· Job Scheduling Software On Oracle ·

Version 4.1

Installation, Upgrade and Administration Guide

www.cronology.co.uk

1 System Requirements 3
1.1 Cronology Server 3
1.2 Cronology Console 3
2 Cronology Server Installation 4
2.1 Pre-requisites 4
2.1.1 Installation Variables 4
2.1.2 Oracle RDBMS Requirements 5
2.2 Installation 6
2.2.1 Copy Installation Media 6
2.2.2 Create and Edit Installation Variables Script 6
2.2.3 Run Installation 6
2.2.4 Set-Up Optional Features 6
2.2.5 Console User Grants 7
2.2.6 Obtain License Key 7
3 Cronology Console Installation 8
3.1 Installation 8
3.2 Cronology Console Access 8
4 Cronology Server Upgrade 9
4.1 Pre-requisites 9
4.1.1 Upgrade Variables 9
4.1.2 Oracle RDBMS Requirements 10
4.2 Upgrade 11
4.2.1 Copy Installation Media 11
4.2.2 Create and Edit Upgrade Variables Script 11
4.2.3 Run Upgrade 11
4.2.4 Set-Up Optional Features 11
5 Cronology Console Upgrade 12
5.1 Upgrade 12
5.2 Cronology Console Access 12
6 Post Installation / Upgrade Considerations 13
6.1 De-installation 13
6.2 Changing the CRONOLOGY schema password 13
6.3 Job Parameters based on Oracle Sequences 13
6.4 IMPORTANT: Server Overwrite / Restore / Copy - License Keys 13
Appendix A - Cronology ADMIN Package 14
Appendix B - Cronology API Package 18
Appendix C - Disabling Job Poller Autostart 19
Appendix D - Resetting an Expired License Key 20

1 SYSTEM REQUIREMENTS

1.1 Cronology Server

Supported Unix Platforms:

- Solaris
- Tru 64 / OSF1
- HP-UX
- Linux (OEL)

If you with to use SNMP messaging you must ensure the relevant SNMP software is installed on your UNIX platform.

Supported Oracle Databases:

- Oracle 8.1.7 -> 11.2.0 (32 and 64 bit)

Supported Cronology Server Versions (applicable only if upgrading):

- 3.2
- 3.3
- 3.4

1.2 Cronology Console

The Cronology console requires Oracle Forms runtime 6.0.8.24.1 (i.e. Forms 6i with Patch 15 or above installed). Currently only Windows based Forms Runtime is supported (i.e. not web enabled). The console may work with lower patch versions or 6i however results may be unpredictable. **Forms Runtime 6i for Windows will NOT run against a database with an AL32UTF8 character set**

Supported Cronology Console Versions (applicable only if upgrading):

- 3.2
- 3.3
- 3.4

2 CRONOLOGY SERVER INSTALLATION

2.1 Pre-requisites

2.1.1 Installation Variables

During the installation you will need to specify the following information:

Company Name (max 32 chars - case insensitive)
Your company name - this information is stored as part of the license key information.

Database TNS Alias (case insensitive)
Leave null ('') for a local install (assuming your Oracle environment, ORACLE_SID etc are set correctly) - or specify a TNS alias for a remote installation.

SYS Password (case sensitive for 11g -> if SEC_CASE_SENSITIVE_LOGON = TRUE)
The password for the SYS account - required for granting privileges on objects/packages owned by SYS.

DBA Account Name (case insensitive)
The name of a database account/schema with the DBA role assigned.

DBA Password (case sensitive for 11g -> if SEC_CASE_SENSITIVE_LOGON = TRUE)
The password for the DBA account named above.

CRONOLOGY Password (case sensitive for 11g -> if SEC_CASE_SENSITIVE_LOGON = TRUE)
The required password for the CRONOLOGY schema. This schema owns all the CRONOLOGY tables and packages. It is a highly privilieged account - its password should be set accordingly.

oratab Directory (case sensitive)
The directory location of the Oracle oratab file.

Hide UNIX Commands (ON | OFF - case sensitive)
Determines whether CRONOLOGY operating system commands are visible on a ps listing. A value of ON means commands are not visible, OFF means commands are visible. The recommended setting is 'ON'. WARNING! If turned off then database logon details for running jobs may be visible on a ps listing.

Java Executable Directory (case sensitive)
The directory location of java executable used for running Java applications. If you are unsure enter the following command at the UNIX command prompt: unalias -a ; dirname `which java`

Temp Directory (case sensitive)
The directory location of the temp directory on the server (usually /tmp)

UNIX Command Directory (case sensitive)
The directory location of the UNIX commands on the server. If you are unsure enter the following command at the UNIX command prompt: unalias -a ; dirname `which ls`

Please have this information to hand **before** you begin the installation.

2.1.2 Oracle RDBMS Requirements

- Server installations may be done remotely over SQL*Net:
  - ensure all connect strings are working properly (i.e. configured in tnsnames.ora)
  - for 9i and above - as connections as SYS are required for the install, please ensure an oracle password file exists and the database remote_login_passwordfile parameter is not set to NONE
- The OS account running the install must have access to the Oracle loadjava utility
- The target Oracle database must have the Java option installed
- The following Oracle packages must be installed and valid on the target database:
  - DBMS_APPLICATION_INFO
  - DBMS_LOCK
  - DBMS_OUTPUT
  - DBMS_OBFUSCATION_TOOLKIT
  - DBMS_SYSTEM
  - DBMS_SESSION
  - DBMS_UTILITY
  - DBMS_ALERT
  - DBMS_PIPE
  - DBMS_SHARED_POOL
  - DBMS_XPLAN (10g and above)
  - UTL_SMTP
  - UTL_RAW
  - UTL_TCP
  - UTL_FILE
- Oracles initialisation parameters
  - **java_soft_sessionspace_limit**=0 (meaning default 1 MB is used).
  - **java_max_sessionspace_size**=0 (meaning default of 4GB is used).
  - for versions of Oracle below 9.2 **utl_file_dir** MUST contain the server temp directory (usually /tmp). For Oracle 9.2 and above no change is required to utl_file_dir.

IMPORTANT:

- your server application code (e.g. SQL scripts / shell scripts) **MUST** be readable by the Unix account the Oracle database is running from (typically 'oracle')

2.2 Installation

The installation should be undertaken by a DBA.

2.2.1 Copy Installation Media

- logon to the Unix server
- create an installation directory and copy / ftp the Server.zip file from the installation media into the directory (N.B. the zip file should be FTP'd in **BINARY** mode)
- cd into the directory
- unzip Server.zip

2.2.2 Create and Edit Installation Variables Script

Rather than prompt the user for input during the installation, a script containing all required installation parameters must be setup before running the installation. This is useful as different parameter set ups can be saved for future use / reference.

- copy the install_vars_MASTER.sql script to a name of your choosing. You may choose whatever naming convention you like for the file but typically the name would include the server and database name the install is relevant for
- edit the file and enter all the relevant details – the template gives descriptions and examples for each parameter
- validate your entries and save the file

2.2.3 Run Installation

To run the installation:

- logon to SQL*Plus (any user will do as the installation will reconnect as required)
- run the installation script providing your variable script name as a parameter:
    - @install <your variable script name>
- be sure to read the 'Important Information' displayed when the installation completes

The installation should complete without any errors. If the install fails for any reason you should:

- look to address any database related issues (contact Cronology support if needs be)
- de-install (see below)
- re-run the install

2.2.4 Set-Up Optional Features

Use the CRONOLOGY.ADMIN package (see Appendix A) to set up any additional features you require:

- SET_SMTP_SETTINGS: Set your SMTP server details if you wish Cronology to use e-mail messaging
- SET_SOCKET_SETTINGS: Set your socket server details if you wish Cronology to use socket messaging
- SET_SNMP_EXE_DIR: Identify the location of the snmptrap (snmp_trapsnd for Tru64) executable if you wish Cronology to use SNMP trap messaging
- SET_MESSAGE_TEMPLATE: If you wish to define your own message templates for bespoke, socket or SNMP messaging (mandatory for SNMP messaging)

2.2.5 Console User Grants

This can be done logged on as a DBA or the CRONOLOGY schema. Grant CRONOLOGY_READONLY, CRONOLOGY_OPERATOR or CRONOLOGY_ADMINISRATOR role to required user accounts. You will need at least one CRONOLOGY_ADMINISTRATOR user in order that they can grant Cronology Console access to other users.

2.2.6 Obtain License Key

If you do not already have a valid license key for this database obtain one from Cronology Ltd – specify the database name / database ID displayed under 'Important Information' when the installation has completed. This key will be required when the console first connects to this installation.

3 CRONOLOGY CONSOLE INSTALLATION

3.1 Installation

It is recommended that the client console code resides on a shared network drive.

- create a directory on a shared drive
- copy the client code to the new directory
- amend the Cronology shortcut such that:
  - the "target" points to the correct location of the ifrun60.exe (Forms6i runtime executable)
  - the "start in" directory is the new directory just created

Users may then access the shared drive and drag and drop the shortcut to their desktop for easy access to the console. User ini files will be created in this directory as users log on and log off - keeping a history of connections and settings etc. **Please ensure end users have write access to this directory.**

Ensure all users have valid database accounts and have been granted **one** of the available Cronology roles: CRONOLOGY_OPERATOR, CRONOLOGY_ADMINISTRATOR or CRONOLOGY_READONLY.

CRONOLOGY_ADMINISTRATOR
Has full access to all console functions, i.e. they can create jobs and parameters and manipulate the schedule.

For administrator users, when accessing the console for the first time after installation:

- enter your license key
- ensure the messaging options (Main Menu – Settings – Messaging Options) are set correctly, in particular operator e-mail address

CRONOLOGY_OPERATOR
Can only view the schedule, execute, cancel and kill jobs. They may not alter the schedule (i.e. cannot create or amend jobs).

CRONOLOGY_READONLY
Can only view the schedule and associated job logs. Access is read only, they may not alter the schedule, or execute, cancel or kill jobs. They may not alter the schedule (i.e. cannot create or amend jobs).

3.2 Cronology Console Access

You cannot create database accounts via the console - this remains a DBA task (as should be the case) - please ask your DBA to create the relevant database accounts.

Existing users with the CRONOLOGY_ADMINISTRATOR role can then grant subsequent users access via the Console (Main Menu - Settings - Console - Users - Access).

4 CRONOLOGY SERVER UPGRADE

4.1 Pre-requisites

4.1.1 Upgrade Variables

During the upgrade you will need to specify the following information:

Database TNS Alias (case insensitive)
Leave null ('') for a local upgrade (assuming your Oracle environment, ORACLE_SID etc are set correctly) - or specify a TNS alias for a remote upgrade.

SYS Password (case sensitive for 11g -> if SEC_CASE_SENSITIVE_LOGON = TRUE)
The password for the SYS account - required for granting privileges on objects/packages owned by SYS.

CRONOLOGY Password (case sensitive for 11g -> if SEC_CASE_SENSITIVE_LOGON = TRUE)
The password for the existing CRONOLOGY schema.

oratab Directory (case sensitive)
The directory location of the Oracle oratab file.

Hide UNIX Commands (ON | OFF - case sensitive)
Determines whether CRONOLOGY operating system commands are visible on a ps listing. A value of ON means commands are not visible, OFF means commands are visible. The recommended setting is 'ON'. WARNING! If turned off then database logon details for running jobs may be visible on a ps listing.

SMTP Details (password case sensitive)
This version of Cronology no longer relies on the UNIX mailx program for sending e-mails. Instead e-mails are sent directly from within Oracle but the following details must be specified for communication with your e-mail SMTP server:

- SMTP server name / IP address
- SMTP server port

To determine your server name / server IP address issue the following UNIX command:

echo test | mailx -v <your_email_address>

You should see a line similar to: ... Connecting to **<SMTP server name/ IP address>** via relay ...

Port 25 is most common / the default.

Please have this information to hand **before** you begin the installation.

4.1.2 Oracle RDBMS Requirements

- Server installations may be done remotely over SQL*Net:
  - ensure all connect strings are working properly (i.e. configured in tnsnames.ora)
  - for 9i and above - as connections as SYS are required for the install, please ensure an oracle password file exists and the database remote_login_passwordfile parameter is not set to NONE
- The OS account running the install must have access to the Oracle loadjava utility
- The following Oracle packages must be installed and valid on the target database:
  - DBMS_APPLICATION_INFO
  - DBMS_LOCK
  - DBMS_OUTPUT
  - DBMS_OBFUSCATION_TOOLKIT
  - DBMS_SYSTEM
  - DBMS_SESSION
  - DBMS_UTILITY
  - DBMS_ALERT
  - DBMS_PIPE
  - DBMS_SHARED_POOL
  - DBMS_XPLAN (10g and above)
  - UTL_SMTP
  - UTL_RAW
  - UTL_TCP
  - UTL_FILE

4.2 Upgrade

The upgrade should be undertaken by a DBA.

4.2.1 Copy Installation Media

- logon to the Unix server
- create an upgrade directory and copy / ftp the Server.zip file from the installation media into the directory (N.B. the zip file should be FTP'd in **BINARY** mode)
- cd into the directory
- unzip Server.zip

4.2.2 Create and Edit Upgrade Variables Script

Rather than prompt the user for input during the upgrade, a script containing all required upgrade parameters must be setup before running the upgrade. This is useful as different parameter set ups can be saved for reference.

- copy the upgrade_vars_MASTER.sql script to a name of your choosing. You may choose whatever naming convention you like for the file but typically the name would include the server and database name the upgrade is relevant for
- edit the file and enter all the relevant details – the template gives descriptions and examples for each parameter
- validate your entries and save the file

4.2.3 Run Upgrade

As part of the upgrade you will be asked if you wish to backup your existing Cronology installation – **it is highly recommended you do this** in case you need to restore / rollback the upgrade. N.B. The upgrade is re-runnable i.e. if it fails it can simply be run again.

To run the upgrade:

- logon to SQL*Plus (any user will do as the installation will reconnect as required)
- run the installation script providing your variable script name as a parameter:
    - @upgrade <your variable script name>
- be sure to read the 'Important Information' displayed when the installation completes

The upgrade should complete without any errors. If the upgrade fails for any reason you should:

- look to address any database related issues (contact Cronology support if needs be)
- re-run the upgrade script (it may be run as many times as necessary until it completes successfully)

4.2.4 Set-Up Optional Features

Use the CRONOLOGY.ADMIN package (see Appendix A) to set up any additional features you require:

- SET_SNMP_EXE_DIR: Identify the location of the snmptrap (snmp_trapsnd for Tru64) executable if you wish Cronology to use SNMP trap messaging
- SET_MESSAGE_TEMPLATE: If you wish to define your own message templates for bespoke, socket or SNMP messaging (mandatory for SNMP messaging)

5 CRONOLOGY CONSOLE UPGRADE

5.1 Upgrade

Locate your existing installation and replace the following files from the upgrade media (you may wish to backup / rename the previous versions first):

- cronology.fmx
- cronology.wav

5.2 Cronology Console Access

A new database role is included in this release:

CRONOLOGY_READONLY
Can only view the schedule and associated job logs. Access is read only, they may not alter the schedule, or execute, cancel or kill jobs. They may not alter the schedule (i.e. cannot create or amend jobs).

Discuss with your Cronology administrators (users granted the CRONOLOGY_ADMINISTRATOR role) who (if any) should be granted this new role. Either the DBA can grant this role or administrators can grant it via the Console (Main Menu - Settings - Console - Users - Access).

6 Post Installation / Upgrade Considerations

6.1 De-installation

To de-install the Cronology Server simply run deinstall.sql as a DBA - this drops:

- the CRONOLOGY schema and all its associated objects
- the CRONOLOGY_READONLY, CRONOLOGY_OPERATOR and CRONOLOGY_ADMINISTRATOR roles
- the CRONOLOGY context
- the CRONOLOGY network access controls (11g onwards)

THIS COMPLETELY DE-INSTALLS THE CRONOLOGY INSTALLATION (IT DOES NOT ROLLBACK AN UPGRADE).

6.2 Changing the CRONOLOGY schema password

To change the CRONOLOGY password a DBA should use the CRONOLOGY.ADMIN.SET_SERVER_PASSWORD procedure.

This changes the password for the CRONOLOGY schema and also sets the new password (encrypted) in the system parameters table. It is imperative the schema password and system parameter entry are kept in sync, i.e. always use this procedure and do not issue an ALTER USER CRONOLOGY IDENTIFDIED BY <password> command stand alone.

6.3 Job Parameters based on Oracle Sequences

Job parameters are validated / evaluated when saved via the console. This is to ensure all parameters are valid before being used by a job. If a parameter requests the next value of an Oracle sequence then it will be incremented each time it is saved via the parameter maintenance screen.

6.4 IMPORTANT: Server Overwrite / Restore / Copy - License Keys

Keep a log of your supplied license keys and to which database names they belong. If you are overwriting / restoring an installation from another database and the database names differ – you will be prompted to enter the valid license key the first time the console connects to the restored installation.

If you are creating a new database from a copy and you do not already have a key for the new database name you will be required to obtain a key from Cronology Ltd.

Appendix A - Cronology ADMIN Package

The ADMIN package provided with this release allows a DBA to perform useful administrative tasks. **All procedures that update data implicitly perform a commit.** Procedures in the package are as follows:

- **FUNCTION GET_PARAMETER RETURNS VARCHAR2**
  Returns the supplied system parameter value (entries in CRONOLOGY.PARAMETERS table with a PARAM_ID < 0) as seen by the Cronology Server and Console sessions (i.e. what is currently held in the Cronology context)

- **PROCEDURE MARK_JOB_AS_KILLED**
  Forces a jobs status to KILLED (last resort if cannot kill via console / OS ... i.e. process id not captured)

- **PROCEDURE PURGE_HISTORY**
  Purges **all** historical job information but leaves the schedule (jobs, parameters, dependencies etc) intact.

- **PROCEDURE PURGE_SCHEDULE**
  Purges the entire schedule – all parameter and job information is purged along with all historical job information.

- **PROCEDURE RELOAD_CONTEXT**
  Reloads the system parameters into the Cronology context based on the current values in the CRONOLOGY.PARAMETERS table. For troubleshooting purposes only.

- **PROCEDURE RESET_JOB_SEQUENCE**
  Resets the sequence used for jobs. Only use this if you have removed the entire schedule (including offline jobs).

- **PROCEDURE RESET_PARAMETER_SEQUENCE**
  Resets the sequence used for job parameters. Only use this procedure if you have removed all job parameters.

- **PROCEDURE SET_MESSAGE_TEMPLATE**
  Use this procedure to define a template / format for messages used for either the bespoke, socket or SNMP messaging. A default template may be set for all messages or individual templates can be set for each message type. See the MESSAGE_TEMPLATES table for available templates. When using templates Cronology uses the following precedence:

  1) message type specific template (e.g. SOCKET_TEMPLATE_KILLED for job kill notifications via socket)
  2) default template (SOCKET_TEMPLATE_DEFAULT, BESPOKE_MESSAGE_DEFAULT or SNMP_MESSAGE_DEFAULT)
  3) standard message (no templates defined i.e. all templates are set to <NONE>) – only BESPOKE and SOCKET messaging will send a standard message, SNMP messaging MUST have templates defined. N.B. E-mail messaging does not use templates. E-mail messages will always use the 'standard' Cronology message.

For example, to set the default socket template:

exec admin.set_message_template('SOCKET_TEMPLATE_DEFAULT','Job: #JOB_NAME# Status: #STATUS# Timestamp: #SYSDATE_DD-MON-YYYY HH24:MI:SS#')

SNMP trap messaging requires the template to contain all the command line options that would normally be passed to the snmptrap (snmp_trapsnd for Tru64) UNIX executable, e.g.

exec admin.set_message_template('SNMP_TEMPLATE_DEFAULT','**-v 1 -c public localhost TRAP-TEST-MIB::demotraps localhost 6 17 "" SNMPv2-MIB::sysLocation.0 s "**Job: #JOB_NAME# Status: #STATUS# Timestamp: #SYSDATE_DD-MON-YYYY HH24:MI:SS#**"**')

When defining templates the following substitution variables may be used (the # characters must be included in the template for the substitution to take place):

#JOB_NAME#	The job name that triggered the message
#STATUS#	The status of the triggering event, status' can be: 'KILLED', 'OVERDUE', 'CANCELLED', 'FAILED', 'WAITING', 'RUNTIME' (job has exceeded runtime warning threshold), 'INTERNAL' (internal Cronology error)
#SUBJECT#	The default / standard Cronology message
#INSTANCE_NAME#	The Oracle instance name Cronology is running on
#HOST_NAME#	The host name Cronology is running on
#INFO#	Additional information Cronology may provide with the message
#SYSDATE_<DATE FORMAT>#	The current date and time, please provide a valid Oracle date format e.g. #SYSDATE_DD-MON-YYYY HH24:MI:SS#
#USER_NAME#	The database user who initiated the message (only really useful for job kill, cancellation and resolved messages)
#JOB_PRIORITY#	The priority of the job that triggered the message
#JOB_MESSAGE_TEXT#	The supplemental messaging information defined for the job

To reset / clear a template call the procedure with the template name and no / a null template (template gets set to <NONE>). You may provide the Oracle wildcard (%) character in the template name parameter if you wish to update multiple templates at once.

- **PROCEDURE SET_PARAMETER**
  Sets the supplied system parameter in both the CRONOLOGY.PARAMETERS table and the Cronology context.

- **PROCEDURE SET_SERVER_PASSWORD**
  For resetting the CRONOLOGY schema password. This procedure will set the password for the CRONOLOGY schema and also set the encrypted version of the password in the PARAMETERS table for use by the Cronology background processes (Job Poller, Log Writer and Message Server). For 11g onwards – if the CRONOLOGY schema approaches password expiry the password is automatically reset to its existing value by the Cronology application. This is done to avoid the background processes failing / generating ORA-28002 errors**. This effectively means the password will never expire** (assuming immediate password reuse is allowed). If required, use this procedure to change the password at regular intervals as required by your organisation.

- **PROCEDURE SET_SMTP_SETTINGS**
  If e-mail messaging is required, use this procedure to set the SMTP server details. Run with no parameters to "reset" (details set to <NONE>). To determine your server name / server IP address issue the following UNIX command:

  echo test | mailx -v <your_email_address>

  You should see a line similar to: ... Connecting to **<SMTP server name/ IP address>** via relay ...

  Port 25 is most common / the default. If username and password authentication are not required by the SMTP server please leave the sender e-mail address, username and password parameters null. If authentication is required, in most cases the SMTP sender e-mail address should be the same as the SMTP username.

- **PROCEDURE SET_SNMP_EXE_DIR**
  If SNMP trap messaging is required, use this procedure to set the location of the snmptrap (snmp_trapsnd for Tru64) executable. On Linux, Solaris and HP-UX use the following command to determine the correct directory:

  unalias -a ; dirname `which snmptrap`

  On Tru64 use:

  unalias -a ; dirname `which snmp_trapsnd`

  Use the SET_MESSAGE_TEMPLATE procedure to set the command line parameters for the SNMP traps.

- **PROCEDURE SET_SOCKET_SETTINGS**
  If socket messaging is required, use this procedure to set the socket server details. Run with no parameters to "reset" (details set to <NONE>)

- **PROCEDURE START_ALL**
  Starts all Cronology Server processes (Job Poller, Log Writer and Message Server)

- **PROCEDURE START_JOB_POLLER**
  Starts the Cronology Job Poller

- **PROCEDURE START_LOG_WRITER**
  Starts the Cronology Log Writer

- **PROCEDURE START_MESSAGE_SERVER**
  Starts the Cronology Message Server

- **PROCEDURE STOP_ALL**
  Stops all Cronology Server processes (Job Poller, Log Writer and Message Server)

- **PROCEDURE STOP_JOB_POLLER**
  Stops the Cronology Job Poller

- **PROCEDURE STOP_LOG_WRITER**
  Stops the Cronology Log Writer

- **PROCEDURE STOP_MESSAGE_SERVER**
  Stops the Cronology Message Server

- **FUNCTION SYSTEM_INFO RETURNS VARCHAR2**
  Returns Cronology system information e.g. Oracle and UNIX OS details, Cronology version and current system parameters.

- **PROCEDURE TEST_OS_CMDS**
  Tests the environment is configured correctly in order to run OS commands. The procedure will complete successfully if the environment is ok, or will raise an appropriate exception.

- **PROCEDURE UPD_SCHEDULE_CONNECT_STRING**
  Will update all jobs in the schedule, changing an old connect string to a new connect string. This procedure converts connect strings to upper case.

- **PROCEDURE UPD_SCHEDULE_PASSWORD**
  Will update all jobs in the schedule updating a password for a given username and connect string. If no connect string is defined for the jobs then supply NULL as the connect string parameter. This procedure converts connect strings and usernames to upper case.

Appendix B - Cronology API Package

The API package has execute privilege granted to PUBLIC and provides a number of procedures that application developers may use to interface with the Cronology Server processes:

- **PROCEDURE EXECUTE_JOB**
  This procedure will execute a job in the Cronology schedule. This procedure is overloaded to take either a job name (CRONOLOGY.JOB table JOB_NAME column) or the job id (CRONOLOGY.JOB table JOB_ID column – use only if job ids are consistent across your environments). Other parameters: P_RSA = run stand alone (Y or N), P_UOP = use override parameters (Y or N), P_FAIL_ON_ERROR = if the job fails raise an exception (Y or N). This procedure allows jobs to be executed programmatically from within PL/SQL program units if needs be.

- **PROCEDURE LOG_LINE**
  This procedure may be called from within a PL/SQL program unit to allow real-time output to the job log when the job is run via Cronology. Traditionally DBMS_OUTPUT would be used to produce output from within PL/SQL, this however waits until the program unit has completed before producing the output. Programmers may use CRONOLOGY.API.LOG_LINE where they would normally use DBMS_OUTPUT.PUT_LINE. If the program unit is run outside of Cronology then LOG_LINE acts exactly the same as DBMS_OUTPUT.PUT_LINE. An option parameter (P_MODE) exists for the procedure. If LOG_LINE is called with P_MODE = 'A' then the line will be appended to previous line.

- **PROCEDURE DELETE_HISTORY**
  A procedure named DELETE_HISTORY has been provided in the CRONOLOGY.API package to delete historical job information. This is a recommended Cronology housekeeping routine, as such a job should be created in the schedule to call the CRONOLOGY.API.DELETE_HISTORY procedure at an interval and time that suits resource availability on your system. The job requires no parameters (it deletes data based on the JOB_HISTORY_RETENTION parameter, set via the Console under Main Menu -> Settings -> Advanced Settings). The CRONOLOGY.API package has execute privilege granted to PUBLIC, but internal validation inside the DELETE_HISTORY procedure ensures that only users granted the CRONOLOGY_ADMINISTRATOR or CRONOLOGY_OPERATOR role may execute it.

Other procedures exist in the API package are not intended for general use, they are primarily used for supporting the Cronology processes.

Appendix C - Disabling Job Poller Autostart

If a database has been shut down (with Autostart was left ON) and the DBA does not wish the Cronology Job Poller to automatically start (and hence possibly launch jobs) when the database starts up, one of the following two methods may be employed:

Method 1

- startup mount;
- execute dbms_application_info.set_module('CRONOLOGY','DISABLE_AUTOSTART');
- alter database open;

The Job Poller will detect this setting when starting up and shut itself down - it will also set the JOB_POLLER_AUTO_START system parameter to 'OFF' so that subsequent database startups will not start the Job Poller. Autostart can be turned on again via the Cronology Console.

Method 2

- startup mount;
- alter system set "_system_trig_enabled" = FALSE;
- alter database open;

This method will disable the trigger that starts the Job Poller. Depending on how this parameter is set (memory only or permanently in the database pfile / spfile) Autostart should be turned off via the Cronology Console to avoid future Autostarts when the database is started.

WARNING: ALL database startup triggers will be disabled if this method is used.

APPENDIX D - RESETTING AN EXPIRED LICENSE KEY

If a license key has been allowed to expire then it can no longer be updated via the Cronology Console.

A valid key must be obtained from Cronology Ltd. A DBA should then use the following PL/SQL to update the license key:

```
declare
   lic_key varchar2(64) := '&LICENSE_KEY';
begin
   begin
    -- If license has expired the package initialization sections will raise an error, ignore it ...
       cronology.admin.set_parameter('LICENSE_KEY_PART_1',lic_key);
   exception
      when others then
         null;
   end;
   cronology.admin.set_parameter('LICENSE_KEY_PART_1',lic_key);
end;
/
```