

# Janus API State Diagrams

Model 204 - Open Systems Connectivity

Version 4.4

Copyright © 1995 Sirius Software Inc.  
Doc. Rev. 4.400 October 8, 1996

## Table of Contents

Janus Server Functions .....	1
Janus Client Functions .....	5
Open Client User Language coding considerations .....	7

## Table of Figures

Janus Open Server function state diagram .....	4
Janus Open Client function state diagram .....	11
Janus Open Client function state diagram (cont'd) .....	12

## Janus Server Functions

Once a Janus Open Server port is defined and started, Sybase clients can establish a connection to the port via Sybase DB-Library Open Client function calls. The Janus Server Functions (those prefixed with `$$SRV_`) provide the facility by which the User Language server programs communicate back to the client.

When a connection is established, it is associated with an SDAEMON thread (if one is available). A Model 204 logon is then attempted for the userid and password passed by the Sybase client. If the logon succeeds, the file specified in the OPEN clause of the JANUS DEFINE command, if any, is opened. After this the command specified by the CMD clause of the JANUS DEFINE command is executed as if the user had typed this command at a terminal.

At this point the Janus thread acts much the same as any other Model 204 thread with a few exceptions. First, because the thread has no terminal associated with it, it cannot issue any terminal input requests. Any terminal input requests result in a user restart due to a "lost connection". Terminal output, on the other hand gets routed to the audit trail (as type RK journal entries). The other major difference between a Janus thread and any other thread is its ability to issue Janus server function calls.

Once a Janus thread is logged on, the thread is in a "receive" state. That is, it is up to the server to receive data from the client. The first server function a Janus thread should call is `$$SRV_WAIT`. This function indicates that the server should wait for a request (an RPC or language request) from the client. `$$SRV_WAIT` returns when a request has been received. If the client request is an RPC, `$$SRV_WAIT` reads the RPC input parameters into the thread's RPC buffer. In the case of an RPC, the thread switches to "send" state upon return from the `$$SRV_WAIT`. This means that the server is now responsible for sending data back to the client until a `$$SRV_DONE` is issued. After the `$$SRV_DONE`, the server should issue another `$$SRV_WAIT` if another RPC is to be handled.

The Model 204 \$functions that work with and respond to Sybase RPCs are

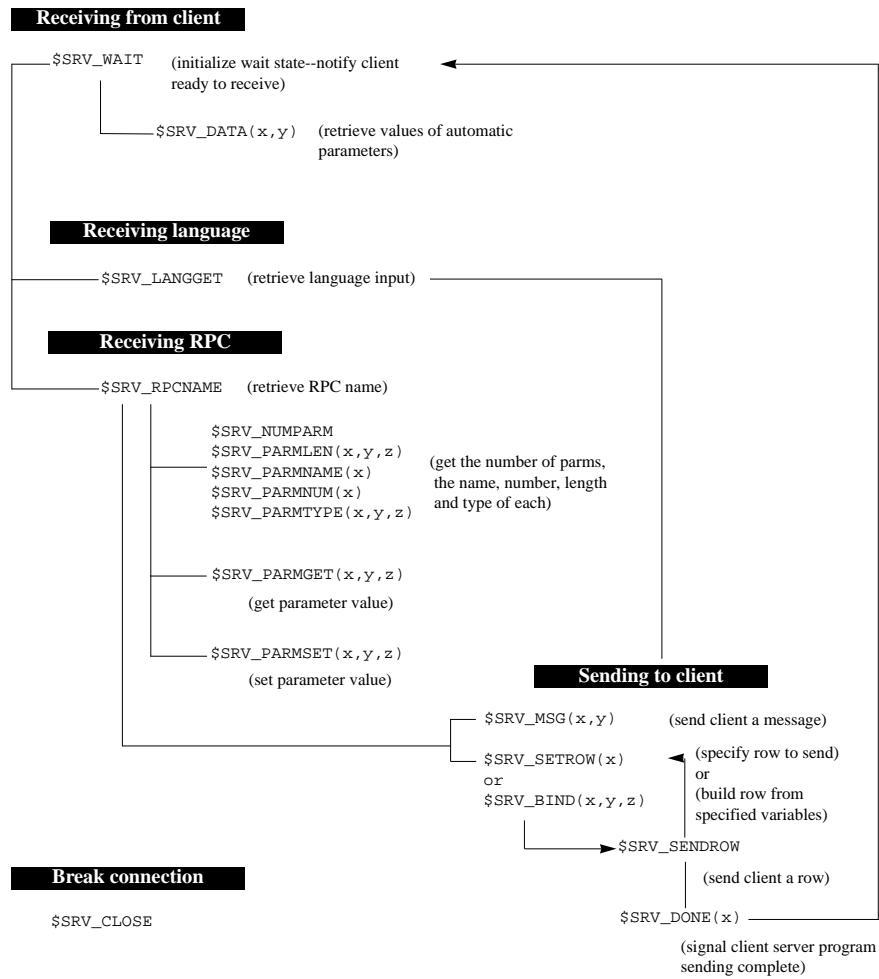
<b>\$function</b>	<b>description</b>
<b>\$SRV_BIND</b>	Bind a variable or literal to a column.
<b>\$SRV_CLOSE</b>	Close a connection.
<b>\$SRV_DATA</b>	Get the value of a parameter.
<b>\$SRV_DONE</b>	Send a "done" message to client.
<b>\$SRV_LANGGET</b>	Retrieve text of language request from client.
<b>\$SRV_MSG</b>	Send a message to the client.
<b>\$SRV_NUMPARAM</b>	Return the number of RPC parameters.
<b>\$SRV_PARMGET</b>	Return the value of an RPC parameter.
<b>\$SRV_PARMLEN</b>	Return the length of an RPC parameter.
<b>\$SRV_PARMNAME</b>	Return the parameter name of a numbered RPC parameter.
<b>\$SRV_PARMNUM</b>	Return the parameter number of a named RPC parameter.
<b>\$SRV_PARMSET</b>	Set the value of an RPC parameter.
<b>\$SRV_PARMTYPE</b>	Return the datatype of an RPC parameter.
<b>\$SRV_RPCNAME</b>	Return the name of the RPC.
<b>\$SRV_SENDRROW</b>	Send a row of data (an image) to the client.
<b>\$SRV_SETROW</b>	Set the image for subsequent rows.
<b>\$SRV_WAIT</b>	Wait for a client request or action.

These functions can only be called by User Language programs running on a Janus server thread. If any of the Janus server functions are called by a user who is not running as a Janus server thread, the results of the function will be meaningless. (See the section on Open Server User Language Coding Considerations for instructions on how to debug a Janus Open Server application).

If any of the Janus server functions are called after the client has been disconnected, a user restart will occur. Because `$SRV_CLOSE` causes the client to be disconnected, care should be taken that no further calls are placed to `$SRV_XXXXXX` functions after any call to `$SRV_CLOSE`.

Furthermore, as a user restart in APSY context will cause the server session to execute its APSY error procedure, care should be taken that no calls to `$SRV_XXXXXX` functions should ever be placed in the APSY error procedure on a logical path that is followed for user restarts resulting from a lost connection. This is the same consideration programmers would give to not coding full-screen I/O in an APSY error proc when the error proc is invoked because of a lost connection.

## Janus Open Server Function Library State Diagram



## Janus Client Functions

Janus Client Functions allow client applications, written in User Language, to place calls to a Sybase SQL Server, a Sybase Open Server or a Janus Model 204 Open Server. Client Functions are prefixed with ***\$DB\_***, and provide the facility by which User Language applications can communicate with *remote servers*.

The Model 204 \$functions that communicate with and respond to remote servers over Janus TCP/IP ports are:

<b>\$function</b>	<b>description</b>
<b>\$DB_ALTBIND</b>	Bind a Model 204 variable to an alternate column.
<b>\$DB_ALTCOLID</b>	Get the base column number that was used to create an alternate column.
<b>\$DB_ALTCOLLEN</b>	Get the length of an alternate column.
<b>\$DB_ALTCOLNAME</b>	Get the name of an alternate column.
<b>\$DB_ALTCOLOP</b>	Get the operator that was used to create an alternate column.
<b>\$DB_ALTCOLTYPE</b>	Get the type of an alternate column.
<b>\$DB_BIND</b>	Bind a Model 204 variable to a column.
<b>\$DB_CLOSE</b>	Close a connection.
<b>\$DB_COLLEN</b>	Get the length of a column.
<b>\$DB_COLNAME</b>	Get the name of a column.
<b>\$DB_COLTYPE</b>	Get the type of a column.
<b>\$DB_LANGPUT</b>	Send language data (probably SQL) to server.
<b>\$DB_MSGHANDLE</b>	Specify the message handler.
<b>\$DB_NEXTROW</b>	Get the next row from the remote server.
<b>\$DB_NUMALT</b>	Get number of alternate rows.

## Janus Client Functions

---

<b>\$DB_NUMALTCOL</b>	Get number of alternate columns.
<b>\$DB_NUMCOL</b>	Get number of columns.
<b>\$DB_NUMRET</b>	Get number of returned parameters.
<b>\$DB_ONCLOSE</b>	Specify action to take on closed connection.
<b>\$DB_OPEN</b>	Open a connection to a remote server.
<b>\$DB_RESULTS</b>	Wait for reply from server.
<b>\$DB_RETDATA</b>	Get the value of a return parameter.
<b>\$DB_RETNAME</b>	Get the name of a return parameter.
<b>\$DB_RETLEN</b>	Get the length of a return parameter.
<b>\$DB_RETSTATUS</b>	Get the return status from the last RPC.
<b>\$DB_RETTYPE</b>	Get the type of a return parameter.
<b>\$DB_RPCINIT</b>	Set up to perform a remote procedure call.
<b>\$DB_RPCPARAM</b>	Add an RPC parameter.
<b>\$DB_SEND</b>	Send request to server.
<b>\$DB_SQL</b>	Send language data (probably SQL) to server.
<b>\$DB_TEST</b>	Test connection to server.
<b>\$DB_WAIT</b>	Wait for reply from server.

## Open Client User Language coding considerations

Before the Open Client functions can be used, a JANUS port must be defined and have a remote server associated with it via the JANUS DEFINEREMOTE command.

Janus Open Server applications--applications that are front-ended on workstations and access the Model 204 online via Janus Open Server ports--will automatically use the same port they came in on for outgoing access to remote servers. For this sort of application, you must execute the JANUS DEFINEREMOTE command to associate a remote server with the port used by the incoming client application, as shown in the example below:

```

      * Specify the domain.
      *
JANUS DOMAIN sirius-software.com

      * Specify the location of the name server
      *
JANUS NAMESERVER 198.242.244.131

      * Define the port the client application
      * will come in on and the action it will take
      *
JANUS DEFINE REGION1 1001 OPENSERV 20 OPEN FILE -
      PRODPROC CMD 'I REGIONAL.QUERIES,1,HIGH'

      * Associate a remote server with REGION1
      *
JANUS DEFINEREMOTE REGION1 REGION1_SQL_SERVER -
      SPARC2 3001 TIMEOUT 120 -
      SITEUSER BART SITEACCT SIMPSONS

      * Make the port available for client access
      *
JANUS START REGION1
```

Janus applications that originate *inside* the Model 204 online (3270-based applications for example) require an OPENSERV port to be defined as MASTER, and a remote server associated with it. The other JANUS commands are similar to those above, so:

## Janus Client Functions

---

```
* Specify the domain. *
JANUS DOMAIN sirius-software.com

* Specify the location of the name server *
JANUS NAMESERVER 198.242.244.131

* Define master port for outgoing calls. *
* Disallow incoming use of the port. *
* Let the submitting user be the site handler. *
JANUS DEFINE REGION1 1001 OPENSERV 20 CMD '*' -
      OUTONLY MASTER

* Define a remote server to REGION1 *
JANUS DEFINEREMOTE REGION1 REGION1_SQL_SERVER -
      SPARC2 3001

* Make the port available for client access *
JANUS START REGION1
```

Once the ports and remote servers are defined, User Language programs using the Open Client functions can be run against them.

The Janus Open Client functions can send either language requests (usually SQL) or RPC's (Remote Procedure Calls) to a remote server. The remote server may reply by sending back rows of data, "alternate rows" (rows generated by SQL COMPUTE statements), return parameters, or nothing. The Janus Open Client functions include functions that let you define characteristics of the server connection, how you want to handle server messages and error conditions, and functions to manipulate the rows, alternate rows or parameters returned by the server.

A Janus Open Client application can establish one or more logical connections with one or more remote servers. Each logical connection is identified by a *connection identifier* that is returned by \$DB\_OPEN at the time it is established. Almost all Janus Open Client functions (except \$DB\_MSGHANDLE and \$DB\_ONCLOSE) operate on a specific connection and hence require the *connection identifier* as their first parameter. In

addition, any connection might be lost at any time (because of a JANUS FORCE, a BUMP of the sitehandler, a termination of the remote server, a network error, etc..) so any Janus Open Client function that operates on a connection might encounter a connection lost error. A connection lost error always results in the Janus Open Client \$function returning a -1. The ERRCLOSE option on the \$DB\_OPEN command eliminates the need to check for this condition with every function call, and the \$DB\_ONCLOSE function allows the programmer to specify a label where processing is directed when the connection to the remote server is unexpectedly lost.

A Janus Open Client User Language program is always in a particular state in relation to its remote server. That state is based on which \$DB\_ functions have been executed and on the response of the remote server. The connection to the remote server is half-duplex, with the User Language client opening the connection, building a request and sending it, and waiting for a response from the remote server before any more action can be directed at the connection. Here, briefly, is the order in which processing proceeds in a client program:

A program has no connection to a remote server until it performs a \$DB\_OPEN. The \$DB\_OPEN returns a *connection identifier*--usually held in a %variable--and places the connection into an uninitialized send state. The program then initializes the connection for *remote procedure calls* (RPCs) using \$DB\_RPCINIT, or initializes it for *language requests*, using \$DB\_LANGPUT.

If a connection is initialized for RPCs, \$DB\_RPCPARAM is used to set parameter names and values for the RPC. If the connection is initialized for language requests, subsequent calls to \$DB\_LANGPUT may be executed to further define the language request. When the request is completely specified it is sent to the remote server with a \$DB\_SEND, which places the program into an uninitialized receive state; that is, the client program can continue executing, but cannot refer to the remote server connection in any way until it executes a \$DB\_WAIT. (If \$DB\_WAIT is called without a previous call to \$DB\_SEND, the request is sent to the remote server by \$DB\_WAIT). \$DB\_WAIT initializes the receive state so that processing pauses, and waits for a response from the server.

The return code from `$DB_WAIT` determines how the receive state proceeds. Aside from error conditions, `$DB_WAIT` will indicate that the remote server sent a row, an alternate row or return data. *Rows* are the returned database information from SQL `SELECT` statements. *Alternate rows* are the return data from SQL `COMPUTE` statements. Alternate rows are sometimes called *compute rows*. *Return data* can be parameter names and values, or status codes for the RPC.

Most of the remaining `$DB_` functions are specific to handling one of the three types of information returned by a server. When receiving row or alternate rows, returned columns must be *bound* to User Language %variables. A program must determine the column structure of the return rows, and then bind the columns of interest to %variables using `$DB_BIND` or `$DB_ALTBIND` (for rows and alternate rows respectively). The %variables become populated with the values in the columns to which they are bound at the next invocation of `$DB_NEXTROW`.

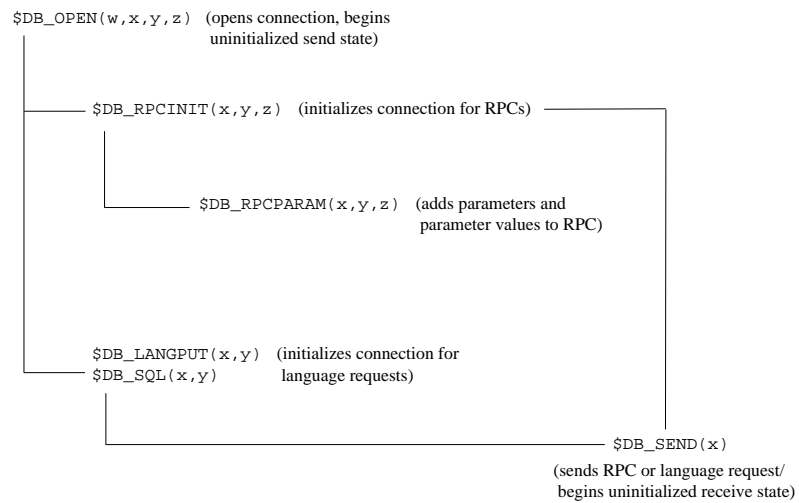
A few further functions handle processing cleanup. `$DB_CLOSE` closes a specific connection or all connections. `$DB_MSGHANDLE` and `$DB_ONCLOSE` provide a way of handling messages and broken connections. Client applications can be written so they open a connection and hold it open while any number of requests are defined and sent. They may also be written so the connection is closed after each client request, and reopened when another interaction is required with the server.

Sample programs in the appendices show how to use the functions in the context of particular applications. These and other sample programs are also in the Model 204 procedure file `JANUS`, installed with the Janus products.

All input parameters to `$DB_` functions are required parameters unless otherwise noted.

## Janus Open Client Function Library State Diagram

### Sending to remote server

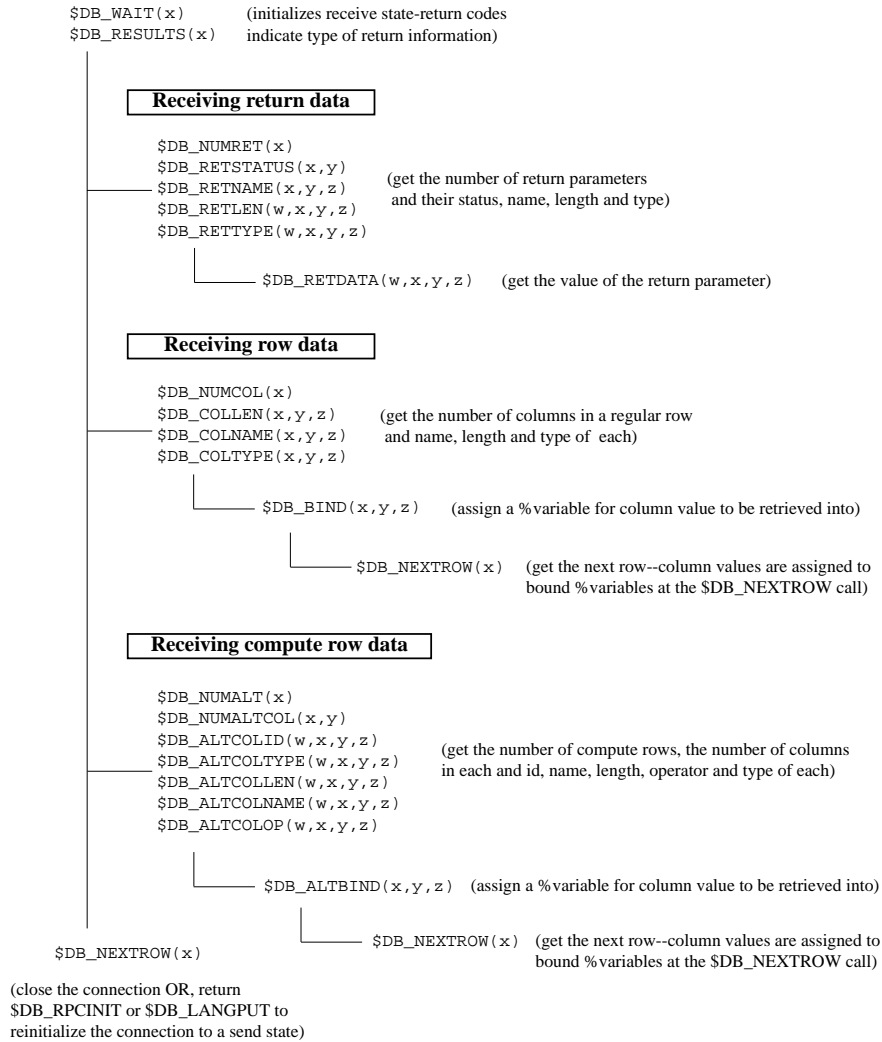


### miscellaneous functions (used regardless of state)

- `$DB_TEST(x)` (tests a connection)
- `$DB_ONCLOSE(x)` (sets a jump label for connection lost conditions)
- `$DB_MSGHANDLE(x)` (sets a subroutine as the message handler)

## Janus Open Client Function Library State Diagram (cont'd)

### Receiving from remote server



## Index

Alternate rows  
defined, 10

binding  
assigning column values to %variables

Compute rows  
defined, 10

Janus Client Functions, 5  
Janus commands  
JANUS DEFINE  
ERRCLOSE -- handling lost  
connections

Open Server User Language Coding, 2

remote server, 5  
return data  
defined, 10  
row data  
defined, 10

User Language coding considerations  
ERRCLOSE  
automatic handling of lost connections  
Open Client applications, 7