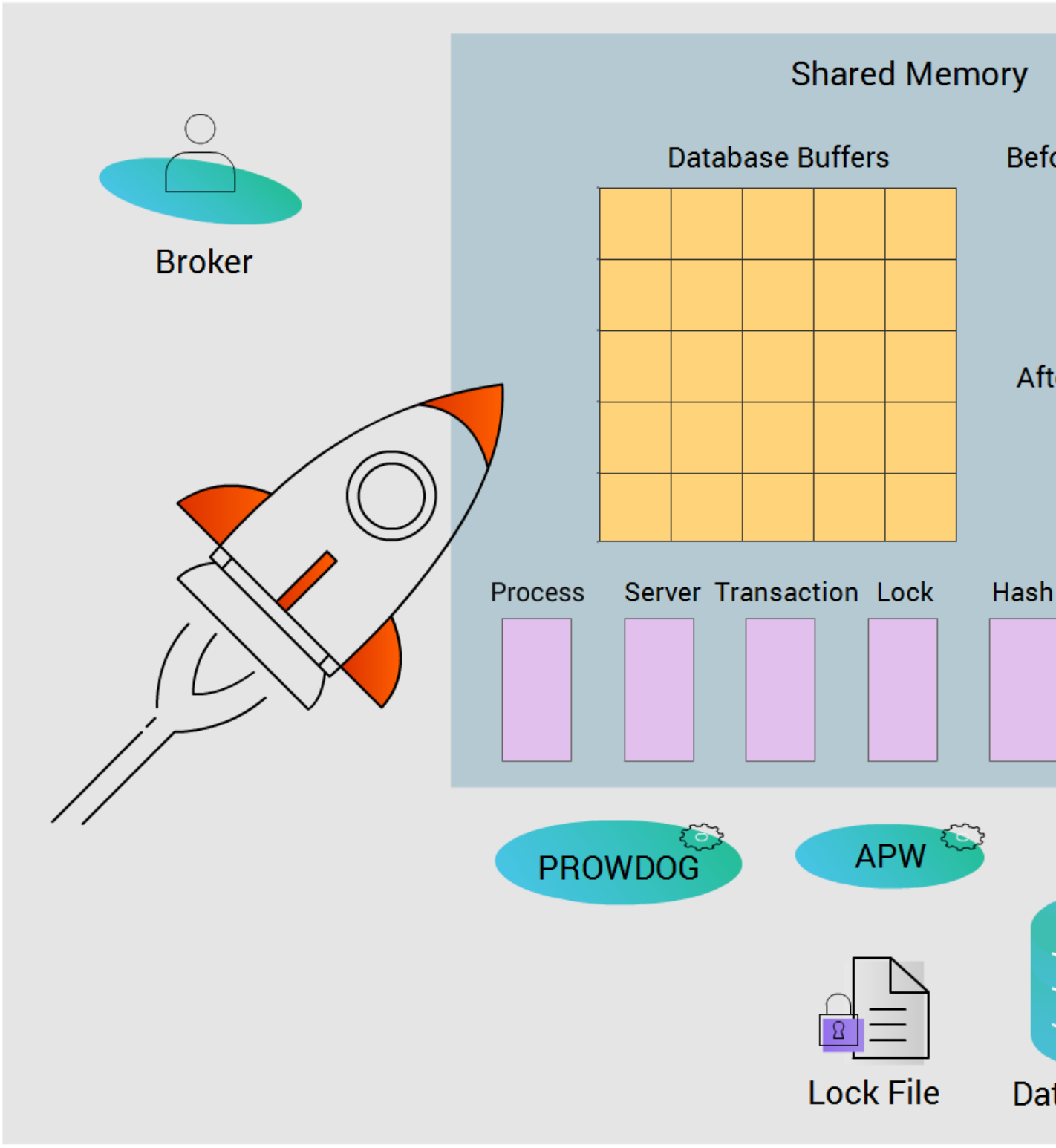
Progress®

Starting Up and Shutting Down a Database

OpenEdge®

1

Starting up a multi-user database in Proenv



Most OpenEdge databases operate in a multi-user environment in which multiple users access a database at the same time.

To start a database in a multi-user environment, you have to start one or more brokers that enable remote clients to connect to the database. After a broker is running, remote clients can access the database.

To start a multi-user database, you perform the following tasks:

- Start one or more brokers.
- Start background processes.

Before you learn how to perform these tasks, you will first learn about different types of brokers and the use of multiple brokers.

Note: In your application environment, you may also have to start other clients such as Progress Application Servers for OpenEdge (PAS for OpenEdge), NameServers, and WebSpeed Agents.

For details, see the following topics:

Types of brokers

A broker spawns servers, listens for connection requests from remote clients, and assigns remote clients to a server. There are two types of servers, ABL and SQL. ABL servers serve only ABL clients, and SQL servers serve only SQL clients.

In general, a broker can handles communication for both types of servers and clients. As a best practice, if you have multiple types of servers, you should have multiple brokers, each dedicated to one type of server.

There are two types of brokers:

- Primary broker—Instantiates and manages the shared memory of a database and performs the watchdog function. If you have multiple types of servers, you should direct the primary broker to serve OpenEdge ABL servers.
- Secondary broker—If you have multiple types of servers, you should direct a secondary broker to serve OpenEdge SQL servers.

Using multiple brokers

You can start multiple brokers for a database. Multiple brokers are used to:

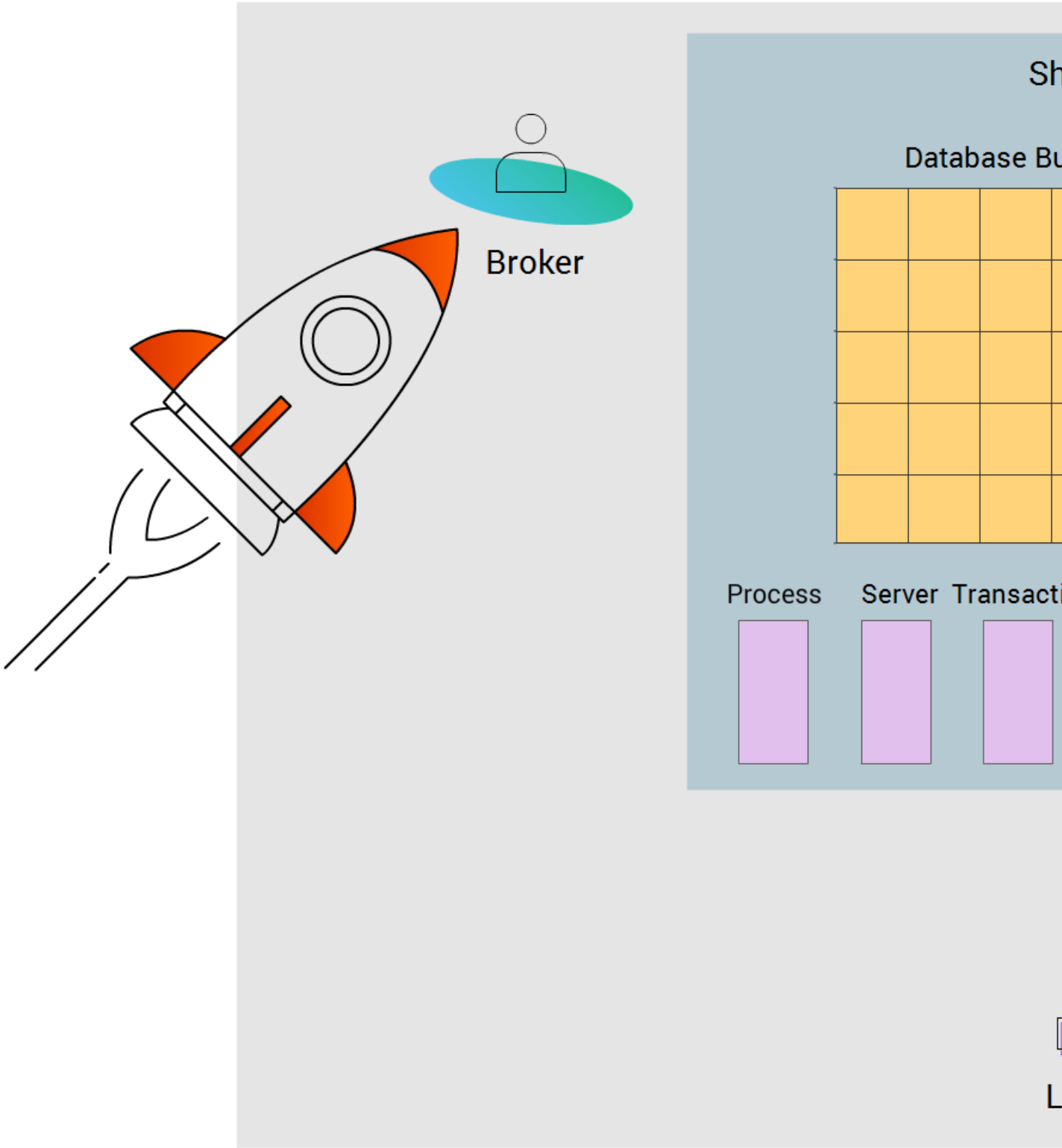
- Scale up the number of users.

- Support different types of users (ABL and SQL).
- Support multiple types of data usage such as online transactional processing, reporting, and business intelligence.
- Facilitate data flow across networks by invoking various startup parameters for clients, servers, and brokers.

With dedicated ABL and SQL brokers, you can:

- Avoid depletion of the connection pool due to simultaneous ABL and SQL client connections.
- Enforce both ABL and SQL security and ensure that only clients with appropriate privileges can access specific data.
- Control the number of servers (ABL or SQL) per broker.
- Control the number of clients (ABL or SQL) per server.

Starting a broker



To make a database available for multi-user access, you start at least one broker with appropriate startup parameters using the PROSERVE utility.

The command to start a broker is:

```
proserve db-name -H host-name -S service-name -n n -Mn n -Mpb n -Ma n -Mi n
-ServerType type
```

Where:

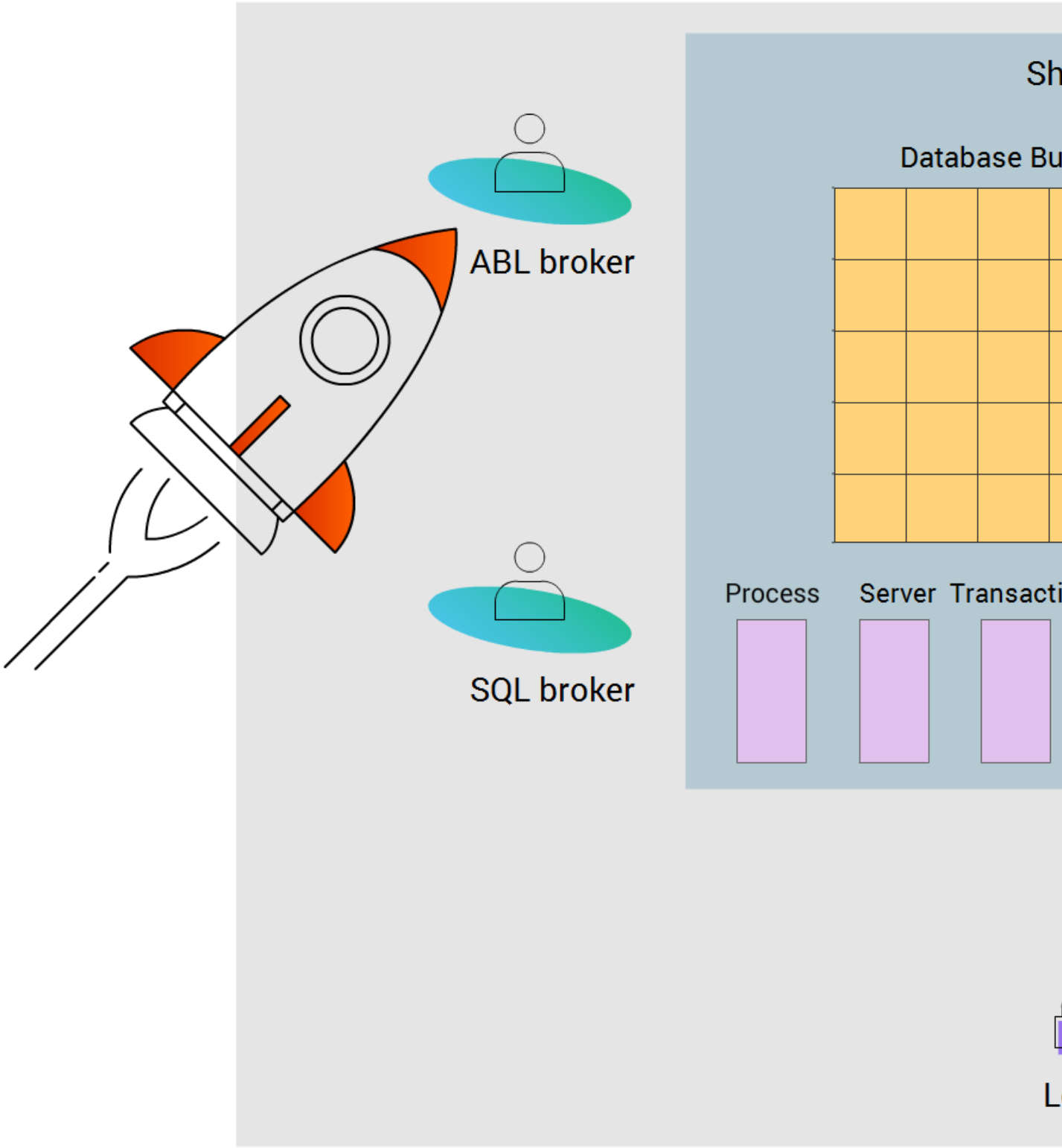
`-H host-name`	The name or IP address of the host computer that contains the database. It is used for remote client connections.
`-S service-name`	The service name or port number to be used by a broker. This parameter is required only for remote clients and SQL clients.Valid port range is from 2000 to 32767.

`-Mn n`	The maximum number of servers that can be started by the broker. It is only required for the primary broker.Default value: 5 To calculate the maximum number of servers in the database, use this formula: `-Mn = (Sum of all -Mpb values) + 1 per secondary broker`
`-Mpb n`	The maximum number of servers per broker.
`-Ma n`	The maximum number of remote clients per server.Default value: 4Recommended value: 10
`-Mi n`	The minimum number of remote clients per server before the broker spawns another server.
`-ServerType type`	The type of server. Valid values are 4GL, SQL, and BOTH.Default value: BOTH Note: 4GL is synonymous with ABL. Use 4GL when you want to start an ABL broker.

Note:

- All startup parameters are case-sensitive.
- When you start a broker without specifying the –ServerType parameter, it creates a default broker that caters to both ABL and SQL clients.
- Even if a client resides on the same computer as the database, the client is treated as a remote client if the hostname, service name, or network type parameters are specified while connecting to a database.
- SQL clients are treated as remote clients even if they reside on the same computer as the database.
- You can also start a broker using the _MPROSRV command in either Windows or UNIX. It has the same syntax as PROSERVE.

Starting multiple brokers



To start multiple brokers for a database, you first start the primary broker with the startup parameters discussed previously.

Then, to start each additional secondary broker, you use a separate PROSERVE with the -m3 parameter.

The command to start each additional broker is:

```
proserve db-name -H host-name -S service-name -Mpb n -Ma n -Mi n -ServerType
type -m3
```

Where:

-H host-name	The name or IP address of the host computer that contains the database. It is used for remote client connections.
-S service-name	The service name or port number to be used by a secondary broker.
-Mpb n	The maximum number of servers on a secondary broker.
-Ma n	The maximum number of remote clients per server on a secondary broker.
-Mi n	The minimum number of remote clients per server on a secondary broker.
-ServerType type	The type of server served by a secondary broker.
-m3	A secondary login broker.

Example: Starting multiple brokers

Suppose the appdb database must meet the following requirements:

- Runs on the local computer (localhost).
- Is accessed by both ABL and SQL users.
- Uses port 3001 for the ABL broker and port 3101 for the SQL broker.
- Must support 40 ABL users, 20 SQL users, and 40 other processes.
- Uses at most 10 servers per ABL broker and 4 servers per SQL broker.
- Uses at least 2 remote clients per ABL or SQL server.

To support these requirements, you would need to start two brokers: the primary broker to support ABL clients and the secondary broker support SQL clients. To start each broker, you would use a separate PROSERVE command.

To start the primary ABL broker, you would use the following PROSERVE:

```
proserve appdb -H localhost -S 3001 -n 101 -Mn 15 -Mpb 10 -Ma 4 -Mi 2
-ServerType 4GL
```

The following table lists each parameter with its description:

Parameter	Description
`-H localhost`	Specifies that the host on which the database resides is localhost.
`-S 3001`	Specifies that the port on which the primary broker has to run is 3001.
`-n 101`	Specifies that the maximum number of users for the database is 101. 40 (ABL) + 20 (SQL) + 40 (other processes) + 1 (secondary broker) = 101
`-Mn 15`	Specifies the maximum number of servers is 15. 10 (ABL) + 4 (SQL) + 1 (secondary broker) = 15
`-Mpb 10`	Specifies that the maximum number of ABL servers per broker is 10.
`-Ma 4`	Specifies that the maximum number of remote ABL clients per server is 4.
`-Mi 2`	Specifies that the minimum number of remote ABL clients per server before the primary broker starts another server is 2.
`-ServerType 4GL`	Specifies that the type of server served by the primary broker is ABL.

To start the secondary SQL broker, you would use the following PROSERVE:

```
proserve appdb -H localhost -S 3101 -Mpb 4 -Ma 5 -Mi 2 -ServerType SQL -m3
```

The following table lists each parameter with its description:

Parameter	Description
`-H localhost`	Specifies that the host on which the database resides is localhost.
`-S 3101`	Specifies that the port on which the secondary SQL broker has to run is 3101.
`-Mpb 4`	Specifies that the maximum number of SQL servers on the secondary broker is 4.
`-Ma 5`	Specifies that the maximum number of remote SQL clients per server on the secondary broker is 5.

Parameter	Description
`-Mi 2`	Specifies that the minimum number of remote SQL clients per server on the secondary broker is 2.
`-ServerType SQL`	Specifies that the type of server served by the secondary broker is SQL.
`-m3`	Starts the secondary broker.

Starting background processes

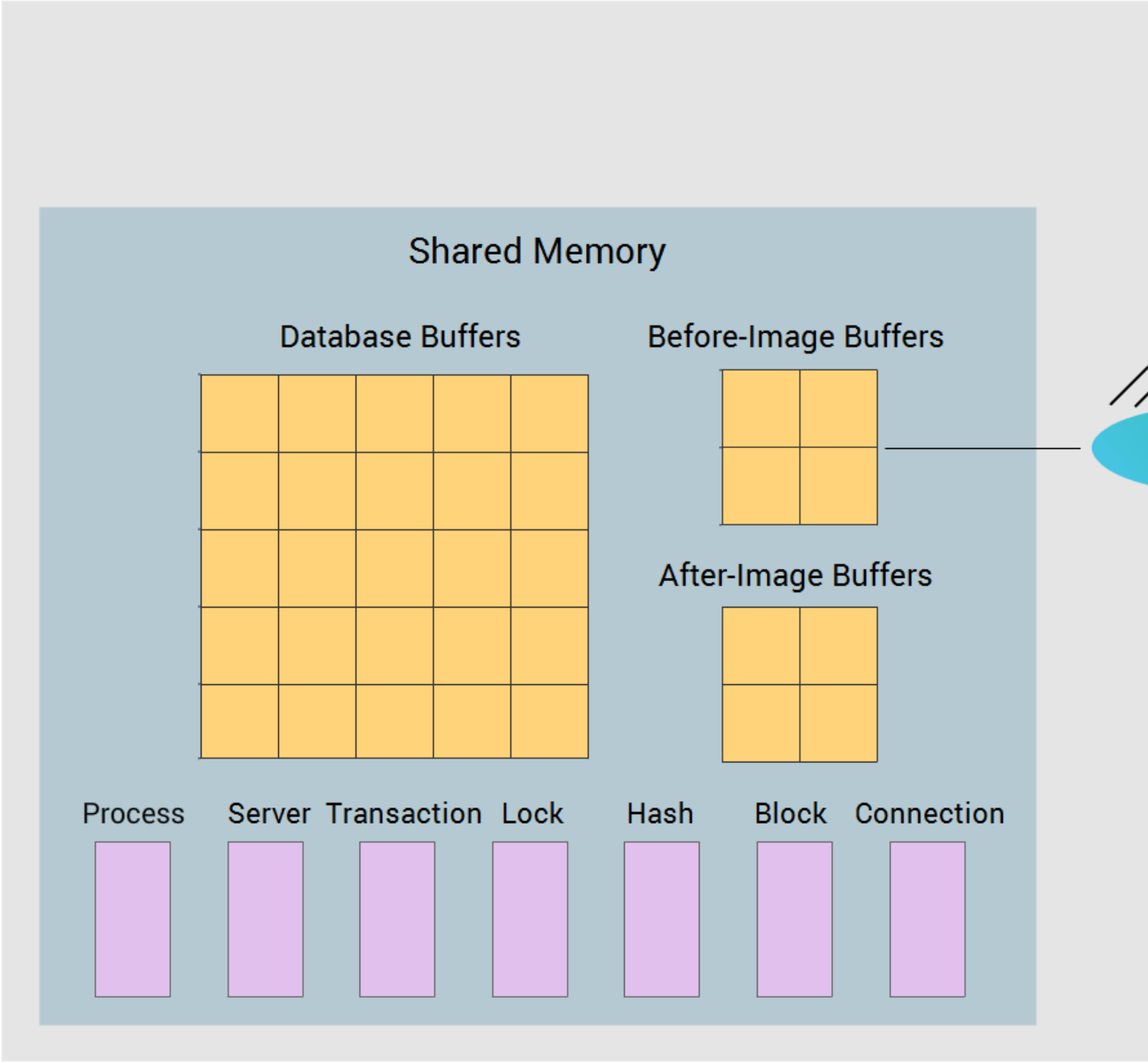
Background processes continually performing overhead functions in the background. There are five types of background processes—BIW, AIW, APWs, and watchdog.

Although these background processes improve performance by offloading work from OpenEdge RDBMS, they consume resources associated with database processes. Therefore, you should increase the value of the Number of Users (`-n`) parameter to accommodate these processes.

Next, you learn how to start these background processes.

Note: All background processes except the watchdog are available only in the OpenEdge Enterprise RDBMS license.

Starting the BIW



The Before-Image Writer (BIW) is an optional background process that continually writes modified BI buffers to disk. The BIW enhances database performance because it ensures a steady supply of empty BI buffers.

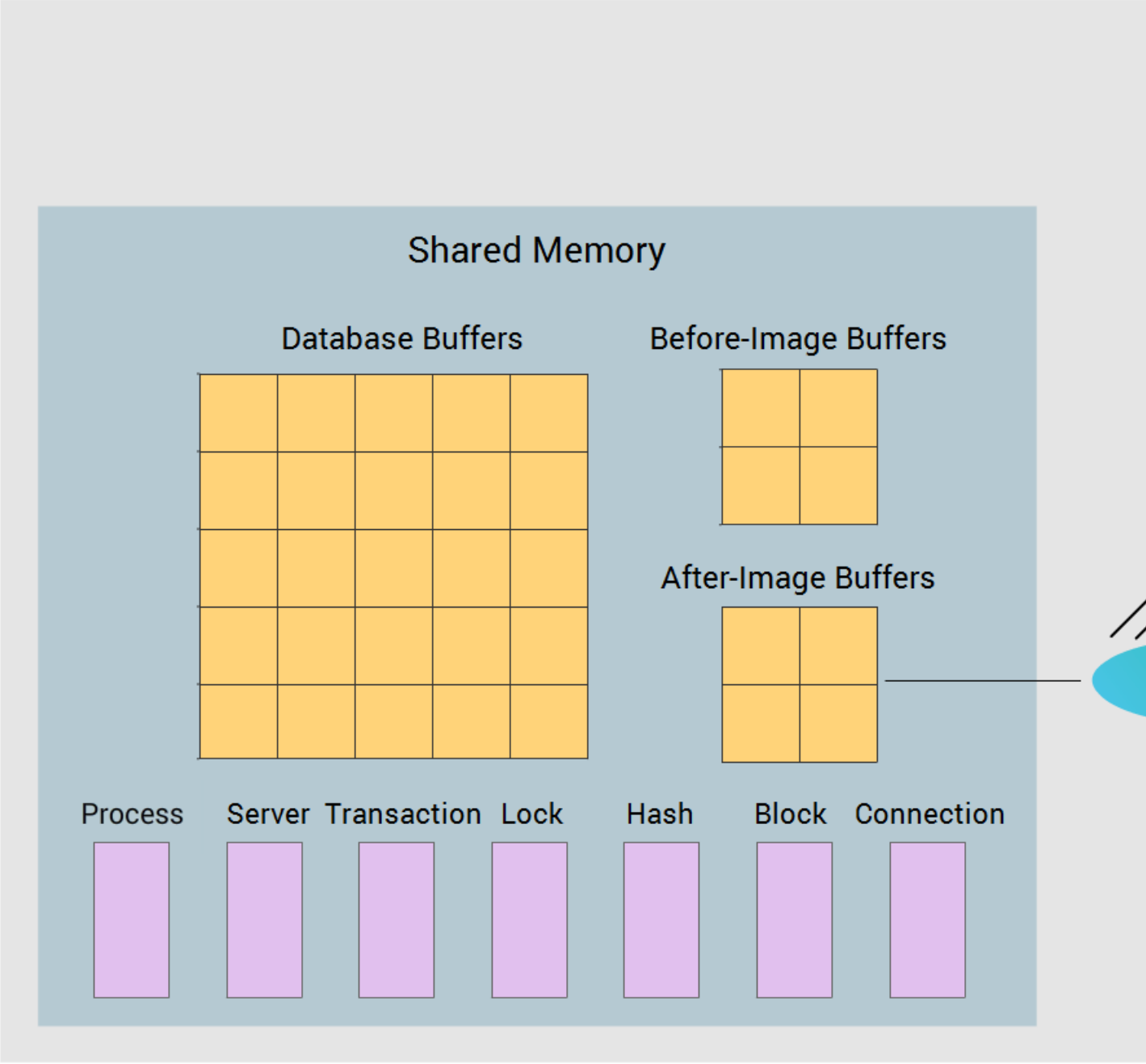
As a best practice, after you start up a database, you should start the BIW.

The command to start the BIW for a running database is:

```
probiw db-name
```

Note: The BIW is available only with the OpenEdge Enterprise RDBMS license. You can run only one BIW per database.

Starting the AIW



The After-Image Writer (AIW) is an optional background process that continually writes filled AI buffers to disk. The AIW enhances database performance because it ensures a steady supply of empty AI buffers.

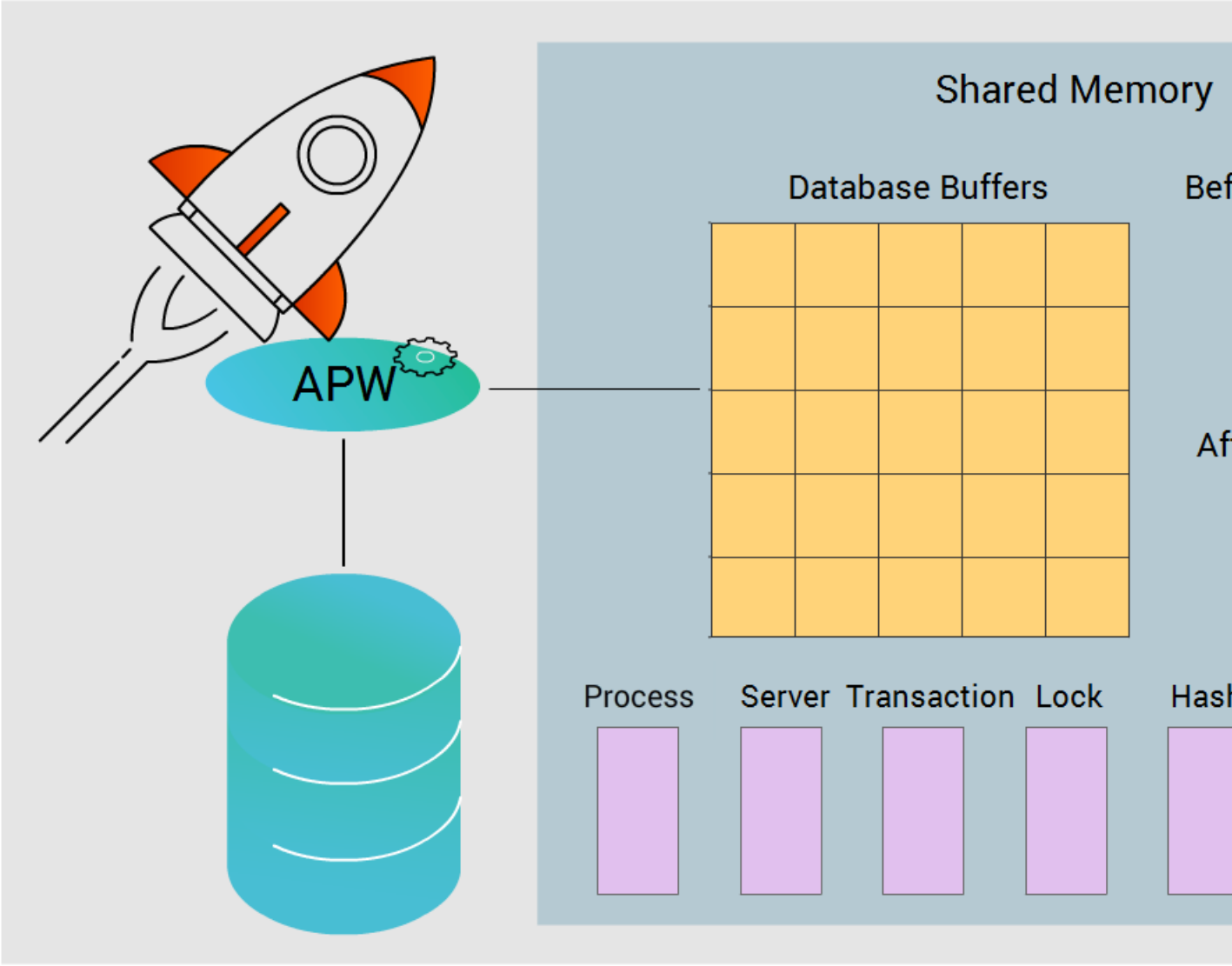
As a best practice, after you start up an AI-enabled database, you should start the AIW.

The command to start the AIW for a running database is:

```
proaiw db-name
```

Note: The AIW is available only with the OpenEdge Enterprise RDBMS license. You can run only one AIW per database.

Starting an APW



An asynchronous page writer (APW) is an optional background process that continually writes unlocked, modified database buffers to disk. An APW enhances database performance because it:

- Ensures that a steady supply of empty database buffers is available so OpenEdge RDBMS can use them when needed.

- Reduces the number of database buffers that OpenEdge RDBMS must examine before writing an unlocked, modified database buffer to disk.
- Reduces the overhead associated with checkpointing because fewer modified database buffers have to be written to disk at every checkpoint.

You can start one or more APWs. As a best practice, after you start up a database, you should start at least one APW.

The command to start an APW for a running database is:

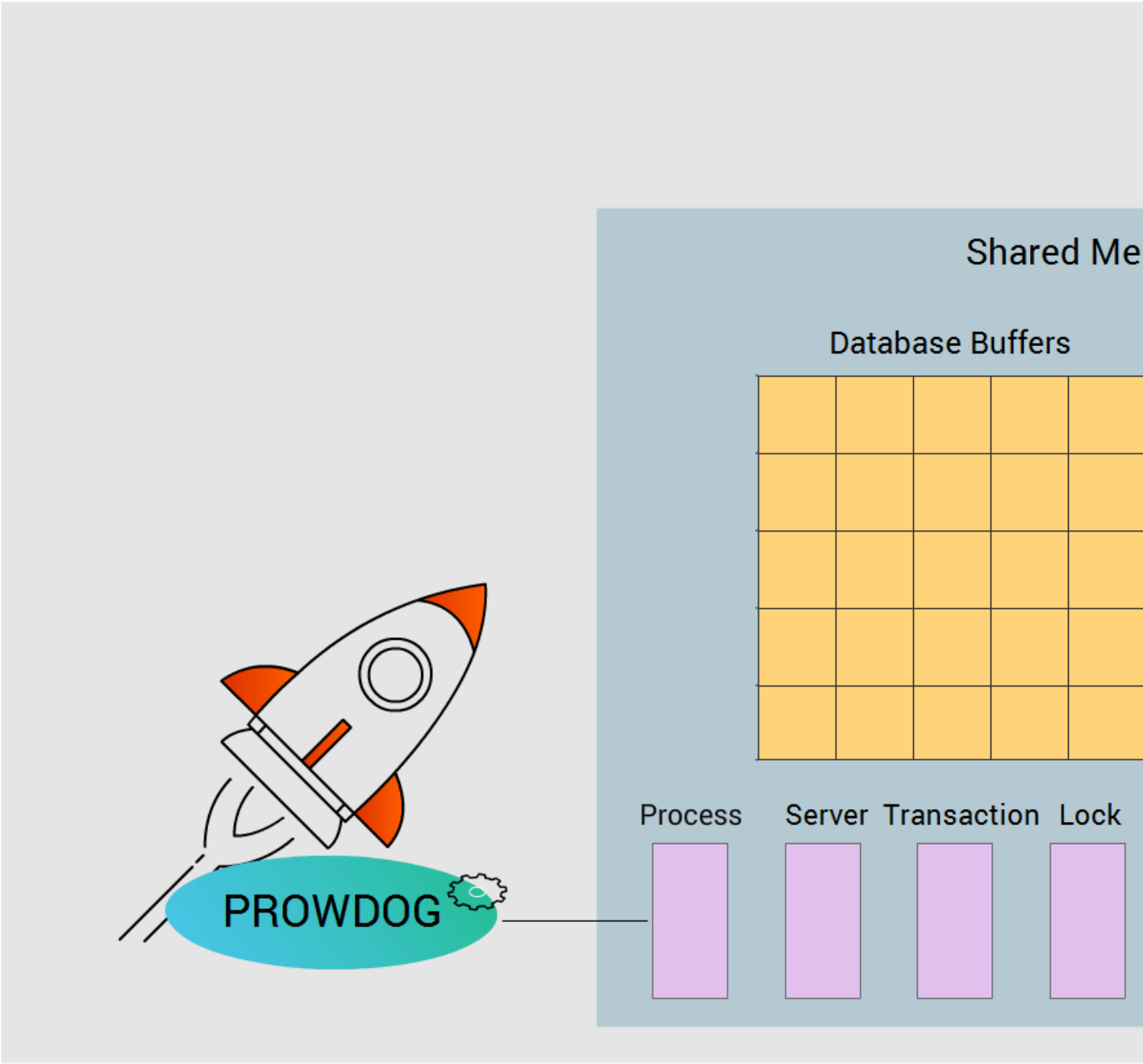
```
proapw db-name
```

Repeat this command for each APW instance you want to start.

Note:

- The APW is available only with the OpenEdge Enterprise RDBMS license.
- Applications that perform fewer updates to a database require fewer APWs.
- Applications that are read-only do not require any APW.

Starting the watchdog



The watchdog (PROWDOG) is an optional background process that:

- Checks for the existence of the lock (.lk) file and makes sure that the .lk file is valid. If the file disappears or is corrupted, the database is automatically shutdown.
- Cleans up after improperly terminated self-service clients as well as the remote clients of lost servers.
- Checks if the broker is still alive. If for some reason the broker is not alive, the database is automatically shut down.

Without a watchdog process, the primary broker performs the watchdog function in addition to all its other duties. So, having a dedicated watchdog improves database performance. As a best practice, after you start up a database, you should start the watchdog.

The command to start the watchdog for a running database is:

```
prowdog db-name
```

Note: You can run only one watchdog per database.

Using a parameter file

As the number of startup parameters for a database increases, starting a broker could become tedious. A parameter file makes it easier to start a broker. You can enter all the parameters into a parameter file, and then use the parameter file to start the broker with PROSERVE.

A parameter file is a text file in which each line represents a parameter. Every line has a name and, optionally, a value and a comment. Some parameters do not have values; they only have flags. A parameter file can include any number of startup parameters.

You can also have multiple parameter files, each dedicated to a specific purpose. For example, you can specify application-specific parameters in one parameter file, database-specific parameters in another parameter file, and user-specific parameters in yet another file. You can also reference one parameter file from within another.

There are two ways to create a parameter file:

- Create and populate a file with parameters using a text editor. You must use this method if you use UNIX.
- Populate a parameter file template with values using the Data Administration tool in Windows.

Next, you will learn how to create a parameter file and then how to invoke it.

Creating a parameter file using a text editor

Follow these guidelines to create a parameter file using a text editor:

- Specify each parameter and its associated value on a single line.
- Ensure that every parameter has a name and a value if the value is required.
- For parameter values that include spaces, enclose the value in single or double quotes.
- Use the pound sign (#) before each comment.
- If you want to reference another parameter file, use the –pf parameter and the parameter filename.
- Save the parameter file with a .pf extension.

When you use a parameter file from the command line, OpenEdge combines any parameters you specify on the command line with the parameters in the .pf file.

Note:

- The parameters not listed in the .pf file assume their default values.
- In case of duplicate parameters from the command line or the .pf file, the last duplicate parameter value takes precedence.

Here is an example of a parameter file, **appdb.pf**:

```
# Parameter file for the appdb database
# appdb.pf
# Startup parameters for brokers, servers, and users
#
-db appdb # database name
-H localhost # host machine name
-S 5001 # port number
-n 100 # maximum number of users
-Mn 12 # maximum number of servers and brokers
-Mpb 5 # maximum number of servers per broker
-Ma 5 # maximum number of remote clients per server
-Mi 2 # minimum number of remote clients per server
-ServerType 4GL # type of server
```

Procedure: Creating a parameter file in Windows

If you use Windows, you can create a parameter file using the Data Administration tool.

Follow these steps to create a parameter file in Windows:

Step	Action
1	Select **Start > All Programs > Progress > OpenEdge 11.6 > Data Administration**.
2	Select **Utilities > Editor for Parameter Files**.
3	Enter a fully qualified filename with the .pf extension in the **Parameter File** field, and then click **OK**.
4	Select each parameter you want to use and then enter the required value.
5	Click **OK** to save the file.

Invoking a parameter file

After you create a parameter file, you can invoke it to start a broker by using PROSERVE and the parameter file.

The command to invoke a parameter file to start a broker is:

```
proserve –pf parameter-file.pf
```

One reason you invoke a parameter file from another parameter file is when you want to allow individual users to use additional parameters at the user level.

Assume that the **primary.pf** parameter file sets the global parameters for the appdb database. Individual users can create and maintain a private myparam.pf file, which enables them to set additional parameters for their own requirements.

To call **myparam.pf** from **primary.pf**, enter a `-pf` parameter and the parameter filename in the primary.pf file, as shown:

primary.pf

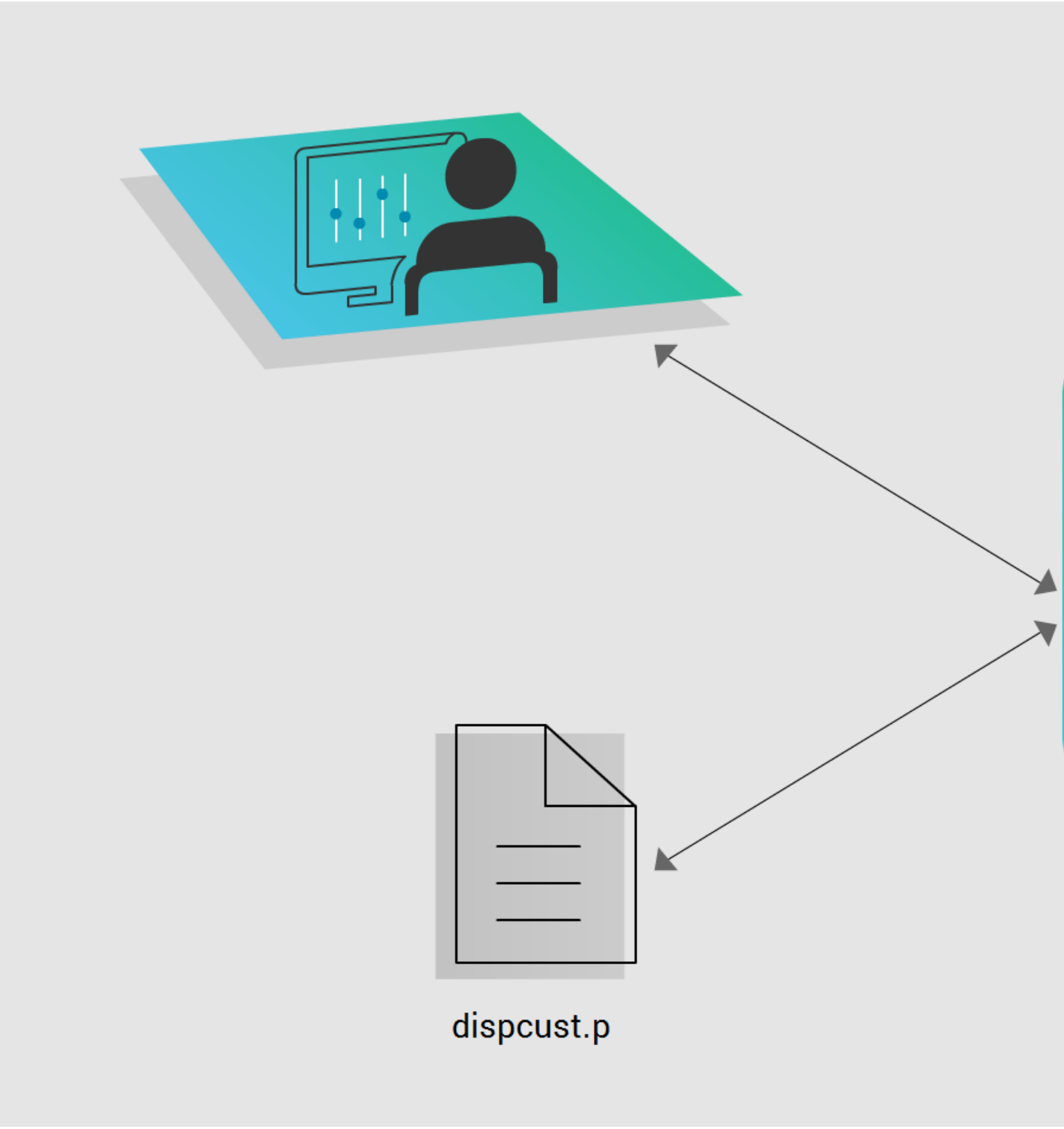
```
#-------------------------------------------------------------------
# This file sets the global parameters for the appdb database.
#-------------------------------------------------------------------
-dbappdb# database
. . .
-pfmyparam.pf# personal parameter file
```

myparam.pf

```
-T /usr/tmp# location for user-specific temp file
-mmax1024# maximum memory allocated for r-code
```

2

Starting multi-user database sessions in Proenv



After you start one or more brokers for a database, the database is ready for simultaneous access by multiple users.

In a multi-user environment, a client can interact with a database by establishing one of these two sessions:

- Multi-user interactive session—which enables a client to interact with the database.
- Multi-user batch session—which processes a batch of commands (also known as a batch job) at a scheduled time. It does not involve any user interaction.

If you are using UNIX, before you can start a multi-user session, you must first configure UNIX environment variables. You will learn how to do this next.

For details, see the following topics:

- Configuring UNIX environment variables
- Multi-user interactive session
- Multi-user batch session
- Procedure: Disconnecting user sessions

Configuring UNIX environment variables

If you are using UNIX, before you can start a multi-user session, you must first configure UNIX environment variables in the `.profile` file. You edit the .profile file using a text editor.

There are four mandatory environment variables and one optional variable.

The mandatory environment variables are:

Mandatory variable	Description
DLC	The pathname of the directory in which OpenEdge RDBMS is installed.
PATH	The list of directories the UNIX kernel searches for executable files. Use colons (:) to separate directory names.
TERM	The type of terminal. This must be set to run OpenEdge character-mode (TTY) client sessions (including batch mode). When you specify a terminal type, OpenEdge searches for that type in the PROTERMCAP file (see below) and uses the specified definitions.
PROPATH	The list of directories that contain OpenEdge ABL application code. At runtime, OpenEdge searches for application code in PROPATH in the order listed.

The optional variable is:

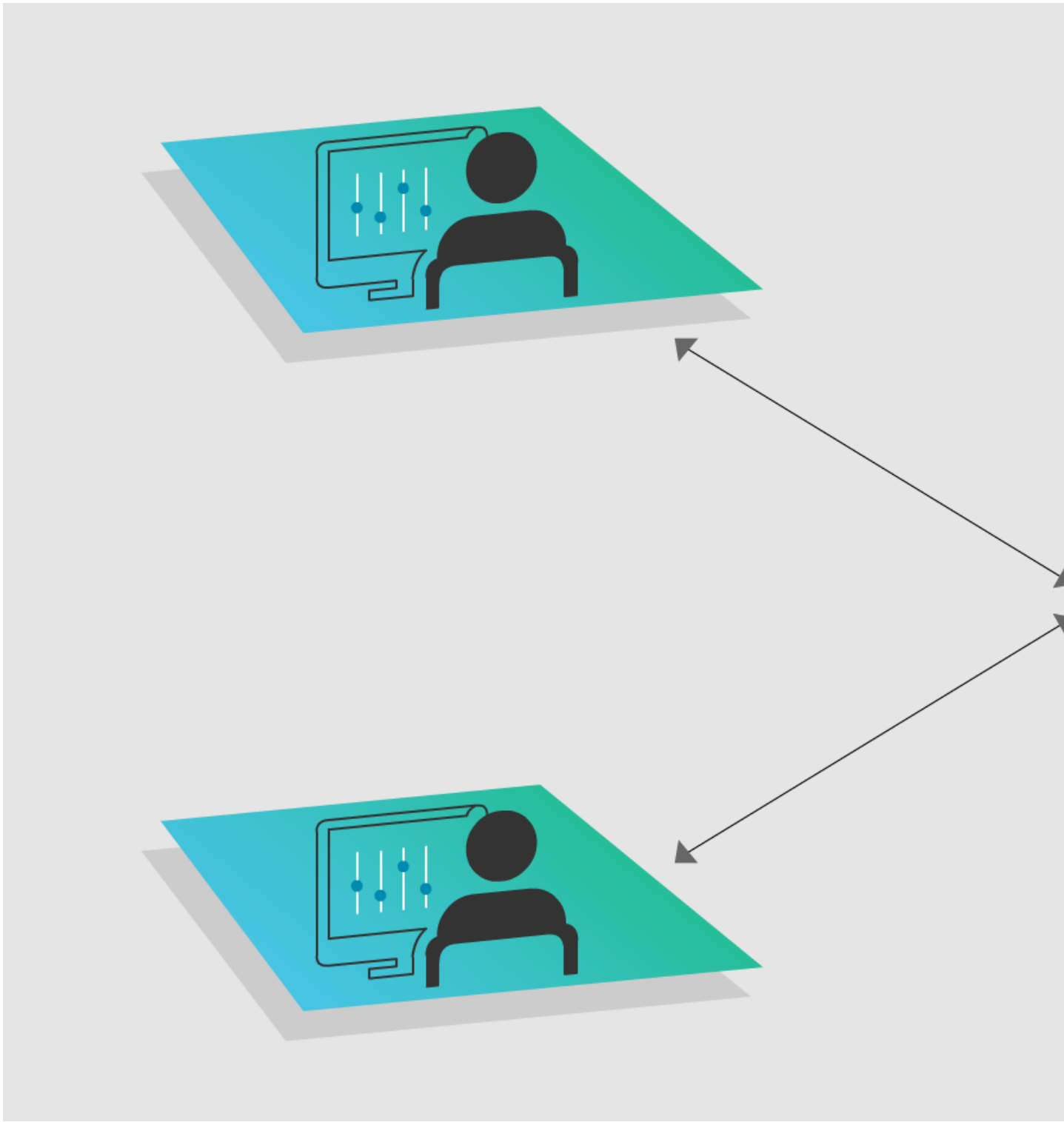
Optional variable	Description
PROTERMCAP	The full pathname of the terminal definition file that you want your OpenEdge session to use. The default terminal definition file is called PROTERMCAP and is installed in the OpenEdge installation directory. You only have to set PROTERMCAP if you want to use a different terminal definition file.

Note: You can also set the environment variables in the .login script in UNIX or in the script used to start OpenEdge.

Here is an example of a .profile file:

```
. . .
DLC=/usr/dlc; export DLC
PATH=$DLC:$DLC/bin:$PATH; export PATH
TERM=vt220; export TERM
PROPATH=.:$DLC:/usr/apps; export PROPATH
#
PROTERMCAP=$DLC/protermcap; export PROTERMCAP
. . .
```

Multi-user interactive session



A multi-user interactive session is one in which a client interacts with a database that is being accessed simultaneously by multiple users.

As a database administrator (DBA), you can use a multi-user interactive session to perform database administration tasks, such as:

- Online backup of a database
- Online dumping and loading of data
- Online adding files to the database
- Online enablement of after-imaging
- Online enablement of automatic AI file management
- Online index activation for a specific index
- Online index activation or index rebuild for a specific tenant or tenant group, if you have a multi-tenant database
- Online moving of a table or index for a specific tenant or tenant group
- Online index activation or index rebuild for a specific table partition, if you have a table-partitioned database
- Online moving of a table or index for a specific table partition

Starting a multi-user interactive session

You can start a multi-user interactive session against a running database using:

- The PROWIN command in Windows
- The MPRO command in UNIX

Note: When you start a multi-user interactive session using PROWIN or MPRO, the Procedure Editor is launched.

In Windows, the syntax of the PROWIN command is:

```
prowin db-name [-H host-name -S service-name]
```

In UNIX, the syntax of the MPRO command is:

```
mpro db-name [-H host-name -S service-name]
```

Note: You can also start a multi-user interactive session against a database in character mode using the _PROGRES command in either Windows or UNIX. It has the same syntax as PROWIN.

Assume that the appdb database is running.

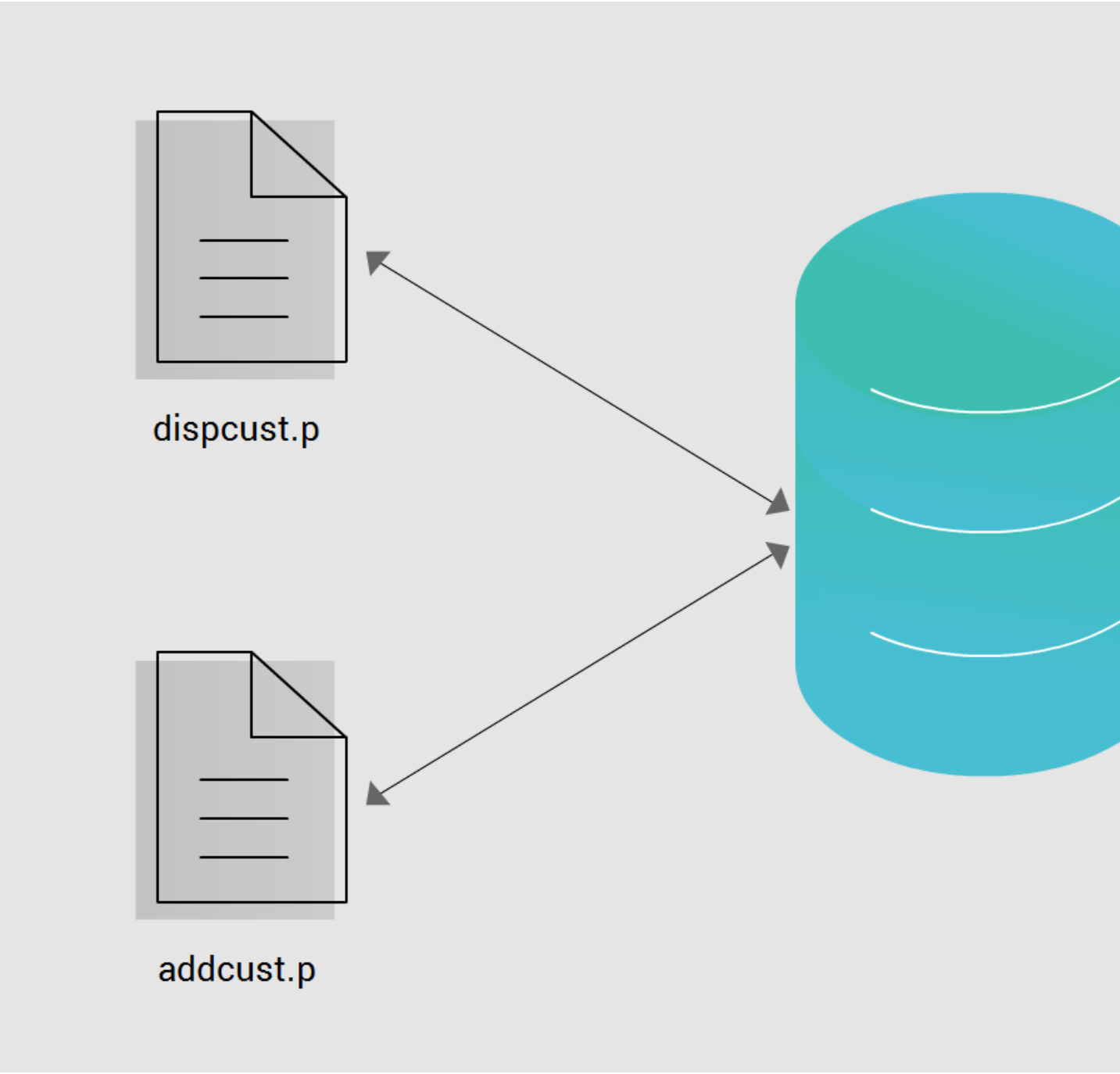
In Windows, to start a multi-user interactive session against appdb, enter:

```
prowin appdb
```

In UNIX, to start a multi-user interactive session against appdb, enter:

```
mpro appdb
```

Multi-user batch session



A multi-user batch session runs batch jobs on a database that is being accessed simultaneously by multiple users.

As a DBA, you start a multi-user batch session when you have to:

- Perform large-scale database updates.
- Create daily, weekly, or monthly reports.

- Generate database statistics.

Starting a multi-user batch session

You can start a multi-user batch session against a running database using:

- The PROWIN[1] command in Windows
- The MBPRO command in UNIX

In Windows, the syntax of the PROWIN command is:

```
prowin db-name -b -p procedure-name [> output-file]
```

In UNIX, the syntax of the MPRO command is:

```
mbpro db-name -p procedure-name [> output-file]
```

Note: You can also start a multi-user batch session against a database in character mode using the _PROGRES command in either Windows or UNIX. It has the same syntax as PROWIN.

Assume that the appdb database is running and there is an OpenEdge procedure file, dispcust.p, which displays customer information.

In Windows, you can run **dispcust.p** in a multi-user batch session against the appdb database and output the results to a text file, **dispcust.txt**:

```
prowin appdb –b –p dispcust.p > dispcust.txt
```

In UNIX, you can perform the same task using this command:

```
mbpro appdb –p dispcust.p > dispcust.txt
```

Procedure: Disconnecting user sessions

While your database is running, sometimes it may be necessary to disconnect an individual session that is currently connected to the database. Note that user sessions include background processes, PROMONs, and online backups.

For example, a session might be locking records that are required by other users, but the person who started the session has left for the day. Disconnecting a session terminates any active transactions that the session is incurring as well as releasing any record locks.

Another scenario is that you might have launched multiple APWs at database startup and over time found out that fewer APWs were actually required. In such a case, you could disconnect one or more APW instances to improve performance.

[1] Use PROWIN on a 64-bit computer and PROWIN32 on a 32-bit computer.

To disconnect a session, you first identify the user ID associated with the session, and then terminate the session using the user ID.

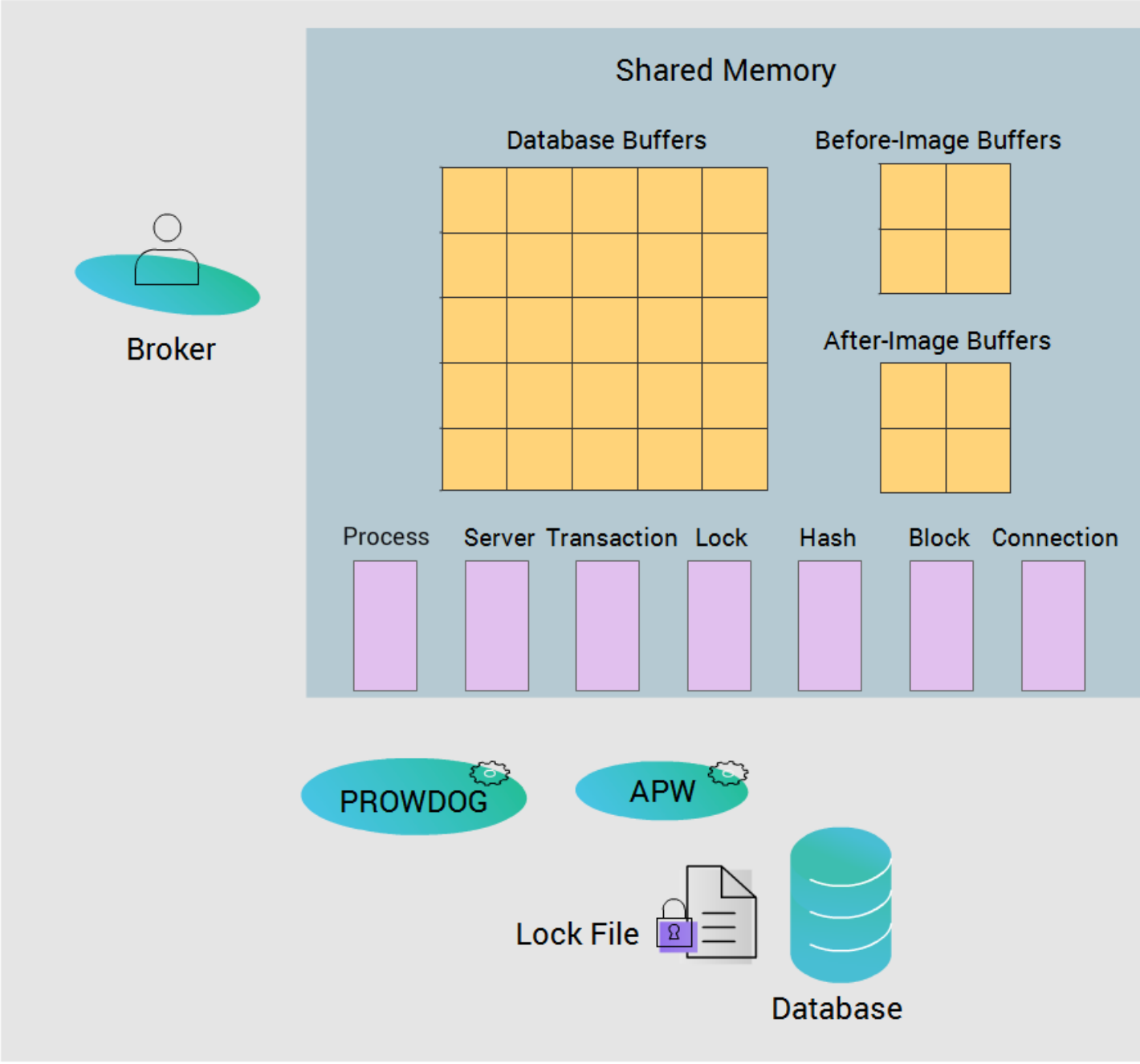
Follow these steps to disconnect a session using PROMON:

Steps	Action
1	Invoke PROMON on your running database: `promon db-name`
2	At the “Enter your selection:” prompt, enter 1 (for User Control).
3	At the next “Enter your selection:” prompt, enter 1 (for Display all entries). Information about different database sessions such as user ID, user name, type, and login time, is displayed. Make a note of the user ID of the session you want to disconnect.
4	Enter q to return to the main PROMON prompt.
5	At the “Enter your selection:” prompt, enter 8 (for Shut Down Database).
6	At the “Enter choice>” prompt, enter 1 (for Disconnect a User).
7	At the “Enter the user number which is to be disconnected:” prompt, enter the user ID found in Step 3. The session is terminated.
8	At the “Enter choice>” prompt, enter x to return to the main PROMON prompt.
9	At the “Enter your selection:” prompt, enter q to exit PROMON.

Note: In UNIX, do not disconnect a user session with the command kill –9 because it could shut down and possibly damage the database.

3

Shutting down a multi-user database in Proenv



There are a number of database maintenance tasks that require that a multi-user database be shut down. In addition, if your environment allows you to have regular downtime for database maintenance, you can shut down a multi-user database periodically. In either case, you should choose a time when there is little or no database activity and inform your users ahead of time. You can then start a single-user session to perform your maintenance tasks.

There are two kinds of shutdown you can perform:

- Unconditional—This is the most common method. The database is shutdown with no warning or grace period. Any open transactions are aborted.
- Emergency—This is used if an unconditional shutdown gets hung up or freezes. It is harsher than an unconditional shutdown because it actually crashes the database. If there were any active transactions, crash recovery is performed when the database is restarted in multi-user or single-user mode. You should perform an emergency shutdown only as a last resort.

You can shut down a multi-user database using either the PROSHUT or PROMON utility.

For details, see the following topics:

- Shutting down a database using PROSHUT

Shutting down a database using PROSHUT

The command to shut down a database using PROSHUT is:

```
proshut db-name [-bn|-shutdownTimeout [immed|maximum|n]| -by|-F]
```

Where:

`-bn`	Shuts down the database only when there are no active users.
`-shutdownTimeout`	Allows the database a specified amount of time to halt activity before initiating an immediate shutdown. Normal shutdown proceeds until all activity ceases or the specified time elapses. At the end of the shutdown timeout, all database output stops, active users are disconnected, and the database is shut down immediately. If this parameter is not specified, the default interval is 10 minutes. The minimum shutdown timeout (immed) value is 60 seconds and the maximum shutdown timeout (maximum) value is 24 hours. This parameter is ignored when forced shutdown (-F) is used.

-by	Shuts down the database unconditionally, disconnects all the users, and rolls back any active transactions.
-F	Forces an emergency shutdown. OpenEdge RDBMS kills the broker and all background processes, disconnects all clients, removes shared memory, marks the database as crashed, and removes the lock file. When the database is restarted, OpenEdge RDBMS performs crash recovery and rolls back any incomplete transactions left behind by the emergency shutdown. Note: If both -F and -shutdownTimeout are used, then -shutdownTimeout is ignored.

Assume you want to shut down the appdb database immediately. Enter:

```
proshut appdb -by
```

4

Starting up and shutting down a single-user database in Proenv

To perform database maintenance tasks in a single-user environment, you need to access the database in single-user mode.

A single-user client can start a database by establishing one of these two sessions:

- Single-user interactive session—which enables a client to interact with the database.
- Single-user batch session—which processes a batch of commands at a scheduled time. This does not involve any user interaction.

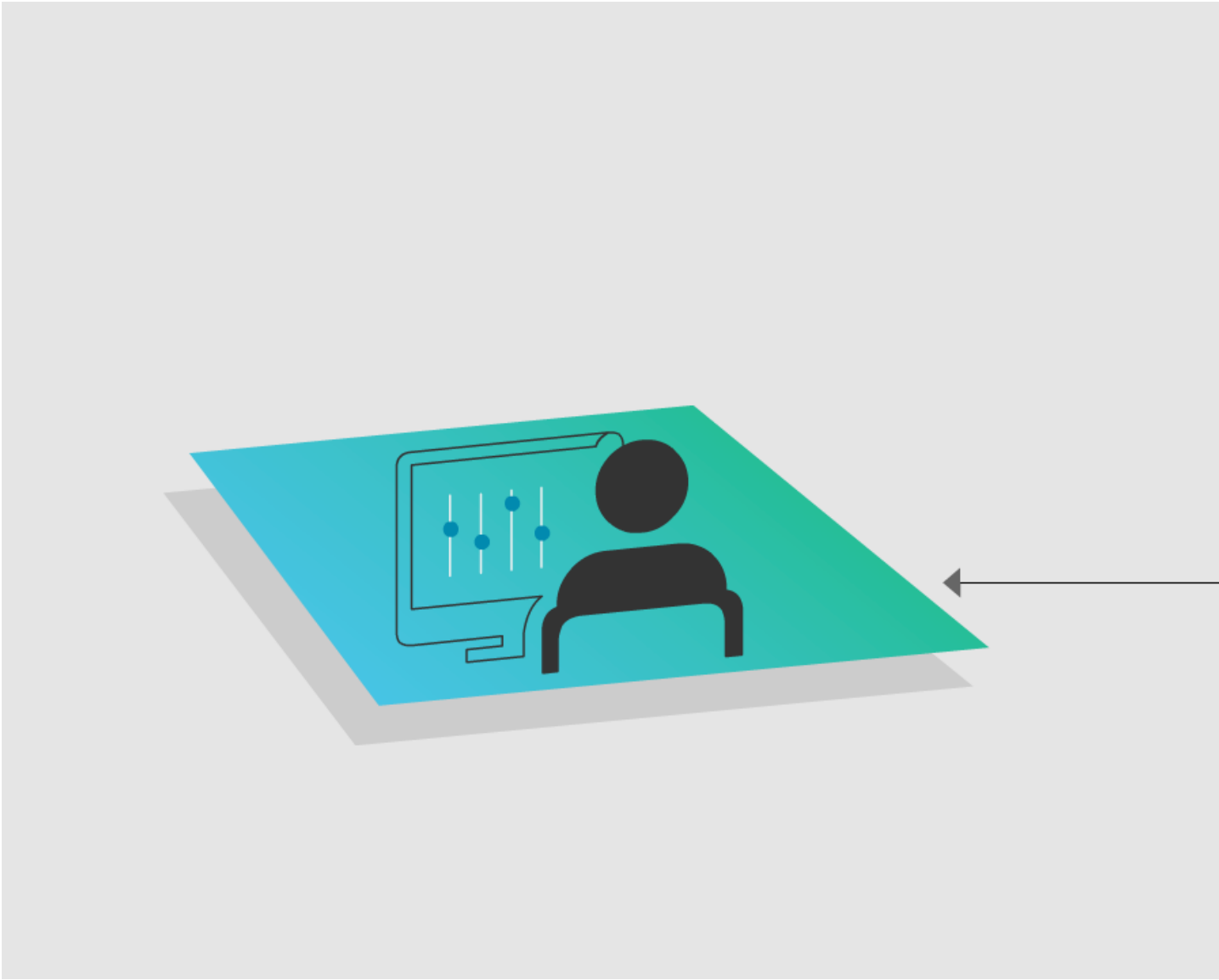
In a single-user session, brokers are not involved. When a client starts a single-user database session, the database also starts, allowing only the client to connect directly to the database. When the client stops the session, the database shuts down.

You will learn about each of these sessions next.

For details, see the following topics:

- Single-user interactive session
- Single-user batch session

Single-user interactive session



A single-user interactive session is one in which only a single user can interact with a database.

There are a number of administration tasks that require exclusive access to a database, such as:

- Schema changes
- Roll-forward recovery of a database
- Rebuilding of indexes
- Truncation of the BI files
- Converting a database from an older version of OpenEdge to a newer version

In addition, if your environment allows you to have regular downtime for database maintenance tasks, there are a number of tasks you may want to perform in single-user mode, such as:

- Offline dumping and loading of data
- Offline bulk loading of a database
- Offline backup and restoring of a database
- Offline adding files to the database
- Offline enablement of after-imaging
- Offline enablement of automatic AI file management

Starting and stopping a single-user interactive session

You can start a single-user interactive session against a database using:

- The PROWIN command in Windows
- The PRO command in UNIX

In Windows, the syntax of the PROWIN command is:

```
prowin db-name -1 –p procedure-name
```

In UNIX, the syntax of the PRO command is:

```
pro db-name –p procedure-name
```

Note: You can also start a single-user interactive session against a database in character mode using the _PROGRES command in either Windows or UNIX. It has the same syntax as PROWIN.

Assume that the appdb database is offline and there is an OpenEdge procedure file, **dispcust.p**, which displays customer information.

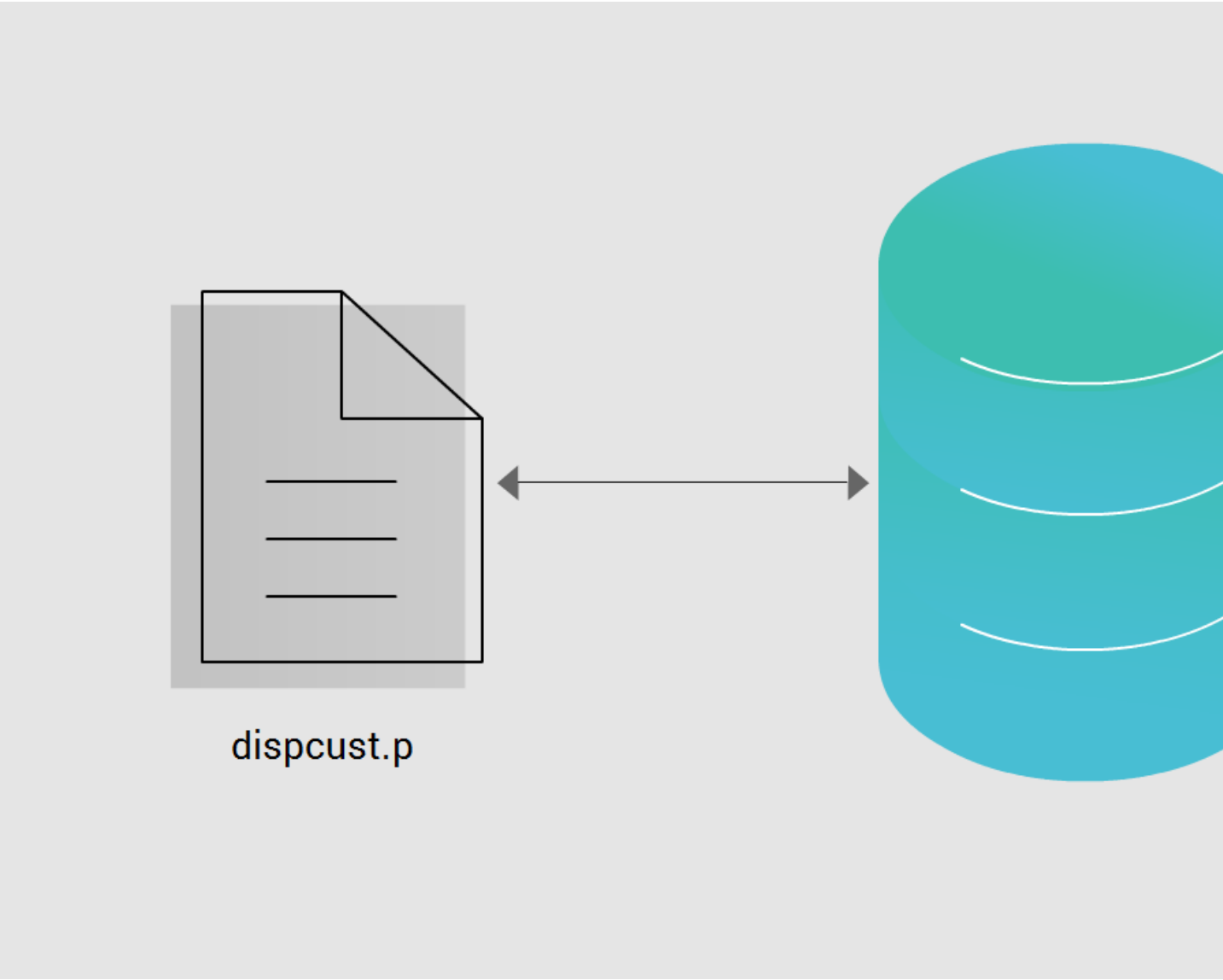
In Windows, you can run **dispcust.p** in a single-user interactive session against **appdb**:

```
prowin appdb -1 –p dispcust.p
```

In UNIX, you can perform the same task using this command:

```
pro appdb –p dispcust.p
```

Single-user batch session



A single-user batch session is one in which only a single user can connect to a database at a scheduled time, execute a procedure, and disconnect from the database after the procedure runs completely.

There are a number of resource-intensive batch jobs that you may need to perform periodically, such as:

- Create daily, weekly, or monthly reports.
- Generate database statistics.
- Perform large-scale database updates.
- Update database schema.

In these cases, you should start a database in a single-user batch session.

Starting and stopping a single-user batch session

You can start a single-user batch session against a running database using:

The PROWIN command in Windows

The BPRO command in UNIX

In Windows, the syntax of the PROWIN command is:

```
prowin db-name -1 -b –p procedure-name [> output-file]
```

In UNIX, the syntax of the BPRO command is:

```
bpro db-name -p procedure-name [> output-file]
```

Note: You can also start a single-user batch session against a database in character mode using the _PROGRES command in either Windows or UNIX. It has the same syntax as PROWIN.

Assume that the appdb database is offline and there is an OpenEdge procedure file, **dispcust.p**, which displays customer information.

In Windows, you can run **dispcust.p** in a single-user batch session against the **appdb** database and output the results to a text file, **dispcust.txt**:

```
prowin appdb -1 –b –p dispcust.p > dispcust.txt
```

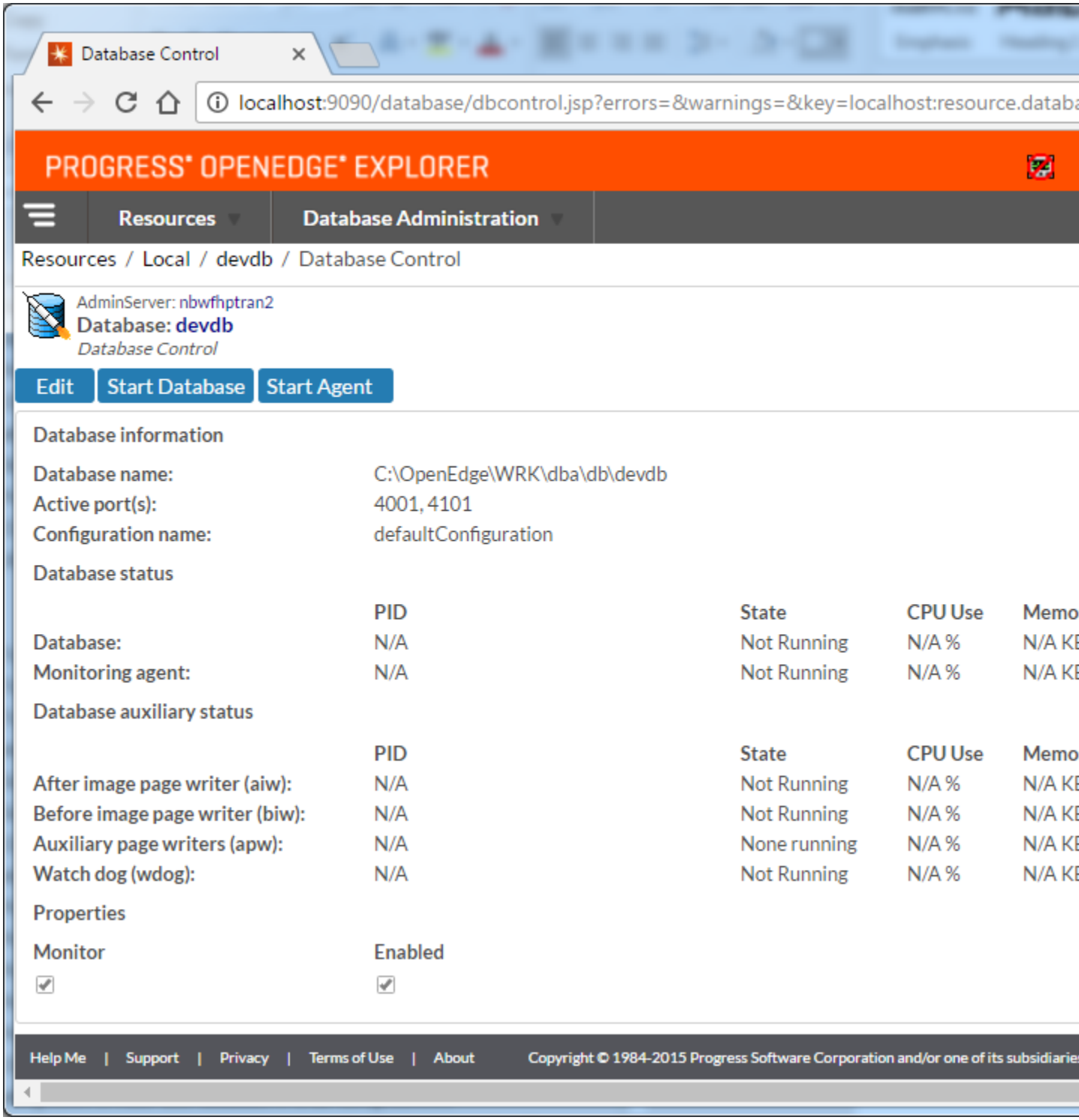
In UNIX, you can perform the same task using this command:

```
bpro appdb –p dispcust.p > dispcust.txt
```

After the batch job runs completely, the session ends and the database shuts down.

5

Starting up a multi-user database using OpenEdge Explorer



OpenEdge Explorer makes it easy for you to start up and shut down a database in a multi-user environment.

Recall that to start a multi-user database in Proenv, you needed to:

- Create a parameter file that specifies required startup parameter values.
- Start multiple brokers, including primary and secondary brokers.
- Start background processes.

Before you can start a database using OpenEdge Explorer, you must perform the same one-time setup tasks:

- Add a database to OpenEdge Explorer (including primary broker and background processes).
- Set database configuration properties (startup parameter values).
- Set server group properties (secondary brokers).

OpenEdge Explorer then maintains database configurations so that you can easily start up, monitor, manage, and shut down databases.

Next, you will learn how to how to perform each of these tasks.

For details, see the following topics:

- Procedure: Adding a database to OpenEdge Explorer
- Procedure: Setting database configuration properties
- Procedure: Setting server group properties
- Procedure: Starting up a multi-user database using OpenEdge Explorer
- Procedure: Shutting down a multi-user database using OpenEdge Explorer

Procedure: Adding a database to OpenEdge Explorer

The following procedure assumes that both the database and OpenEdge Explorer reside on the same local computer.

OpenEdge Explorer is part of the OpenEdge Unified Broker administration framework. The controlling component of that framework is the AdminServer. It enables you to configure and manage OpenEdge resources in your installation. Before you perform this procedure, you must ensure that the AdminServer is running.

Follow these steps to add a database to OpenEdge Explorer:

Steps	Action
1	In Windows, launch OpenEdge Explorer by clicking **Start > All Programs > Progress > OpenEdge 11.6 > OpenEdge Explorer**. In UNIX, launch a Web browser of your choice and enter the URL for OpenEdge Explorer using this syntax: `http://host-name:port-number` **Note:** The default port number for the WebServer for OpenEdge Explorer is 9090. The **Authentication** page opens.
2	Enter the username and password of the admin user and then click **Login**. The OpenEdge Explorer page opens.
3	Click **Resources > Database**. The **Adding a Managed Database** page appears.
4	In the **Database Display Name** field, enter the name of the database.
5	In the **Database Path** and **Filename** field, enter the full path and filename of the database. The filename is that of the .db file of the database; for example, `C:\OpenEdge\WRK\dba\db\appdb.db`.
6	In the **Database Port** field, enter a port for the database. Remember that the valid port range is from **2000** to **32767**.
7	In the **Database Broker Type** field, if you have multiple server types, select **4GL** for the primary broker. Later, you can configure SQL for the secondary broker (also known as a server group).
8	Select the **AutoStart Database Broker** option to start the broker automatically when the AdminServer starts.
9	Select **Watch Dog Process (WDOG)** so that the database uses a dedicated watchdog for better performance.
10	If you have the OpenEdge Enterprise RDBMS license: • Select After Image Process (AIW) to start the AIW for optimal performance, if you have enabled after-imaging for the database. • Select Before Image Process (BIW) to start the BIW for optimal performance. • Set Asynchronous Page Writers (APWs) to 1. You can add more APWs later if needed.
11	Click **Submit** to add the database as a managed database resource. The **Database** page opens.

It is recommended that you use the breadcrumbs at the top-left corner of the Resources page to navigate within different areas of a database resource.

Procedure: Setting database configuration properties

After you add a database as a managed resource to OpenEdge Explorer, you set database configuration properties for the resource. These database configuration properties are the same as the startup parameters you learned about earlier.

In addition, you can also set other properties such as primary and alternate buffer pools, background writers, AI file management utility, performance parameters, and SQL configuration properties.

Follow these steps to set database configuration properties for a database resource:

Steps	Action
1	From the **Database** page of the database resource you want to configure, click the **Configuration** link, and then click the **configuration.db-name.defaultconfiguration** link. The **Database Configuration** page opens.
2	Click **Edit**, and then specify values for database configuration properties.
3	Click **Save** to save your changes.

Procedure: Setting server group properties

Configuring server group properties is the same as configuring a secondary broker.

By default, every database configuration includes a default server group (the primary broker), which you configured when you added the database. To add a secondary broker, you just add a server group and set its properties.

The server group properties include the same startup parameters you learned earlier:

- Service name or port number (`-S`)
- Number of servers (per broker) (`-Mpb`)
- Maximum clients per server (`-Ma`)
- Minimum clients per server (`-Mi`)

Follow these steps to set server group properties:

Steps	Action
1	From the **Database Configuration** page, scroll to the bottom of the page and click the **servergroup.db-name.defaultconfiguration.defaultservergroup** link. The **Server Group** page opens.
2	If you want to configure the primary broker, click **Edit**, and then specify values for server group properties. Click **Save** to save your changes.
3	To add a server group (that is, a secondary broker), click **Create**, give it a name, and then click **Save** to save it.
4	To configure the secondary broker, click **Edit**, and then specify values for server group properties. Click **Save** to save your changes.
5	To add more secondary brokers, repeat steps 3 and 4.

Procedure: Starting up a multi-user database using OpenEdge Explorer

Follow these steps to start up a multi-user database using OpenEdge Explorer:

Steps	Action
1	Click **Resources > Go to Resources**. The **Resources** page opens.
2	Click the name of the database that you want to start.
3	Click **Start Database**. The database is started. The State values of the database and background processes change to Running. The database is then ready for multi-user access. **Note:** You may have to refresh the page if the status does not change.

Procedure: Shutting down a multi-user database using OpenEdge Explorer

Follow these steps to shut down a multi-user database using OpenEdge Explorer:

Steps	Action
1	Click **Resources > Go to Resources**. The **Resources** page opens.
2	Click the name of the database that you want to shut down.
3	Click **Stop Database** and then click **OK**. The database is stopped. The State values of the database and background processes change to Not Running. **Note:** You may have to refresh the page if the status does not change.

6

Shutting down a multi-user database in UNIX using DBMAN

The Unified Broker framework includes not only OpenEdge Explorer and Progress® OpenEdge® Management, but also the DBMAN utility. You can use DBMAN to start, query, and stop databases. DBMAN is especially useful in UNIX. You can script and automate administration tasks using DBMAN.

One useful way of using DBMAN is to script shutting down a database when you want to perform tasks in single-user mode, such as index rebuild, offline backup or other maintenance tasks.

Before you can use DBMAN to shut down a multi-user database, you must ensure that the database has already been configured in OpenEdge Explorer. You also need to ensure that the AdminServer is running.

The command to stop a database using DBMAN is:

```
dbman –host host-name –port port|service –database db-name –config config-name
 –stop
```

Where:

`host-name`	The name of the host machine on which the AdminServer is running.The default name is localhost.
`port|service`	The port number or service name on which the AdminServer is running.The default port number is **20931**.

db-name	The logical name of the database configured in OpenEdge Explorer.
config-name	The name of the configuration for the database.

Suppose the AdminServer is running and the **appdb** database resource has already been configured in OpenEdge Explorer. In UNIX, to stop the **appdb** database, enter:

```
dbman -host localhost -port 20931 -database appdb -stop
```