQL operations
  - in queries
  - performing SQL operations without using
- bridge process
  - and G2
    - establishing a connection
    - running multiple connections
  - changing the port number
  - command line options
    - MAXCOLS
    - MAXROWS
  - killing
  - starting

**C**

- column attributes
  - defined as list
  - defined as parameter
  - simple
  - specifying
- connections
  - changing configuration of
  - displaying status of
  - resetting

- troubleshooting
- cursor objects
  - creating
  - defined
- customer support services

**D**

- data service priority
- data types
  - in database and in G2
- database cursor
  - resetting
- database operations
  - DML
  - objects, using
  - on individual values
- database query
  - bind variables
    - using
  - components of
  - cursor objects
    - creating
  - performing
  - results
    - returned in G2 structures
    - returned in query items
    - returned to G2-objects
- database triggers
  - redirecting messages
- db-colors-object
  - attributes
- db-commit procedure
- db-configuration procedure
- db-connect procedure
- db-context-event-msg procedure
  - editing
  - enabling and disabling
  - used to handle messages
- db-define-cursor procedure
- db-define-sql procedure
- db-define-sql-obj procedure
- db-disable-all-triggers procedure

- db-disconnect procedure
- db-exec-sql procedure
- db-exec-sql-obj procedure
- db-exec-stored-proc procedure
- db-exec-stored-proc-return procedure
- db-execute-immediate procedure
- db-fetch-object procedure
- db-fetch-query-item procedure
- db-fetch-records procedure
- db-fetch-structure procedure
- db-get-triggers procedure
- db-io-status procedure
- db-kill-bridge procedure
- db-logfile procedure
- db-ping procedure
- db-ql-record, class
- db-ql-table, class
- db-query-object, class
- db-redirect-callback procedure
- db-refresh-cursor procedure
- db-rollback procedure
- db-set-connection-status procedure
- db-set-cursor procedure
- db-set-sql procedure
- db-set-sql-obj procedure
- db-set-trigger procedure
- db-sql-function procedure
- db-startup procedure
- db-text-to-text-list procedure
- db-trigger-event procedure
- db-update-object procedure
- db-update-query-item procedure
- deadlocks
  - troubleshooting
- DML database operations
  - bind variables
  - objects, using
  - performing SQL operations without using procedures
  - simple values

## E

- events
  - trigger

## F

- files
  - log

- accessing
- format
- opening and closing

## G

- G2
  - and bridge process
    - establishing a connection
    - running multiple connections
  - G2 object-passing
    - and performance
  - G2-Database bridge
    - improving performance
    - memory requirements
    - running multiple copies of
    - starting the process
    - text conversion styles
    - user modes
  - G2-Database Connection Configuration
    - workspace
  - G2-Database interface object
    - attributes
      - auto-database-reconnect
      - context-name
      - database-connection-status
      - database-connect-string
      - database-password
      - database-user
      - disable-interleaving-of-large-messages
      - enable-messaging
      - gsi-application-name
      - gsi-connection-configuration
      - gsi-interface-status
      - interface-initialization-timeout-period
      - interface-timeout-period
      - interface-warning-message-level
      - interval-to-poll-external-system
      - log-file
      - maximum-definable-cursors
      - null-number
      - null-string
      - remote-process-initialization-string
      - set-null-number
      - set-null-options
      - set-null-string
    - connection configuration
      - changing
    - connection status

- displaying
- creating
  - icon colors
    - changing with db-io-status
  - icon regions
    - example
  - introduction to
- G2-Database Notes & Information workspace
- G2-Database Procedures workspace
- G2-Database SQL Object Classes workspace
- g2-database workspace
- g2-database.kb

## I

- invoking bridge procedures

## K

- KB conflicts
  - resolving
- keyboard shortcuts

## L

- list
  - defined as column attribute
- log file
  - accessing
  - filtering entries to
  - format
  - opening and closing
  - saving messages to

## M

- MAXCOLS command line option
- MAXROWS command line option
- messages
  - code for handling
  - from trigger events
    - redirecting
    - returned to G2 procedure
    - returned to trigger object
  - handling with db-context-event-msg
    - enable and disable
  - saving in log files
  - techniques for reporting
- multiple connections

## P

- parameter
  - defined as column attribute
- performance
  - data service priority and improving
    - by running bridge remotely
  - cross-join queries
  - data service priority with bind variables
  - with multiple copies of a G2-Database bridge
    - with object passing
  - tuning a database for
- procedures
  - db-commit
  - db-configuration
  - db-connect
  - db-context-event-msg
  - db-define-cursor
  - db-define-sql
  - db-define-sql-obj
  - db-disable-all-triggers
  - db-disconnect
  - db-exec-sql
  - db-exec-sql-obj
  - db-exec-stored-proc
  - db-exec-stored-proc-return
  - db-execute-immediate
  - db-fetch-object
  - db-fetch-query-item
  - db-fetch-records
  - db-fetch-structure
  - db-get-triggers
  - db-io-status
  - db-kill-bridge
  - db-logfile
  - db-ping
  - db-redirect-callback
  - db-refresh-cursor
  - db-rollback
  - db-set-connection-status
  - db-set-cursor
  - db-set-sql
  - db-set-sql-obj
  - db-set-trigger
  - db-sql-function
  - db-startup
  - db-text-to-text-list
  - db-trigger-event

- db-update-object
- db-update-query-item
- invoking
- return values of
- summary

## Q

- queries
  - bind variables used in
  - components of
  - creating cursor objects for
  - cross-join
    - improving performance
  - returning data to G2 from
  - troubleshooting
- query items
  - copying attribute values
  - deleting
- query objects
  - attributes
    - db-code
    - db-cursor-position
    - db-message
    - db-rows-processed
    - db-status
    - gsi-interface-name
  - class definition
    - creating
  - class definition attributes
    - Attribute-initializations
    - class-name
    - class-specific-attributes
    - column
    - direct-superior-classes
  - creating
  - cursor objects
    - creating
  - db-qo-record
  - db-qo-table
  - db-query-object
  - direct superior classes
    - specifying
  - introduction to
  - performing a query

## R

- reset interface menu choice
- return values of bridge procedures

## S

- shortcuts
  - keyboard
- SQL statements
  - setting in a cursor object
    - using db-define-cursor
  - setting in a SQL object
    - using db-define-sql
    - using db-define-sql-obj
- starting bridge process
- status values
  - returned by bridge procedures
- stored procedures, executing with return values

## T

- trigger events
- trigger objects
  - creating
- troubleshooting
  - connections
  - deadlocks
  - queries
  - unexpected behaviors

## U

- UNIX
  - starting bridge process
- user modes

## W

- Windows
  - bridge process
    - starting
- workspaces
  - g2-database
    - G2-Database Connection Configuration
    - G2-Database Notes & Information
    - G2-Database Procedures
    - G2-Database SQL Object Classes