Caché Data Integrity Guide

Version 2018.1
2024-05-02

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Caché Data Integrity Guide
Caché Version 2018.1 2024-05-02
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble® InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

List of Figures

List of Tables

About This Book

As organizations rely more and more on computer applications, it is vital to safeguard the contents of databases. This guide explains the many mechanisms Caché uses to maintain the integrity of your data.

Caché write image journaling technology protects against internal integrity failures due to system crashes. Caché backup and journaling systems provide rapid recovery from physical integrity failures. Logical database integrity is ensured through transaction processing, locking, and automatic rollback.

The following topics are addressed:

- Introduction to Data Integrity
- Write Image Journaling and Recovery
- Backup and Restore
- Journaling
- Shadowing
- Cluster Journaling
- Data Consistency on Multiple Systems

For detailed information, see the Table of Contents.

For general information, see *Using InterSystems Documentation*.

1

Introduction to Data Integrity

The integrity of the Caché database is protected from system failure by the features described in this guide.

1.1 Fundamental Data Integrity Protection

In general, there are two different levels at which integrity can be viewed:

- Structural database integrity, or physical integrity, refers to the contents of the database blocks on disk. To have structural integrity, the database blocks must be self-consistent and the globals traversable. Structural integrity during a system crash is maintained by Caché write image journal (WIJ) technology, as described in the chapter "Write Image Journaling and Recovery", and Caché's internal algorithms.
- Logical integrity refers to the data represented by the globals within the database, and encompasses the self-consistency of the data created by the application, its transactional integrity, and its being up-to-date with the real world. Logical integrity during a system crash is maintained by Caché journaling (see the "Journaling" chapter) and transaction processing. (Other aspects of logical integrity are under the control of application code, through proper use of interlocks, transactions, and other mechanisms specific to the programming paradigm that the application employs.)

Automatic WIJ and journal recovery are fundamental components of the InterSystems "bulletproof" database architecture that protects Caché databases from system failures.

1.2 Integrity Verification and Recovery Mechanisms

Although system crashes alone cannot lead to a loss of integrity, there is always a possibility that a storage device will fail catastrophically, sustain physical damage, or be tampered with. In that case, the integrity of the database, WIJ and journals can become compromised. To compensate for such disasters, Caché provides the following features:

- Tools for checking the structural integrity of databases, described in Verifying Structural Integrity in this chapter.
- Backup mechanisms, as described in the chapter "Backup and Restore".
- Journaling-based logical data replication for automatic failover and disaster recovery through mirroring (see the "Mirroring" chapter of the *Caché High Availability Guide*) and shadowing (see the "Shadowing" chapter of this guide).
- DataCheck, a tool for checking the consistency of data between multiple systems when technologies such as mirroring and shadowing maintain a replicated copy of data (see the chapter "Data Consistency on Multiple Systems").

- Journaling on ECP-based shared-disk clustered systems in Caché (see the "Cluster Journaling" chapter).

1.3 Verifying Structural Integrity

An integrity check lets you verify the *structural* integrity (see Fundamental Data Integrity Protection) of a set of databases, or subset of globals within the databases.

The benefits of running an integrity check are as follows:

- Integrity check can be integrated into your backup strategy to ensure that at the time of backup, the copy of the database was intact and that no errors were introduced during the backup itself, as discussed in External Backup in the "Backup and Restore" chapter.
- Integrity check can detect corruption before users encounter it, giving time to make a plan before users are impacted.
- Regular integrity checks provide a means by which the origin of any structural integrity problems that are found can be more accurately pinpointed in time, increasing the likelihood of identifying the root cause.

An integrity check lets you verify the integrity of all globals in selected databases, or of selected globals stored in a single specified database. You can run an integrity check from the Management Portal or using the **^Integrity** utility in a Terminal window. This section covers the following topics:

- Integrity Check False Positives
- Integrity Check Output
- Checking Database Integrity Using the Management Portal
- Checking Database Integrity Using the ^Integrity Utility

1.3.1 Integrity Check False Positives

Running an integrity check on a volatile database may result in the false reporting of database integrity errors due to ongoing database updates.

When an integrity check is executed from the Management Portal or by the Task Manager, as described in Checking Database Integrity Using the Management Portal, it runs in the background, and automatically retests any globals in which errors are detected. Output from an integrity check that includes this automatic second pass reports on errors in the following manner:

- If an error was detected in a global in the first pass but not in the second pass, the first error is assumed to be a false positive and no error is reported.
- If the error detected in a global in the second pass differs from the error detected in the first pass, only the second-pass error is reported, with the text `These errors in global <global_name> differ from the errors prior to the retry`.
- If the same error is detected in a global in both passes, the error is reported with the message `When retried the errors in global <global_name> remained unchanged`.

Integrity checks executed manually using the **^Integrity** utility or one of the entry points described in Checking Database Integrity Using the ^Integrity Utility do not retest globals reporting errors on the first pass. If errors are returned, repeat the check for that particular database.

Generally, for an integrity check run on an active system, errors that are not repeated in a second pass are false positives, while errors that persist in a second pass represent actual integrity problems. The latter must be investigated, and the former

may merit investigation as well, depending on the level of activity, the number of errors, and the extent to which false positives have previously occurred. The nature of your investigation will depend on your level of expertise and past experience of false positives. Steps you can take include:

- Running the integrity check again, if possible during a period of lower system activity.
- Running an integrity check on a restored copy of the most recent backup.
- Examining the range of data in question for clues to the root problem.
- Contacting the InterSystems Worldwide Response Center (WRC) for assistance.

The problem of false positives can be avoided by integrating integrity checks into your standard backup procedures, such as those described in External Backup in the "Backup" chapter of the *Caché Data Integrity Guide*, so that databases are checked immediately after taking a snapshot of the logical disk volume on which they reside.

1.3.2 Integrity Check Output

In addition to reporting any errors it encounters, the integrity check reports on the number of blocks in each global and the percentage of those blocks that is in use, breaking this information down by block level as well. For example, the following is a portion of the output of an integrity check on a SAMPLES database populated with 20,000 users:

```
File Name: c:\intersystems\20162555dec15a\mgr\integ.txt

Cache Database Integrity Check - Report Created 01/25/2016 10:41:16
System: BBINSTOCK6440 Configuration: 20162555DEC15A

No Errors were found.


Full Listing of Databases Checked


Directory: C:\InterSystems\20162555DEC15A\Mgr\samples\
0 globals with errors found

Global: Aviation.AircraftD                                  0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (6% full)
 Data Level:           # of blocks=64      512kb (87% full)
 Total:                # of blocks=65      520kb (85% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:15

Global: Aviation.AircraftI                                  0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=4      32kb (83% full)
 Total:                # of blocks=5      40kb (67% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:15

Global: Aviation.Countries                                  0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=1      8kb (52% full)
 Total:                # of blocks=2      16kb (26% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:15

Global: Aviation.CrewI                                      0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (1% full)
 Data Level:           # of blocks=5      40kb (90% full)
 Total:                # of blocks=6      48kb (75% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:15

Global: Aviation.EventD                                     0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (41% full)
 Data Level:           # of blocks=377      3,016kb (78% full)
 Big Strings:          # of blocks=776      6,208kb (72% full) # = 479
 Total:                # of blocks=1,154      9,232kb (74% full)
 Elapsed Time = 0.1 seconds, Completed 01/25/2016 10:41:15

Global: Aviation.EventI                                     0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=3      24kb (77% full)
 Total:                # of blocks=4      32kb (58% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:15

...
```

```
Global: ROUTINE                                              0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (1% full)
 Data Level:           # of blocks=6      48kb (78% full)
 Total:                # of blocks=7      56kb (67% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:16

Global: SYS                                                  0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=1      8kb (0% full)
 Total:                # of blocks=2      16kb (0% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:16

Global: Sample.CompanyD                                      0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=1      8kb (35% full)
 Total:                # of blocks=2      16kb (17% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:16

Global: Sample.CompanyI                                      0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=1      8kb (9% full)
 Total:                # of blocks=2      16kb (4% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:16

Global: Sample.PersonD                                       0 errors found
 Top/Bottom Pnt Level: # of blocks=1      8kb (0% full)
 Data Level:           # of blocks=5      40kb (81% full)
 Total:                # of blocks=6      48kb (67% full)
 Elapsed Time = 0.0 seconds, Completed 01/25/2016 10:41:16

...
```

When run from the Management Portal, the report begins with a listing of errors and warnings generated by the integrity check, if any. When run using the **^Integrity** utility, the error summary is provided at the end of the output.

1.3.3 Checking Database Integrity Using the Management Portal

To check the integrity of selected databases, or of selected globals stored in a single database, navigate to the Databases page of the Management Portal (**Home** > **System Operation** > **Databases**) and use the following procedure:

1. Click **Integrity Check** to display a list of databases.
2. Select the appropriate check boxes for the databases you want to check.

   If you want to check globals stored in a single database, select only the database that contains the globals you want to check, then click **Select Globals** to display a list of globals stored in the selected database. Select the globals you want to check, then click **Save**; if you do not select any globals from the list, all globals in the selected database are checked.
3. Specify the path of a log file in which to save the integrity check output (as described in Integrity Check Output). You can accept the default (*install_dir*/mgr/integ.txt), click **Browse** to choose an existing file, or enter your own file path.
4. If you want to stop checking the integrity of the database(s) upon encountering an error, select the **Stop after any error** check box.
5. Click **OK** to begin the integrity check. The integrity check process runs in the background.

Click **Integrity Log** to view the output from the most recent integrity check run using the portal. The path of this file and its contents are automatically displayed, but you can enter a new path, or browse for a different file, and click **View File** to view that file instead.

Integrity Check is also one of the default scheduled system background tasks in the Task Manager. You can schedule additional integrity checks if you wish, for example of different databases at different times. See Using the Task Manager in the "Managing Caché" chapter of the *Caché System Administration Guide* for more information about scheduling system tasks.

1.3.4 Checking Database Integrity Using the ^Integrity Utility

You can run a manual integrity check using the **^Integrity** utility by opening a Terminal windows, switching to the %SYS namespace, and entering **`do ^Integrity`**. This is similar to running an integrity check from the Databases page of the Management Portal, except that, as noted in Integrity Check False Positives, the **^Integrity** utility cannot recheck globals for which it finds errors in the first pass before completing and reporting its results, and it is therefore important to recheck globals for which errors are reported to eliminate false positives. (The Management Portal integrity check also distributes the integrity check across multiple jobs, instead of running a single job like the **^Integrity** utility.)

This routine includes the following additional entry points:

- **`Do CheckPointer^Integrity`** asks for a directory and a pointer block at which to start checking.
- **`Do Silent^Integrity(logfilename,dirlist)`** starts a background process that does an integrity check on selected or all databases and puts the output in a file specified by the *logfilename* parameter. The optional *dirlist* parameter ($LIST format) identifies selected databases to check; if *dirlist* is not specified, all databases are checked.

  This is the equivalent of doing **Integrity Check** from the **Databases** page (**System Operation** > **Databases**) of the Management Portal.
- **`Do Query^Integrity(logfilename,outdevice)`** does not run an integrity check, but puts the contents of the file specified by the *logfilename* parameter, the results saved from a previous run, out on the current device or the device specified in the optional parameter *outdevice*.

  If not specified, *outdevice* is the current terminal. Examples of *outdevice* are a printer, another display device, or another operating system file name. The latter makes a copy of *logfilename*.
- **`Do SilentGlobalCheck^Integrity(logfilename,dir,gbllist)`** starts a background process that does an integrity check on selected globals in a selected database and puts the output in a file specified by the *logfilename* parameter. The *dir* parameter identifies the database that contains the globals you want to check. The required *gbllist* parameter ($LIST format) identifies one or more globals to check.

  This is the equivalent of choosing **Select Globals** when doing **Integrity Check** from the **Databases** page (**System Operation** > **Databases**) of the Management Portal.
- **`Do CheckList^Integrity(outputglobal,dirlist,stopafteranyerror,listofglolist,maxproc)`** stores the results of integrity check, including information and warnings, in the global specified by the optional *outputglobal* parameter; if it is not specified, the results are stored in **^CacheTempIntegrityOutput($JOB)**. The optional *dirlist* parameter specifies a $LIST of all the directories you want to check; if it is not specified, all directories are checked. If the *stopafteranyerror* parameter is specified, checking on a directory stops when an error is found. The optional *listofglolist* parameter specifies a $LIST of $LISTs of global name, one for each directory specified in the *dirlist* parameter which, for example, lets you check all oddDEF globals in all directories by specifying `$LB($LB("oddDEF"))`; if there are fewer elements in the *listofglolist* parameter than in the list of directories, the last element is used for the rest of the directories. The optional *maxproc* parameter specifies the upper bound to the number of background jobs used: if it is <1, the number of cores (where the number of cores on the machine, as well as the number of directories to be checked, always act as an upper bound) on the machine are used; if it is 1, the integrity check is done directly in the foreground process.
- **`Do Display^Integrity(integritout,flags,dirsum)`** displays the results of integrity check. The optional *integritout* parameter specifies the name of the global where the results if integrity check were placed (by CheckList^Integrity); if it is not specified, it defaults to **^CacheTempIntegrityOutput($JOB)**. The optional *flags* parameter specifies can be: `0` (to display all messages), `1` (to display only errors and warnings), or `2` (to display only errors); if it is not specified, the default is `0`. If the optional *dirsum* parameter is specified and is not `0`, the display will include a summary of blocks for each Directory scanned.
- **`Do Exclude^Integrity`** asks for a list of databases to exclude from checking; entering `?` displays a list of mounted databases.

Note: While it is possible to use the **Mount a database** option of the **^DATABASE** routine to mount any CACHE.DAT file accessible to the instance (see the "Using Character-based Security Management Routines" appendix of the *Caché Security Administration Guide*), if this is done with a database that was deleted from, or was never added to, the Management Portal database configuration (see Configuring Databases in the "Configuring Caché" chapter of this guide), the database is not added to the Management Portal configuration and is therefore unavailable for portal database operations and for some routines, including the **^Integrity** utility described in this section.

2

Write Image Journaling and Recovery

Caché uses write image journaling to maintain the internal integrity of your Caché database. It is the foundation of the database recovery process.

This chapter discusses the following topics:

- Write Image Journaling
- Recovery
- Limitations

2.1 Write Image Journaling

Caché safeguards database updates by using a two-phase technique, write image journaling, in which updates are first written from memory to a transitional journal, CACHE.WIJ, and then to the database. If the system crashes during the second phase, the updates can be reapplied upon recovery. The following topics are covered in greater detail:

- Write Image Journal (WIJ)
- Two-Phase Write Protocol

2.1.1 Write Image Journal (WIJ) File

The Write daemon is activated at Caché startup and creates the write image journal (WIJ) file. The Write daemon records database updates in the WIJ before writing them to the Caché database.

By default, the WIJ file is named CACHE.WIJ and resides in the system manager directory, usually *install-dir*/Mgr, where *install-dir* is the installation directory. To specify a different location for this file, use the Management Portal:

1. Navigate to the Journal Settings page of the Management Portal (**System Administration** > **Configuration** > **System Configuration** > **Journal Settings**).
2. Enter the new location of the WIJ in the **Write image journal directory** box and click **Save**. The name must identify an existing directory on the system and may be up to 63 characters long. If you edit this setting for a clustered instance, restart InterSystems IRIS to apply the change; no restart is necessary for a standalone instance.
3. Enter the target size for the WIJ at the **Target size for the wij (MB) (0=not set)** prompt. The default of zero allows the WIJ to grow as needed but does not reserve space for this; entering a non-zero value reserves the specified space on the storage device.

For information about the two settings described, which are included in the instance's cache.cpf file, see targwijsiz and wijdir in the [config] section of the *Caché Parameter File Reference*).

2.1.2 Two-Phase Write Protocol

Caché maintains application data in databases whose structure enables fast, efficient searches and updates. Generally, when an application updates data, Caché must modify a number of blocks in the database structure to reflect the change.

Due to the sequential nature of disk access, any sudden, unexpected interruption of disk or computer operation can halt the update of multiple database blocks after the first block has been written but before the last block has been updated. The two-phase write protocol prevents this incomplete update from leading to an inconsistent database structure, which could occur with it. The consequences could be as severe as a database that is totally unusable, all data irretrievable by normal means.

The Caché write image journaling technology uses a two-phase process of writing to the database to protect against such events as follows:

- In the *first* phase, Caché records the updated blocks in the WIJ. Once it enters all updates to the WIJ, it sets a flag in the file and the second phase begins.
- In the *second* phase, the Write daemon writes the same set of blocks recorded in the WIJ to the database on disk. When this second phase completes, the Write daemon sets a flag in the WIJ to indicate it is deleted.

When Caché starts, it automatically checks the WIJ and runs a recovery procedure if it detects that an abnormal shutdown occurred. When the procedure completes successfully, the internal integrity of the database is restored. Caché also runs WIJ recovery following a shutdown as a safety precaution to ensure that database can be safely backed up.

2.2 Recovery

WIJ recovery is necessary if a system crash or other major system malfunction occurs. When Caché starts, it automatically checks the WIJ. If it detects that an abnormal shutdown occurred, it runs a recovery procedure. Depending on where the WIJ is in the two-phase write protocol process, recovery does the following:

- If the crash occurred after the last update to the WIJ was completed but before completion of the corresponding update to the databases—that is, during the second phase of the process—the WIJ is restored as described in WIJ Restore.
- If the crash occurred after the last WIJ update was durably written to the databases—that is, after both phases were completed—a block comparison is done between the most recent WIJ updates and the affected databases, as described in WIJ Block Comparison (Windows and UNIX®/Linux only).

2.2.1 WIJ Restore

If the WIJ is marked as "active," the Write daemon completed writing modified disk blocks to the WIJ but had not completed writing the blocks back to their respective databases. This indicates that WIJ restoration is needed. The recovery program, cwdimj, does the following:

- Informs the system manager in the console log (cconsole.log) file; see Monitoring Log Files in the "Monitoring Caché Using the Management Portal" chapter of the *Caché Monitoring Guide*.
- Performs Dataset Recovery.

Typically, all recovery is performed in a single run of the cwdimj program.

2.2.1.1 Dataset Recovery

A *dataset* is a specific database directory on a specific Caché system. The *cwdimj* program restores all datasets configured in the Caché instance being restarted after an abnormal shutdown.

The *cwdimj* program can run interactively or non-interactively. The manner in which it runs depends on the platform, as follows:

- Windows — Always runs non-interactively.
- UNIX®/Linux — Runs non-interactively until encountering an error, then runs interactively if an operator is present to respond to prompts.

Note: When the **ccontrol start quietly** command is used on UNIX/Linux systems, it always runs noninteractively.

When the recovery procedure is complete, *cwdimj* marks the contents of the WIJ as "deleted" and startup continues.

If an error occurred during writing, the WIJ remains active and Caché will not start; recovery is repeated the next time Caché starts unless you override this option (in interactive mode).

CAUTION: If you override the option to restore the WIJ, databases become corrupted or lose data.

The following topics are discussed in more detail:

- Interactive Dataset Recovery
- Noninteractive Dataset Recovery

Interactive Dataset Recovery

The recovery procedure allows you to confirm the recovery on a dataset-by-dataset basis. Normally, you specify all datasets. After each dataset prompt, type either:

- `Y` — to restore that dataset
- `N` — to reject restoration of that dataset

You can also specify a new location for the dataset if the path to it has been lost, but you can still access the dataset. Once a dataset has been recovered, it is removed from the list of datasets requiring recovery; furthermore, it is not recovered during subsequent runs of the cwdimj program should any be necessary.

Noninteractive Dataset Recovery

When the recovery procedure runs noninteractively, Caché attempts to restore all datasets and mark the WIJ as deleted. On Unix and Windows platforms, Caché first attempts a fast parallel restore of all datasets; in the event of one or more errors during the fast restore, datasets are restored one at a time so that the databases that were fully recovered can be identified. If at least one dataset cannot be restored:

- The cwdimj program aborts and the system is not started.
- Any datasets that were not successfully recovered are still marked as requiring recovery in the WIJ.

2.2.2 WIJ Block Comparison

Typically, a running Caché instance is actively writing to databases only a small fraction of the time. In most crashes, therefore, the blocks last written to the WIJ were confirmed to have been durably written to the databases before the crash; the WIJ is not marked "active", and there is no WIJ restore to be performed. When Caché starts up after such a crash, however, the blocks in the most recent WIJ updates are compared to the corresponding blocks in the affected databases as a form of rapid integrity check, to guard against starting the instance in an uncertain state after a crash that was accompanied

by a storage subsystem failure. The comparison runs for a short time to avoid impacting availability and asynchronous I/O is utilized to maximize throughput. If all blocks match, or no mismatch is detected within 10 seconds, startup continues normally. If a mismatch is found within this time, the results are as follows:

- The comparison operation continues until all available WIJ blocks have been compared.
- The mismatching WIJ blocks are written to a file called MISMATCH.WIJ in the WIJ directory.
- Normal startup is aborted and Caché starts in *single-user* mode with a message like the following:

```
There exists a MISMATCH.WIJ file.
        Startup aborted, entering single user mode.
        Enter Cache' with
             csession [instancename] -B
        and D ^STURECOV for help recovering from this error.
```

CAUTION: If you encounter MISMATCH.WIJ, contact InterSystems Worldwide Response Center (WRC) before proceeding.

This situation has implications for the integrity of your data and calls for immediate attention. However, performing the wrong action with MISMATCH.WIJ can worsen the situation. Unless you have experience with MISMATCH.WIJ, it is safer to revert to a known good backup and proceed from there or to contact the WRC for guidance.

Use the information that follows to determine the appropriate course of action. When your recovery procedures are complete, you must rename the MISMATCH.WIJ file, either using the **STURECOV** routine or externally, before Caché startup can continue. While the file is persistent and prevents normal startup of the instance, its contents are helpful in diagnosing why it was created.

Run the indicated command to perform an emergency login as system administrator (see Connecting to a Caché Instance in the "Using Multiple Instances of Caché" chapter of the *Caché System Administration Guide*).

You are now in the manager's namespace and can run the startup recovery routine with the command **Do ^STURECOV**. The following WIJ mismatch recovery message and menu appear on a UNIX®/Linux system:

```
The system crashed and some database blocks do not match what was
expected based on the contents of write image journal (the WIJ).
The WIJ blocks have been placed in the MISMATCH.WIJ file.  If any
database files, or the WIJ, were modified or replaced since the crash,
you should rename the MISMATCH.WIJ. Otherwise, MISMATCH.WIJ probably
contains blocks that were lost due to a disk problem.  You can view
those blocks and apply them if necessary.  When finished, rename the
MISMATCH.WIJ in order to continue startup.

1) List Affected Databases and View Blocks
2) Apply mismatched blocks from WIJ to databases
3) Rename MISMATCH.WIJ
4) Dismount a database
5) Mount a database
6) Database Repair Utility
7) Check Database Integrity
8) Bring up the system in multi-user mode
9) Display instructions on how to shut down the system

------------------------------------------------------------
H) Display Help
E) Exit this utility
------------------------------------------------------------
```

On a Windows system, options 8 and 9 are replaced by `8) Bring down the system prior to a normal startup`.

The appropriate actions in the event of a WIJ mismatch differ based on the needs and policies of your enterprise, and are largely the same as your site's existing practices for responding to events that imply data integrity problems. Considerations include tolerance for risk, criticality of the affected databases, uptime requirements, and suspected root cause.

The following represent some considerations and recommendations specific to the WIJ block comparison process:

- Replacing, restoring, or making any changes to the databases or WIJ files after a crash and before recovery can lead to discrepancies that are then found during WIJ comparison and recorded in the MISMATCH.WIJ file. If this has occurred, rename MISMATCH.WIJ.

  Note: If a database is to be restored following a crash, ensure that prior to the restore you start the instance without WIJ and journal recovery (see Starting Caché Without Automatic WIJ and Journal Recovery in the "Backup and Restore" chapter of this guide). This avoids both creating discrepancies that will be detected by the WIJ comparison and incorrectly applying WIJ blocks or journal data (see the "Journaling" chapter of this guide) to a version of a database for which they were not intended.

- Some storage subsystems, particularly local drives on laptops and workstations, use an unsafe form of write-back caching that is not backed by battery or by non-volatile memory. This defeats the two-phase write protocol that Caché performs and can lead to corruption following a hardware crash or power loss that is detected during WIJ compare. If this applies to your system, it is likely that MISMATCH.WIJ contains more up-to-date data and therefore can be safely applied to the databases (assuming the system is one with which an abundance of caution is not required).
- If you have made any changes to the databases following the WIJ comparison, MISMATCH.WIJ is no longer valid and it is not safe to apply the WIJ blocks.
- For servers with enterprise-class storage or any storage subsystem that does not utilize an unsafe form of write-back caching, mismatches found during WIJ compare are always unexpected and warrant careful attention, as they may be a sign of a more serious or more widespread problem.
- Depending on the root cause of the problem, it may be that the databases are intact and it is the WIJ that is corrupted. An integrity check of the affected databases can help determine whether this is likely.
- Because WIJ comparison only covers the blocks written most recently, there may be problems affecting additional blocks that could be detected by a full integrity check (see Verifying Structural Integrity in the "Introduction to Data Integrity" chapter of this guide).
- If the databases are small and/or time allows you can follow a procedure similar to the following for optimal safety:

  1. Run a full integrity check on the databases.
  2. If none are corrupt, rename MISMATCH.WIJ and start up.
  3. If one or more databases are corrupt, copy the CACHE.DAT files for all databases, apply all blocks from MISMATCH.WIJ and run a full integrity check again.
  4. If any database is corrupt after applying MISMATCH.WIJ, the databases can be reverted to the previous copy or restored from a previous backup.

2.3 Limitations of Write Image Journaling

While the two-phase write protocol safeguards structural database integrity, it does not prevent data loss. If the system failure occurs prior to a complete write of an update to the WIJ, Caché does not have all the information it needs to perform a complete update to disk and, therefore, that data is lost. However, data that has been written to a journal file is recovered as described in Recovery in this chapter.

In addition, write image journaling cannot eliminate database degradation in the following cases:

- A hardware malfunction that corrupts memory or storage.
- An operating system malfunction that corrupts memory, the filesystem, or storage.
- The WIJ is deleted.

- The loss of write-back cache contents. In the event of a power outage, the write-back cache could be lost, leading to database degradation. To prevent this degradation, ensure that either the storage array uses nonvolatile memory for its write-back cache or the volatile write-back cache has battery backup.

If you believe that one of these situations has occurred, contact the InterSystems Worldwide Response Center (WRC).

3

Backup and Restore

This chapter outlines the factors to consider when developing a solid plan for backing up your Caché system. It discusses techniques for ensuring the integrity and recoverability of your backups, as well as suggested backup strategies. In addition, the chapter contains details about the procedures used to perform these tasks using Caché utilities or other third-party utilities. It discusses the following topics:

- Backup Integrity and Recoverability
- Importance of Journals
- Backup Strategies
- Restoring from a Backup
- Configuring Caché Online Backup Settings
- Managing Caché Online Backups
- Caché Online Backup Utilities
- Caché Online Backup Restore Utility

Backup strategies can differ depending upon your operating system, preferred backup utilities, disk configurations, and backup devices. If you require more information to help you to develop a backup strategy tailored for your environment, or to review your current backup practices, contact the InterSystems Worldwide Response Center (WRC).

3.1 Backup Integrity and Recoverability

Regardless of the backup strategies you use, it is critical to restore backups on a regular basis to validate your procedures. The best practice — to restore every backup of the production environment to an alternate server and then validate the integrity of the restored databases (see Verifying Structural Integrity in the "Introduction to Data Integrity" chapter of this guide) — provides the following advantages:

- Validates the recoverability of the backup media.
- Validates the physical integrity of the databases in the backup, avoiding the problem of false reporting of integrity errors when running integrity checks on volatile databases affected by ongoing updates.
- Provides a warm copy of the backup, substantially reducing the time required to restore the backup in the event of a disaster. If such an event occurs, you need only restore the updates in the journal files.
- Establishes a last known good backup.

The backup strategies described in this document preserve the physical structure of the database; therefore, a clean integrity check of the restored copy implies that the integrity of the production database was sound at the time of the backup. The converse, however, is not true: an integrity error detected on the restored copy of a database does not necessarily imply that there are integrity problems on the production database; there could, for example, be errors in the backup media. If you discover an integrity error in the restored database, immediately run an integrity check on the production database to verify the integrity of the production system.

To further validate that the application is working correctly on the restored database, you can also perform application-level checks. To perform these checks, you may need to restore journal files to restore transactional integrity. See the Importance of Journals section for more information.

Once you restore the backup and establish that it is a viable source of recovery, it is best to preserve that restored copy until you establish the next good backup. Therefore, the server on which you are validating the backup should ideally have twice the storage space required by production—space to store the last-known good backup as well as the backup you are currently validating. (Depending on your needs, you may have less stringent performance requirements of the storage device used for restoring backups, allowing for a less expensive storage solution.) In this way, the last-known good backup is always available for use in a disaster even if validation of the current backup fails. To protect enterprise databases from a disaster that could destroy the data center, regularly ship backup media to a secure off-site location.

3.2 Importance of Journals

The backup of a Caché database alone is not enough to provide a viable restore of production data. In the event of a failure that requires restoring from backup, you must also apply journal files to the restored copy of the database. Applying journal files restores all journaled updates from the time of the backup to the time of the failure. Also, applying journals is necessary to restore the transactional integrity of your database by rolling back uncommitted transactions.

Be sure to see Journaling Best Practices in the "Journaling" chapter of this guide for important information about ensuring journal availability for recovery. In particular, as explained in that section, in the interests of performance and recoverability, InterSystems recommends placing the primary and alternate journal directories on storage devices that are separate from the devices used by databases and the write image journal (WIJ), as well as separate from each other.

Important: It is critical to periodically test your entire disaster recovery procedure from start to finish. This includes backup restore, journal restore, and running simulated user activity on the restored environment.

3.3 Backup Strategies

The best strategies for backing up Caché data are external backup and Caché online backup. This section describes these and other special-purpose strategies:

- External Backup
- Caché Online Backup
- Cold Backup
- Legacy Concurrent External Backup

3.3.1 External Backup

External backup is currently the recommended best practice for backing up Caché. It integrates easily with your existing system backup procedures and typically allows for a zero downtime backup. It is used primarily in conjunction with technology that provides the ability to quickly create a functional "snapshot" of a logical disk volume. Such technologies exist at various levels, from storage arrays, to operating systems, to simple disk mirroring. There are special considerations, discussed in this section, for systems on which snapshot technology is not available.

To ensure the integrity of the snapshot, Caché provides methods to freeze writes to databases while the snapshot is created. Only physical writes to the database files are frozen during the creation of the snapshot, allowing user processes to continue performing updates in memory uninterrupted. The snapshot is typically a snapshot of all file systems in use by the system. At a minimum, this includes all directories used by Caché in any way, such as the installation directory, database directories, journal and alternate journal directories, WIJ directory, and any directory containing external files used by Caché. After writes are thawed, the snapshot may be backed up to tape and then either rejoined to production or left online as a warm backup (depending on the specific technology used).

Note: To avoid the problem of false reporting of errors when running integrity checks on databases while they are in use, you can integrate the integrity checks into procedures like those described in this section, so that databases are checked immediately after the file system snapshot is taken. (See Verifying Structural Integrity in the "Introduction to Data Integrity" chapter of this guide for information about checking database integrity.)

The following table lists the advantages and disadvantages of the external backup strategy:

Advantages	Disadvantages
Allows zero downtime backups with no user interruption for most systems.	If writes are to be frozen for longer than 10 minutes, special consideration is required if you want to allow users to continue uninterrupted.
Integrates easily with existing backup procedures.	

The class methods that perform the database freeze and thaw operations are **Backup.General.ExternalFreeze()** and **Backup.General.ExternalThaw()**, respectively. In addition to pausing writes, the freeze method also handles switching journal files and writing a backup marker to the journal. The journal file continues to be written normally while physical database writes are frozen. If the system were to crash while the physical database writes are frozen, data would be recovered from the journal as normal during startup.

To summarize, the external backup involves the following steps:

1. Freeze writes to the database using the **Backup.General.ExternalFreeze()** method. Examples of the use of this method on various platforms are included in the class documentation.

   Note: When the security configuration requires that the backup script supply Caché credentials, you can do this by redirecting input from a file containing the needed credentials. Alternatively, you can enable OS-level authentication and create a Caché account for the OS user running the script.

2. Create a snapshot of the file system using an external snapshot utility.
3. Resume Caché writes using the **Backup.General.ExternalThaw()** method.
4. Copy the snapshot to the backup media.

See the Backup.General class documentation in the *InterSystems Class Reference* for platform-specific examples of these methods.

For systems where snapshot technology is not available, a slower filesystem copy may be used in the external backup approach, described above, by replacing the creation of the snapshot with a filesystem copy. This can be done in one of the following ways or, depending on your needs, Caché Online Backup may be an alternative:

- **Zero Downtime**: Specify a value for the *ExternalFreezeTimeOut* parameter when calling **Backup.General.ExternalFreeze()** and ensure that you have configured sufficient database cache to allow database updates to be buffered in memory for the duration of the freeze. In this case the system allows users to continue working for extended periods with physical writes frozen, up to the specified *ExternalFreezeTimeOut*. Journaling is critical to prevent data loss in the event that the system crashes while writes are frozen; in the event of a crash, system startup may take longer than usual. The Journal setting for **Freeze on Error** should be set to **Yes** (see the Journal I/O Errors section in the "Journaling" chapter of this guide for more information).
- **Some User Interruption**: Allow the expected freeze time to default to 10 minutes, after which users are paused until **Backup.General.ExternalThaw()** is called.

Important: When the instance being backed up is the primary failover member in a mirror (see the chapter "Mirroring" in the *Caché High Availability Guide*), an external freeze must not suspend updates for longer than the specified *ExternalFreezeTimeOut* parameter of **Backup.General.ExternalFreeze()**. If this happens, the mirror may fail over to the backup failover member, thereby terminating the backup operation in progress.

When the backup is complete, you can use the **Backup.General.ExternalSetHistory()** method to add a the backup to the backup history. Note that this can trigger a journal purge, depending on the **When to purge journal files > After this many successive backups** setting on the Journal Settings page (the default is 2); for more information, see Configuring Journal Settings in the "Journaling" chapter of this guide.

Note: Caché supports the Volume Shadow Copy Service (VSS) on Windows by acting as a writer on behalf of its databases. Copies of Caché databases included in a VSS shadow are physically consistent, although not logically consistent with respect to transactions, and therefore may be restored individually. To ensure the transactional integrity of these restored databases, journal files should also be restored. Only databases that are mounted at the time of VSS shadow creation are included in the VSS shadow.

The VSS writer for Caché can be started only by an administrator.

On Windows systems, the EnableVSSBackup parameter in the cache.cpf file is set to 1 (enabled) by default. At Caché startup, the message "Caché VSS Writer started" is written to the console log. When you create a VSS shadow copy, Caché automatically calls **Backup.General.ExternalFreeze()** and **Backup.General.ExternalThaw()**, as indicated by messages in the console log.

3.3.2 Caché Online Backup

Caché implements a proprietary backup mechanism designed to cause very minimal or no downtime to users of the production system. Caché online backup, which only backs up data in CACHE.DAT files, captures all blocks in the databases that are allocated for data. The output goes to a sequential file. This must be coordinated with a system backup to copy the Caché online backup output file to the backup media along with other files. Typically the system backup should include all filesystems in use by the system, excluding CACHE.DAT files. At a minimum it must include the installation directory, journal and alternate journal directories, CSP files, and any directory containing external files used by Caché; exclude CACHE.DAT files.

The Caché online backup procedure uses multiple passes to copy data, where each consecutive pass copies an incrementally reduced list of data blocks that changed during the previous pass. Generally, three passes are sufficient to complete a backup. During the entire final pass and for a brief moment during each prior pass, the process pauses writes to the database. The backup pauses physical writes to the database while allowing user processes to continue performing updates in memory.

The following table lists the advantages and disadvantages of the online backup strategy:

Advantages	Disadvantages
Allows zero downtime backups for most systems. With old-format databases, there is minimal user interruption.	Backs up only databases; it does not back up external files.
Supports cumulative and incremental backups on a regular basis.	Restoring a single database requires processing the entire backup file.
Does not require enterprise-class storage.	Requires a running instance of Caché to perform a restore.
	Restores can be cumbersome with extremely large amounts of data.
	Backups of encrypted databases are unencrypted.

There are different types of online backup, which you can combine to manage the trade-offs between the size of the backup output and the time needed to recover from the backup:

- **Full Backup** — Writes an image of all in-use blocks to the backup media.
- **Cumulative Backup** — Writes all blocks that have been modified since the last full backup. Must be used in conjunction with a previous full backup.
- **Incremental Backup** — Writes all blocks that have been modified since the last backup of any type. Must be used in conjunction with a previous full backup and (optionally) subsequent cumulative or incremental backups.

When using online backup, you must first run a full backup, after which you can run cumulative and/or incremental backups.

Online backup writes all database blocks to a single file (or set of tapes) in an interleaved fashion. When you back up an extremely large amount of data using the online backup, restores can become somewhat cumbersome; consider this when planning your backup strategy.

The restore validation process helps resolve any limitations by providing an online, restored copy of the databases. Use the same backup validation strategy when running incremental or cumulative backups. After you perform each incremental or cumulative backup, you can immediately restore to the alternate server. For example, a strategy of weekly full backups and daily incremental backups can work well because each daily backup contains only the blocks modified that day. Using this strategy, you should restore the incremental backup to the alternate server each day, and check the integrity of the restored databases.

Avoid overwriting the warm copy of the last known good backup when restoring the backup you are currently validating. The same concept applies when restoring an incremental to the existing restored database. After you establish that the backup is the last known good backup and before applying the next day's incremental or cumulative backup to it, save a copy so that the last known good backup is always online and ready for use in case the subsequent incremental restore fails. If a restored backup fails an integrity check, you must discard it; you cannot use it as a target of a subsequent incremental restore.

When restoring a system from an online backup, first restore the most recent full backup, followed by the most recent cumulative backup, and then all incremental backups taken since the cumulative backup.

For information about configuring Caché online backup, see Configuring Caché Online Backup Settings in this chapter; for information about managing Caché online backup, see Managing Caché Online Backups in this chapter.

3.3.3 Cold Backup

Cold backup refers to an external backup of the system taken when Caché has been shut down normally. The backup is typically a backup of all filesystems in use by the system. At a minimum, all directories used by Caché in any way, including the installation directory, database directories, journal and alternate journal directories, WIJ directory, and any directory containing external files used by Caché. This strategy may be appropriate for systems where Caché is typically shut down at night.

The following table lists the advantages and disadvantages of the cold backup strategy:

Advantages	Disadvantages
Simple procedure (stop Caché and copy entire instance).	Caché is not available.
Occasionally, it is useful to back up the system while Caché is shut down.	

3.3.4 Legacy Concurrent External Backup

The Legacy Concurrent External Backup strategy is also known as "dirty backup." It was designed before Caché had the capability of performing a zero downtime backup that is available in both External Backup and Caché Online Backup; the design allowed for an external backup to be taken with only a minimum amount of downtime. Today, this backup strategy is practical only in very special cases, typically only if your circumstances include one or more of the following conditions exist:

- Caché is clustered and downtime during backup is required to be minimal. It is only in such a cluster that other backup strategies do not provide a zero downtime backup.
- Snapshot technology is not available. If snapshot technology is available, External Backup can be used without incurring more downtime than Legacy Concurrent External Backup.
- Caché Online Backup does not meet your needs. Caché Online Backup is a simpler strategy and does not incur any more downtime than Legacy Concurrent External Backup.

The Legacy Concurrent External Backup strategy works by allowing an external system backup to run while normal databases writes are taking place. The copy of the databases captured in this manner are expected to be corrupt ("dirty"). This copy is then followed by an Incremental Caché Online Backup, which captures any database blocks that were modified while the dirty copy took place. When restoring the backup, first the copy of the databases are restored, then the incremental backup is restored. The steps in detail are as follows:

1. Clear the list of database blocks modified since the last backup:

   ObjectScript

   ```
   Do CLRINC^DBACK("QUIET")
   ```

   For information, see CLRINC^DBACK in this chapter.

2. Copy the CACHE.DAT database files, using your chosen operating system or third-party backup utility.
3. Call **$$BACKUP^DBACK** with the *E* parameter to indicate you are using an external backup utility; for example:

   ObjectScript

   ```
   Set x=$$BACKUP^DBACK("","E","Dirty external backup - incrementals must be applied.","","","")
   ```

For information, see BACKUP^DBACK in this chapter.

4. Perform a Caché incremental backup, which copies any blocks that changed while the CACHE.DAT files were being copied; this may cause a very brief suspension of user processes in some configurations:

   ObjectScript

   ```
   Set x=$$BACKUP^DBACK("","I","Nightly","test.bck","N","bck.log","QUIET","N","Y")
   ```

   For information, see BACKUP^DBACK in this chapter.

The following table lists the advantages and disadvantages of the legacy concurrent external backup strategy:

Advantages	Disadvantages
Provides a backup strategy with little or no interruption for sites that are not able to use one of the other recommended approaches.	Multiple files need to be restored (CACHE.DAT database files and incremental backup files), which causes the restore process to take longer.
	The procedure is complex.
	Requires a running instance of Caché to perform a restore.

3.4 Restoring from a Backup

It is critical to restore backups on a regular basis to test your procedures. As described in the Backup Integrity and Recoverability section in this chapter, InterSystems recommends restoring every backup onto an alternate server for validation and to serve as a warm copy of the restore.

Backup restores can be complex and time consuming, and the practice of regular restores helps mitigate the risks involved. Backup restore should be considered as part of your larger disaster recovery plan. Caché Shadowing and Mirroring also provide alternate mechanisms for disaster recovery.

For additional help with backup restore scenarios, contact the InterSystems Worldwide Response Center (WRC).

This section includes the following topics:

- Backup Restore Scenarios
- Starting Caché for Maintenance
- Journal Restore Following Backup Restore
- Starting Caché Without Automatic WIJ and Journal Recovery

3.4.1 Backup Restore Scenarios

There are many scenarios where a backup restore is performed, often with unique sets of requirements and restore procedures. This section describes the following general restore scenarios, including variations for different backup strategies:

- Full System Restore — All or most of the storage used by Caché or the system as a whole has been rendered unusable. The full system backup has been restored from the backup media and now Caché must be recovered to a production-ready state.

- Database-Only Restore — Some or all Caché databases have been rendered unusable and must be recovered from backup.
- Restore/Migrate Database to Different Systems — One or more databases are being restored to a different system. The target system may be newer hardware, a different platform, or another instance of Caché on the same hardware. The target system may have a different filesystem layout. This may be used to add databases to a backup or async mirror member (see the "Mirroring" chapter in the *Caché High Availability Guide*) or synchronize with the shadowing destination (see the "Shadowing" chapter in this guide).

Important: To avoid data incompatibility, the following requirements must be met when restoring or migrating a database to a different Caché instance than the one in which the backup was created:

- The target instance must use the same character width (8-bit or Unicode; see Caché Character Width in the *Caché Installation Guide*) and the same locale (see Using the NLS Settings Page of the Management Portal in the "Configuring Caché" chapter of the *Caché System Administration Guide*) as the source instance.

  The one exception to this requirement is that an 8-bit instance using a locale based on the ISO 8859 Latin-1 character set is compatible with a Unicode instance using the corresponding wide character locale. For example, a database created in an 8-bit instance using the **enu8** locale can be used in a Unicode instance using the **enuw** locale.
- If the source and target instances are on systems of different endianness (see "Platform Endianness" in the "Supported Technologies" section of the online *InterSystems Supported Platforms* document for this release), the database must be converted to the endianness of the target instance before being used; see Considering Endianness for procedures.

Note: If you have auto-start configured, you may want to prevent the instance from starting automatically when the host OS restarts. This can only be done at the OS level and is dependent on your configuration.

3.4.1.1 Full System Restore

This scenario assumes the following starting point:

- The system where the restore is being performed is the same as, or identical to, the system that generated the backup.
- All filesystems (or at least all files backed up) have been restored from the backup media and the path names at the operating system level are unchanged.
- Caché is down and has not been started since the restore.

Full System Restore from External Backup or Cold Backup

If you used the External Backup or Cold Backup procedures described earlier in this chapter, your databases have been restored to a physically consistent state, but require at least transaction rollback to restore transactional consistently. You may also have journal files from after the time of the backup that you wish to apply before rolling back transactions. This section provides two recommended procedures, one of which you should use (depending on the details of the backup and restore you performed) for the required journal restore and transaction rollback.

If the image of the system that was restored is a snapshot of all elements of Caché, including the CACHE.WIJ file and journal files, from a single moment in time—that is, a *crash-consistent image*—and you have no newer journal files to apply, you can simply start Caché normally; Caché automatically performs journal recovery and transaction rollback. This is also true if you start Caché in maintenance mode, as described in Starting Caché for Maintenance.

Note: The preceding recommendation applies after a crash-consistent snapshot of the entire system is restored even if normal external backup procedures (as described in External Backup and Cold Backup) were not used. In that case, the database files in the restored image may not be physically consistent, but are recovered automatically from the CACHE.WIJ file before automatic journal recovery and transaction rollback. For details on recovery, see Fundamental Data Integrity Protection and the sections that follow in the "Introduction to Data Integrity" chapter of this guide.

If the restored image does not includes all elements of Caché—that is, the CACHE.WIJ file, the journal files, and other Caché files—from a single moment in time, or if you have newer journal files to apply, you must use the following procedure to perform journal restore manually, which includes transaction rollback.

Important: This procedure assumes that the CACHE.WIJ file was backed up as part of the backup snapshot and restored along with the databases and other files and file systems, which is the recommended way to plan your backup and restore procedures. If the CACHE.WIJ file is not included in your backup, see the Starting Caché Without Automatic WIJ and Journal Recovery section for information about how to start Caché.

1. Start Caché in maintenance mode as described in Starting Caché for Maintenance to perform a journal restore before users are allowed to access the system. If the instance is already running, you should stop it first and then start it again in maintenance mode.

   Note: If the system is down and you have auto-start configured, you may want to prevent the instance from starting automatically, so that you can start the instance in maintenance mode directly. This can only be done at the OS level and is dependent on your configuration.

2. Restore the journals as described in Journal Restore Following Backup Restore.
3. After the journal restore is complete, you can perform application validation or additional maintenance as needed.
4. Restart Caché normally to allow users to access the system.

Full System Restore from Caché Online Backup or Legacy Concurrent External Backup

The remainder of this section discusses the full system restore for users of Caché Online Backup or Legacy Concurrent External Backup. You must perform a Caché online backup restore to restore the databases from the backup output file. For Caché online backup, the backup output file is written to the backup media as part of the full system backup, as described in the Caché Online Backup section in this chapter.

Caché online backup restore can be performed only on a running instance of Caché. Therefore, in this full system restore scenario, you must have a separate, secondary instance of Caché installed to run the restore of the primary instance's databases. The databases being restored do not need database or namespace definitions in the secondary instance.

The secondary instance can be installed as part of this restore procedure by running the Caché installer and selecting an instance name and installation directory that is different from the primary instance. Alternatively, to avoid this step at restore time install the secondary "restore" instance when you decide to use the Caché online backup strategy. This secondary instance should be shut down at all times and its installation directory backed up as part of the full system backup (that is, it would be a Cold Backup for the secondary instance). It could simply be started during a full system restore in order to run the Caché online backup restore of the primary instance.

To restore from a Caché online backup or legacy concurrent external backup after a full system restore:

1. Using the secondary instance of Caché, follow the instructions in the Caché Online Backup Restore Utility section in this chapter to:

   a. Restore the last full backup (if using Legacy Concurrent External Backup, the full backup was already restored externally during the filesystem restore).

   b. If you have done cumulative backups since the full backup, restore the last one; otherwise, continue with the next step.

c. Restore all incremental backups done since the last cumulative backup (if there is one), or the last full backup (if there is no cumulative), in the order in which the backups were run.

2. Using the secondary instance of Caché, follow the instructions in the Journal Restore Following Backup Restore section in this chapter.
3. Shut down the secondary instance of Caché.
4. **IMPORTANT:** Start the primary instance as described in the Caché Without Automatic WIJ and Journal Recovery section in this chapter. This step leaves Caché running in maintenance mode, which lets you perform application validation or additional maintenance.
5. Restart Caché normally to allow users to access the system.

3.4.1.2 Database-Only Restore

This scenario assumes the following starting point:

- The system where the restore is being performed is the system that generated the backup and the databases are to be restored to their original location.
- You have identified the set of databases that require restoration.
- Caché may be up and users may be accessing other databases normally; the databases requiring restoration may not be required for startup.
- Caché may be unable to start fully; the databases requiring restoration may be required for startup, or damaged in a way that is preventing startup from completing.

Note: If you used External Backup or Cold Backup and you have many database that need to be restored, a simpler procedure may be to shut down Caché, restore ALL CACHE.DAT files, and perform the restore procedures described in the Full System Restore section in this chapter. This is especially true if the CACHESYS database must be restored.

To perform a database-only restore:

1. If Caché is not started already, start it as described in the Starting Caché for Maintenance section in this chapter. Starting Caché ensures that any pending automatic recovery to existing databases occurs immediately and does not conflict with the databases that you are about to restore on subsequent startup.

   If Caché is already started and you are restoring databases that users may attempt to access during the restore process, InterSystems recommends that you shut down and restart as described in the Starting Caché for Maintenance.

   The following table describes recommended procedures to resolve the problems that prevent Caché from starting.

Problem	Solution
1. Caché fails to start	Starting Caché for maintenance allows Caché to start even if a database that is required for startup cannot be mounted. However, if there is automatic journal recovery pending for a database that is marked as required for startup, Caché does not skip it. Therefore, if damaged databases are still preventing Caché from starting: a. Rename the CACHE.DAT files to make the database completely inaccessible. b. Change the configuration to mark those databases as not required for startup, c. Try again to start Caché as described in the Starting Caché for Maintenance.

Problem	Solution
2. If Caché cannot start due to a damaged CACHESYS database	Temporarily replace the CACHESYS database with a copy from an External Backup, or from a new install of the same version of Caché. This is temporary solution to get Caché started to complete the restore procedure.
3. Caché still does not start	You may need to restore all databases following the procedure in Full System Restore. For additional help, contact the InterSystems Worldwide Response Center (WRC).

2. Depending on the strategy you used to back up the database, restore each database as follows:

Backup Strategy Used	Database Restoration Sub-procedure
External Backup or Cold Backup	For all databases except CACHESYS: a. Dismount the database from Caché. b. Copy the CACHE.DAT files from the backup media to their original locations. c. Remount the database. If you are restoring the CACHESYS database, restore it to an alternate directory temporarily.
Caché Online Backup	Follow the instructions in the Caché Online Backup Restore Utility section in this chapter to: a. Restore the last full backup. b. If you have done cumulative backups since the full backup, restore the last one; otherwise, continue with the next step. c. Restore all incremental backups done since the last cumulative backup (if there is one), or the last full backup (if there is no cumulative), in the order in which the backups were run.
Legacy Concurrent External Backup	Perform the same steps as documented for External Backup (in this table) to restore the "dirty" copy of the CACHE.DAT files, then apply the incremental backup as documented for Caché Online Backup (in this table).

3. Follow the instructions in the Journal Restore Following Backup Restore section in this chapter.
4. If you started Caché as described in the Starting Caché for Maintenance section in this chapter, and you are ready to allow users on the system, stop and restart Caché normally. You can perform application validation or additional maintenance before restarting.

Note: If you restored the CACHESYS database from External Backup or Cold Backup to a temporary alternate directory as described in the table in step 2 of this procedure, you must replace the existing CACHESYS database with the CACHE.DAT file from the temporary directory after shutting down (*before restarting*) Caché.

3.4.1.3 Restore/Migrate Database to Different Systems

This scenario assumes the following starting point:

- One or more databases are being restored to a target system that is different than the system that generated the backup.
- The target system may be newer hardware, a different platform, or simply a different instance of Caché on the same hardware.
- The target system may have a different filesystem layout than the source.
- Caché is already installed on the target system (you cannot copy a Caché installation or its CACHESYS database to a target machine that is not identical to the source).

The purpose of this type of restore includes, but is not limited to:

- Hardware migration
- Adding databases to a backup or async mirror member (see the "Mirroring" chapter in the *Caché High Availability Guide*) or synchronizing with the shadowing destination (see the "Shadowing" chapter in this guide)
- Copying databases to another system for development, testing, or deployment

To restore/migrate databases to a different system:

Important: Do not attempt to replace the CACHESYS database on the destination system with a restore of the CACHESYS database from the source unless it is part of a full system restore. Any data needed from the source CACHESYS database can be exported and then imported into the destination CACHESYS database.

If the source and target systems use a different endianness, see the "Considering Endianness" subsection following this procedure. If the source or target systems use database encryption, see the "Considering Database Encryption" subsection following this procedure.

1. Start Caché if not started already. Starting Caché ensures that any pending automatic recovery to existing databases occurs immediately and does not conflict with the databases that you are about to restore.

   If you are restoring databases that users may attempt to access during the restore process, you should start as described in the as described in the Starting Caché for Maintenance section in this chapter.

2. Depending on the strategy you used to back up the database, restore each database as follows:

Backup Strategy Used	Database Restoration Sub-procedure
External Backup or Cold Backup	To restore databases: a. Dismount any databases being restored if they are mounted on the target instance. b. Copy the CACHE.DAT files from the backup media to the desired locations. c. Add any new databases to the configuration. d. Remount the database that you dismounted.

Backup Strategy Used	Database Restoration Sub-procedure
Caché Online Backup	Follow the instructions in the Caché Online Backup Restore Utility section in this chapter to: a. Restore the last full backup. b. If you have done cumulative backups since the full backup, restore the last one; otherwise, continue with the next step. c. Restore all incremental backups done since the last cumulative backup (if there is one), or the last full backup (if there is no cumulative), in the order in which the backups were run.
Legacy Concurrent External Backup	Perform the same steps as documented for External Backup (in this table) to restore the "dirty" copy of the CACHE.DAT files, then apply the incremental backup as documented for Caché Online Backup (in this table).

3. Follow the instructions in the Journal Restore Following Backup Restore section in this chapter.

   Note: If you are restoring from Caché Online Backup to add databases to a backup or async mirror member (see the "Mirroring" chapter in the *Caché High Availability Guide*) or synchronize with the shadowing destination (see the "Shadowing" chapter in this guide), skip this step; journals are applied automatically in these cases.

4. If you started Caché as described in the Starting Caché for Maintenance section in this chapter, and you are ready to allow users on the system, stop and restart Caché normally.

Considering Endianness

Depending on the strategy you used to back up the database, an extra step is required to restore a backup on a target system whose Endianness (big-endian vs. little-endian) is different from the source system. For information about the Endianness of supported platforms, see "Platform Endianness" in the "Supported Technologies" section of the online *InterSystems Supported Platforms* document for this release.

- For External Backup or Cold Backup — convert a copied CACHE.DAT file as described in the Using cvendian to Convert Between Big-endian and Little-endian Systems section of the "Migration and Conversion Utilities" chapter of *Caché Specialized System Tools and Utilities*.
- For Caché Online Backup or Legacy Concurrent External Backup — You cannot restore a CACHE.DAT on a target system whose endianness is different from the source system. Instead, do the following:

  1. Use the procedure outlined in this section to restore on a system whose endianness is the same as the source. This restore must include any required journal restore (see step 3 in the procedure).
  2. Shut down Caché or dismount the databases.
  3. Convert the database as described in the Using cvendian to Convert Between Big-endian and Little-endian Systems section of the "Migration and Conversion Utilities" chapter of *Caché Specialized System Tools and Utilities*.
  4. Copy the CACHE.DAT files to the target system.
  5. Use the procedure outlined in this section to restore on the target system with the converted databases, as described here for External Backup.

Considering Database Encryption

When you restore a database on a target system, it may be necessary to change the database encryption key for the restored database. Depending on the strategy you used to back up the database, you may have to convert the key on the restored database, as follows:

- For Caché Online Backup — Caché Online Backup stores the database contents unencrypted. When restoring a Caché online backup file to an existing encrypted database on a target system, the Caché online backup restore utility encrypts the file dynamically on the target system; otherwise the restored database is unencrypted.
- For External Backup, Cold Backup or Legacy Concurrent External Backup — Depending on state of the CACHE.DAT file on the source system, use the cvencrypt utility to manage database encryption after restoring it on the target system; for example, if the database files on the source and target systems use different encryption keys, you must convert the restored database to use the encryption key on the target system. For more information about the cvencrypt utility, see the "Using the cvencrypt Utility" appendix of the *Caché Security Administration Guide*.

3.4.2 Starting Caché for Maintenance

During any restore procedure it is critical that no application activity occurs in the databases being restored until you are completely finished with the backup restore and journal restore, and you have completed any desired application-level validation. If you have, in your application or operating environment, a mechanism to ensure that no user activity can occur when Caché is started you can use that mechanism; for example, if your application is entirely web-based, shutting down the web servers may be sufficient to ensure that only administrators involved in the restore procedure have access to the system.

Caché provides a mechanism to let you start Caché with limited access while maintenance is performed.

CAUTION: The following procedure is modified from earlier versions to utilize enhancements to emergency access mode introduced with version 2011.1; if you are using an earlier version of Caché, see the version 2010.2 documentation.

To start Caché for maintenance:

1. Start Caché in emergency access mode, specifying a unique username and password to be used for all maintenance activity; this username and password should not match any existing user:
   - On Windows, invoke the following command as administrator from the bin directory of your Caché installation:

     ```
     ccontrol start <cache-instance-name> /EmergencyId=<username>,<password>
     ```

   - On UNIX®/Linux and macOS platforms, invoke the following command:

     ```
     ccontrol start <cache-instance-name> EmergencyId=<username>,<password>
     ```

   For more information, see the Emergency Access section of the "System Management and Security" chapter of the *Caché Security Management Guide*.
2. Perform the remaining portions of the restore procedure as documented for your restore scenario.
3. After completing the restore procedure — including any journal restore, application validation, or other maintenance activities — make the system available to users as follows:
   a. Shut down Caché.
   b. Start Caché without the EmergencyId switch.

3.4.3 Journal Restore Following Backup Restore

It is necessary to apply journal files following backup restore. As described in the Importance of Journals section of this chapter, you usually have journal files available at restore time that are newer than the backup. Applying these journal files restores all journaled updates from the time of the backup to the time of the disaster.

Additionally, applying journals is necessary to restore the transactional integrity of your restored database (which may have partial transactions) by rolling back uncommitted transactions. This is required even if you only have journal files up to the moment of the backup and not newer.

Important: If you do not have journal files at least up to the moment of the backup, do not apply any journal files.

When you start the restore, you must select a start point for journal restore. The journal restore may then need to look backwards in the journal stream to find the beginning of a transaction that began prior to the backup. When you use Caché Online Backup, the backup stores a suggested start point; when you back up using the ^DBACK or ^BACKUP utilities, the utility displays information about the oldest journal file required for transaction rollback during restore at the beginning of the third and final pass.

For information about how to apply journals following the backup restore, see Restore All Databases Using ^DBREST and the Restore Journal Files section in the "Journaling" chapter of this guide.

3.4.4 Starting Caché Without Automatic WIJ and Journal Recovery

Some of the restore procedures described in this chapter require starting Caché in a way that prevents the automatic recovery of databases at startup from occurring. This is specified when all of the recovery is to be done manually in the restore procedure and the automatic startup recovery may be incompatible with the manual recovery effort.

CAUTION: The restore procedures documented in this chapter explicitly state when the following procedure is necessary. Performing the procedure at any other time is extremely dangerous and could damage database integrity. In addition, it is modified from earlier versions to utilize enhancements to emergency access mode introduced with version 2011.1; if you are using an earlier version of Caché, see the documentation for that specific version.

To start Caché without automatic WIJ and Journal Recovery:

1. Remove but do not delete the CACHE.WIJ file, so that it remains available for analysis, as follows:
   a. Check the startup.last file in the manager directory for the line `wijlast=`.
   b. Move or rename the CACHE.WIJ file in the directory specified in that line. If a null directory is specified, the CACHE.WIJ file is located in the manager directory.

2. Start Caché in emergency access mode, specifying a unique username and password to be used for all maintenance activity; this username and password should not match any existing user:
   - On Windows, invoke the command from the bin directory of your Caché installation:

     ```
     ccontrol start <cache-instance-name> /EmergencyId=<username>,<password>
     ```

   - On UNIX®/Linux and macOS platforms, invoke the command:

     ```
     ccontrol start <cache-instance-name> EmergencyId=<username>,<password>
     ```

   For more information, see the Emergency Access section in the "System Management and Security" chapter of the *Caché Security Management Guide*.

3. In most cases, Caché does not start completely due to the missing journal recovery information in the WIJ. If Caché starts normally, skip the remainder of this procedure.
4. Follow the instructions in the cconsole.log to enter Caché with the `-B` option and run **do ^STURECOV**:
   a. Select option 8 to reset Caché to start without journal recovery.
   b. Select option 9 to shut down Caché (using the displayed instructions).
5. Start Caché in emergency access mode again, as described in step 2.
6. Perform the remaining portions of the restore procedure as documented for your restore scenario.
7. After completing the restore procedure — including any journal restore, application validation, or other maintenance activities — you can make the system available to users as follows:
   a. Shut down Caché.
   b. Start Caché without the EmergencyId (/EmergencyId on Windows platforms) switch.

3.5 Configuring Caché Online Backup Settings

Caché online backup is one of the backup strategies that you can implement. For information about the supported strategies and the best practices for using them, see the Backup Strategies section in this chapter.

You can configure the Caché online backup settings using the choices in the **System Administration > Configuration > Database Backup** menu of the Management Portal. You can also perform some of these tasks using the methods in the Backup.General class. This section points you to the appropriate method where it applies.

When a Caché online backup is complete, your description of the backup is written to the backup history (see View Backup History). Note that this can trigger a journal purge, depending on the **When to purge journal files > After this many successive backups** setting on the Journal Settings page (the default is 2); for more information, see Configuring Journal Settings in the "Journaling" chapter of this guide.

From the System Management portal you can perform the following configuration tasks:

- Define Database Backup List
- Configure Backup Tasks
- Schedule Backup Tasks

Note: The configuration tasks are in the management portion of the Management Portal application; you must have the appropriate manager-level security to configure the backup settings.

3.5.1 Define Database Backup List

Caché maintains a database list that specifies the databases to back up. You can display this list by navigating to the **System Administration** > **Configuration** > **Database Backup** > **Database Backup List** page of the Management Portal.

Use the arrow buttons to move the databases you *do not* want to back up to the **Available** list and the databases you *do* want to back up to the **Selected** list. Click **Save**. If you do not select any databases, the system backs up all databases.

When you add a new database to your system, the system automatically adds it to the database list. If you do not need to include the new database in your backup plan, be sure to remove it from the **Backup Database List**.

Note: The defined database list is ignored by the *FullAllDatabases* backup task, which performs a backup of all databases excluding the CACHE, CACHETEMP, CACHELIB, DOCBOOK, and SAMPLES databases.

You can also maintain the backup database list using the **^BACKUP** routine or the **Backup.General.AddDatabaseToList()** and **Backup.General.RemoveDatabaseFromList()** methods. See the Backup.General class description in the *InterSystems Class Reference* for details on using these methods.

3.5.2 Configure Backup Tasks

Caché provides different types of backup tasks; each is listed as an item on the **Database Backup Settings** menu. The configuration backup tasks are:

- **Configure Full Backup of All Databases**
- **Configure Full Backup of the Database List**
- **Configure Incremental Backup of the Database List**
- **Configure Cumulative Backup of the Database List**

These are predefined backup tasks that you can run on-demand from the **System Operation** > **Backup** page of the portal. You can also schedule combinations of these backup tasks using the Task Manager. See the Schedule Backup Tasks section in this chapter for details.

The process for configuring each of these tasks is the same. The **Name**, **Description**, and **Type** fields are read-only and reflect the menu choice as described in the following table.

Table 3–1: Backup Task Descriptions

Name	Description	Type
FullAllDatabases	Full backup of all commonly updated databases, whether or not they are in the **Backup Database List**.	Full
FullDBList	Full backup of the Caché databases listed in the **Backup Database List**.	Full
IncrementalDBList	Incremental backup of changes made to the data since the last backup, whether full or cumulative. Backup is performed on the databases currently listed in the **Backup Database List**.	Incremental
CumuIncrDBList	Cumulative and Incremental backup of all changes made to the data since the last full backup. Backup is performed on the databases currently listed in the **Backup Database List**.	Cumulative

You can send backup output to a directory on disk or to magnetic tape. Select one of the two options:

1. To back up to a directory on disk, specify the file pathname in the **Device** field. Click **Browse** to select a directory.
2. To back up to magnetic tape, select the **Save to Tape** check box, and specify a **Tape Number** from the list of available tape device numbers.

   See the Identifying Devices section of the *Caché I/O Device Guide* for detailed information regarding tape numbers.

   Note: InterSystems does not recommend backing up to tape.

The Define Database Backup List section describes how to maintain the **Backup Database List**.

3.5.2.1 Backup File Names

By default, backup files are stored in install-dir\Mgr\Backup. The backup log files are stored in the same directory. Backup files have the suffix .cbk. Backup log files have the suffix .log.

Backup files and backup log files use the same naming conventions:

- The name of the backup task, followed by an underscore character (_)
- The date of the backup, in *yyyymmdd* format, followed by an underscore character (_)
- An incremental number, *nnn*, for that task, for that day
- The .log or .cbk suffix

Where *nnn* is a sequence number incremented for that backup task on that date.

3.5.3 Schedule Backup Tasks

You should ideally set up a schedule for running backups. Backups are best run at a time when there are the least amount of active users on the system.

In addition to the four on-demand backup tasks supplied with Caché, you can create additional definitions of these four backup tasks. For example, you could create two full backup tasks, one to save the backup to a disk file, the other to save the backup to a tape. Or, to alternate backups between two disk drives, you could create a backup task for each drive.

Use the Caché Task Manager to schedule these backup tasks:

1. Navigate to the **Task Scheduler** wizard in the Management Portal (**System Operation** > **Task Manager** > **New Task**).
2. Specify the **Task name** > **Description** > **Task type**, and any changes to the defaults as described in New Task in the "Managing Caché" chapter of the *Caché System Administration Guide* and click **Next**.
3. Enter your desired scheduling information and click **Finish**.

You can now view your task on the **View Task Schedule** page (**System Operation** > **Task Manager** > **View Task Schedule**) and manage it as you do other tasks by clicking **Details** on its row.

3.6 Managing Caché Online Backups

Caché online backup is one of the backup strategies that you can implement. For information about the supported strategies and the best practices for using them, see the Backup Strategies section in this chapter.

You can manage Caché online backups from both the Management Portal and the Caché **^BACKUP** utility. This section focuses on the portal procedures and, where applicable, points to the section that describes the utility procedure.

You can run Caché database backup tasks and view backup history from the **Backup** page (**System Operation** > **Backup**) of the Management Portal. If you schedule additional backup tasks using the Task Manager, you can manage those from the **Task Schedule** page (**System Operation** > **Task Manager** > **Task Schedule**) of the Management Portal.

You can perform the following backup tasks from the System Management portal:

- Run Backup Tasks
- View Backup Status
- Abort a Running Backup
- View Backup History

When you add a new database to your system, you must perform a full backup. You cannot perform an incremental backup until a full backup exists. If you intend to use the incremental and cumulative backup options, you must maintain the backup database list and run a full backup on that list.

After installing Caché, InterSystems recommends that you perform a *FullDBList* backup to establish a complete backup for subsequent use by the other backup tasks.

Note: These backup tasks are in the operations portion of the Management Portal application; you must have the appropriate operator-level security to run the backup tasks.

3.6.1 Run Backup Tasks

There are four types of backup tasks you can run from the Management Portal **Backup** menu option (**System Operation** > **Backup**), each having its own menu item and page:

- **Run Full Backup of All Databases**
- **Run Full Backup of the Backup Database List**
- **Run Incremental Backup of the Backup Database List**
- **Run Cumulative Backup of the Backup Database List**

You must have performed a *full* backup on a database before performing an *incremental* or *cumulative* backup on that database. Use the following procedure to run an online backup:

1. Click the menu item of the appropriate backup type.
2. Verify that the settings in the **Run Backup Task** box are correct.
3. If the backup options are correct, click **OK** to start the backup.

   Otherwise, follow the procedures in the Configuring Caché Online Backup Settings section to update them.
4. While the backup is running, you can view the status of the running backup by clicking the text next to **Backup started**. See View Backup Status for details.

You can also run the last three of these backup tasks by choosing option `1) Backup` from the **^BACKUP** utility or running the **^DBACK** utility. See Back Up Databases Using ^DBACK for details.

Multivolume Backups

A backup may require multiple tape volumes, or multiple disk files. Currently, there is no way to perform a multivolume backup using the Management Portal. If you require a multivolume backup, use the **^BACKUP** utility. If a disk-full condition occurs, **^BACKUP** prompts you for the name of another disk file on another disk.

In the event of an error during backup, you cannot restart the backup on a second or subsequent volume. You must restart the backup from the beginning.

See Estimate Backup Size Using ^DBSIZE for instructions on estimating the size of your backup before you run these tasks.

3.6.2 View Backup Status

Click **View Backup Status** from the **Backup** page (**System Operation** > **Backup**) or click **View** on the running backup process to monitor the progress of the backup operation. The same information is recorded in the log file for that backup operation, which you can later view from the **View Backup History** page.

When Caché begins a backup, it updates the **Time** and **Status** columns of the listing. The **Time** column records the date and time you initiate the backup, not when it completes. The **Status** column is updated to `Running`.

Upon completion of a backup, Caché again updates the **Status** column to indicate the final status of the backup. For example:

- `Completed` indicates the backup successfully completed.
- `Failed` indicates the backup could not be performed or was aborted by the operator.
- `Warning` indicates problems occurred during the backup, but it completed.

The associated backup log contains helpful information; click **View** next to the task to view the backup log.

The next section, *Abort a Running Backup* describes an additional option that appears for a backup that is running.

3.6.3 Abort a Running Backup

There are three ways for you to abort a running backup:

- Navigate to the **View Backup Status** page, click **Abort** next to the running task and then click **OK** to verify your request to abort the backup.
- Call the methods of the Backup.General class; see the **Backup.General.GetAbortStatus()**, **Backup.General.ClearAbortStatus()**, and **Backup.General.AbortBackup()** method descriptions in the *InterSystems Class Reference* for details.
- Choose the abort option in the **^BACKUP** routine; see Abort a Running Backup Using ^BACKUP for details.

3.6.4 View Backup History

Every backup operation creates a separate backup log file. The logs follow the naming convention described in Backup File Names.

From the portal you can view a list of system backup logs from completed backup tasks:

1. Navigate to the **Backup** page (**System Operation** > **Backup**) of the Management Portal.
2. Click **View Backup History** in the right-hand column to display the **Backup History** page (**System Operation** > **Backup** > **Backup History**), which lists information about the completed backups.
3. To view the results of a particular backup, click **View** in the right-hand column of the appropriate row. You can view the contents of a backup log file and search for a string within that file.

Backup Errors

If a backup encounters any I/O errors, the backup aborts and logs a system error in the backup log file. Information in these log files allows you to quickly see where the problem occurred so that you can fix it.

One cause of backup failure is trying to perform a backup on a dismounted database. In the event of an error during backup, the backup utility allows you to retry the device on which the error occurred. Alternatively, you can abort the backup.

3.7 Caché Online Backup Utilities

Caché provides utilities to perform backup tasks. The **^BACKUP** routine provides menu choices to run common backup procedures, which you can also run independently and in some cases non-interactively using scripts.

The utility names are case-sensitive. Run all these utilities from the *%SYS* namespace:

- Estimate Backup Size Using ^DBSIZE

- Perform Backup and Restore Tasks Using ^BACKUP
- Backup Databases Using ^DBACK
- Edit/Display List of Directories for Backup Using ^BACKUP
- Abort a Running Backup Using ^BACKUP
- Display Information About a Backup Volume Using ^BACKUP
- Monitor Backup or Restore Progress Using ^BACKUP

Caché maintains a bitmap list of database blocks modified since the last backup; you can use this list to keep track of database updates during backup stages. The **^DBSIZE** utility, which inspects these bitmaps, helps you calculate the size of a backup. The Caché **^BACKUP** utility uses a multipass scan of these lists to back up only the blocks modified since the last pass.

3.7.1 Estimate Backup Size Using ^DBSIZE

Immediately before performing a Caché backup, you can estimate the size of its output using the **^DBSIZE** utility. It is only an estimate, since there is no way of knowing how many blocks will be modified once the backup starts. You can obtain a more accurate estimate by preventing global updates while running **^DBSIZE**, and then running your backup before resuming global updates.

You can estimate the size of backups in two ways:

- Run ^DBSIZE interactively
- Call ^DBSIZE from a routine

Note: A database must be in the database backup list before you can evaluate it with **^DBSIZE**. See the *Define Database Backup List* section for details.

3.7.1.1 Run ^DBSIZE Interactively

The following procedure describes the steps necessary to run **^DBSIZE** interactively:

1. Open the Terminal and run the utility from the %SYS namespace:

```
%SYS>Do ^DBSIZE
```

2. Caché displays the **^DBSIZE** main menu:

```
               Incremental Backup Size Estimator

What kind of backup:
   1. Full backup of all in-use blocks
   2. Incremental since last backup
   3. Cumulative incremental since last full backup
   4. Exit the backup program
1 =>
```

3. Enter the appropriate menu option to select the type of backup for which you want an estimate: full, incremental, or cumulative. The prompts are the same for each type.
4. At the following prompt either press **Enter** to suspend updates so that you get a more accurate estimate or enter **No** to allow updates:

```
Suspend Updates?  Yes=>
```

If you choose to suspend updates, you receive the following message:

```
WARNING: Switch is set and may affect production for up to 30 seconds.
Waiting for disk cleanup to finish... ready.
```

5. The output first shows you how many Caché blocks you need to do the type of backup you selected. It shows an estimate for each directory in the backup list and a total. The following is an example of the display for a full backup:

```
                                                    In-Use   Block
     Directory                                      Blocks    Size

c:\mycache\mgr\                                     19,963   (8KB)
c:\mycache\mgr\cacheaudit\                             189   (8KB)
c:\mycache\mgr\docbook\                             16,015   (8KB)
c:\mycache\mgr\user\                                    61   (8KB)
                                                --------------
     Total number of database blocks:               36,228
```

The display for an incremental and cumulative backup looks similar to this:

```
                                                  Modified  Block
     Directory                                     Blocks    Size

c:\mycache\mgr\                                          5   (8KB)
c:\mycache\mgr\cacheaudit\                               3   (8KB)
c:\mycache\mgr\docbook\                                  1   (8KB)
c:\mycache\mgr\user\                                     1   (8KB)
                                                --------------
     Total number of database blocks:                   10
```

6. The utility then displays information about backup disk file space:

```
Total backup size, including overhead of volume and pass labels:

     For a disk file:
          Number of 512-byte blocks:  598,092  (306,223,104 bytes)
```

7. Finally, the utility provides information about the amount of space used if you send the backup to magnetic tape:

```
     For magnetic media:
          Number of 58KB blocks:  4  (237,568 bytes)
```

3.7.1.2 Use the ^DBSIZE Function

You can also call **^DBSIZE** from a routine. To do so, you use the following function:

```
$$INT^DBSIZE(backup_type)
```

Important: The values of *backup_type* differ from the menu option numbers when running **^DBSIZE** interactively.

backup_type	Description
1	Incremental backup
2	Full backup
3	Cumulative incremental backup

For example, to see the size estimate for running a full backup:

```
%SYS>w $$INT^DBSIZE(2)
18950^5^160952320^2710^160309248^313104^313104
```

The returned value is seven numbers separated by a caret (^). In this example, the returned value `18950` is the total estimated size of the backup, in blocks; the returned value `5` indicates the number of database directories to back up. The following table shows what is contained in each piece of the output.

Position of output	Description
Piece 1	Number of database blocks for backup (`-1` if error during processing or invalid parameter)
Piece 2	Number of directories to back up
Piece 3	Number of bytes for magnetic media (not including inter-record gaps)
Piece 4	Number of blocks for magnetic media
Piece 5	Number of bytes for a disk file
Piece 6	Number of 512-byte blocks for a disk file

The following shows the output of calling the **^DBSIZE** function for each type of backup:

```
%SYS>w $$INT^DBSIZE(1)
466^5^4157440^70^4122624^8052^8052

%SYS>w $$INT^DBSIZE(2)
18950^5^160952320^2710^160309248^313104^313104

%SYS>w $$INT^DBSIZE(3)
466^5^4157440^70^4122624^8052^8052
```

3.7.2 Perform Backup and Restore Tasks Using ^BACKUP

The Caché **^BACKUP** utility allows you to perform Caché backup and restore tasks from a central menu as shown in the following example:

```
%SYS>Do ^BACKUP


1) Backup
2) Restore ALL
3) Restore Selected or Renamed Directories
4) Edit/Display List of Directories for Backups
5) Abort Backup

6) Display Backup volume information
7) Monitor progress of backup or restore

Option?
```

Enter the appropriate menu option number to start the corresponding routine. Press **Enter** without entering an option number to exit the utility.

Subsequent sections in this document describe the utilities started by choosing the corresponding option:

1. Back Up Databases Using ^DBACK
2. Restore All Databases Using ^DBREST
3. Restore Selected Databases Using ^DBREST
4. Edit/Display List of Directories for Backup Using ^BACKUP
5. Abort a Backup Using ^BACKUP
6. Display Information About a Backup Volume Using ^BACKUP

7. Monitor Backup or Restore Progress Using ^BACKUP

3.7.3 Back Up Databases Using ^DBACK

The Caché **^DBACK** utility allows you to backup Caché databases using Caché online backup. Running the **^DBACK** utility is equivalent to choosing option `1` from the **^BACKUP** utility menu. The following is an example of running the utility from the Terminal:

1. First, make sure you have defined your database backup list and choose your output device for the backup file.
2. From the %SYS namespace run the **^DBACK** routine:

```
%SYS>Do ^DBACK

                Cache Backup Utility
              --------------------------
What kind of backup:
   1. Full backup of all in-use blocks
   2. Incremental since last backup
   3. Cumulative incremental since last full backup
   4. Exit the backup program
```

3. Select which Caché backup type (full, incremental, or cumulative) to run by entering the appropriate menu option number. The procedure for running the backup is the same regardless of what type you run.
4. Enter the output device and a description of the backup:

```
Specify output device (type STOP to exit)
Device: c:\mycache\mgr\backup\FullDBList_20081201_003.cbk =>

Backing up to device: c:\mycache\mgr\backup\FullDBList_20081201_003.cbk
Description: Full Backup using ^DBACK
```

   The device defaults to that of the last backup. This example uses the format of the online backup device.

5. The utility then displays a list of the database directories to back up (those in your database backup list). For example:

```
 Backing up the following directories:
 c:\mycache\mgr\
 c:\mycache\mgr\cacheaudit\
 c:\mycache\mgr\docbook\
 c:\mycache\mgr\user\
```

6. Answer `Y` to start the backup. The journal file switches and you see a display of the backup progress:

```
Start the Backup =>  [answer Y or N] => y
Journal file switched to:
c:\mycache\mgr\journal\20081201.004
.
Starting backup pass 1
```

7. At the beginning of the third and final pass, the utility displays information about the oldest journal file that will be required for transaction rollback when the backup is restored, for example:

```
Starting backup pass 3

Journal file 'c:\intersystems\cache62\mgr\journal\20120919.003' and the subsequent ones
are required for recovery purposes if the backup were to be restored

Journal marker set at
offset 197180 of c:\intersystems\cache62\mgr\journal\20120919.003
```

   This information is available in the backup log.

8. When the backup finishes successfully, you see the following:

```
***FINISHED BACKUP***
Global references are enabled.
Backup complete.
%SYS>
```

Note: Following successful completion of the backup, **^DBACK** writes your description of the backup to the backup history (see View Backup History). Note that this can trigger a journal purge, depending on the **When to purge journal files > After this many successive backups** setting on the Journal Settings page (the default is 2); for more information, see Configuring Journal Settings in the "Journaling" chapter of this guide.

The next section, External Entry Points for ^DBACK, explains how to call this utility from your external backup procedures using the provided entry points.

3.7.3.1 External Entry Points for ^DBACK

The entry points provided for the **^DBACK** utility are for you to use in your external backup procedures. You must use them as described to ensure the integrity of the backup. All of the entry points return `1` if they are successful or `0` if they fail. The following external entry points are available for the **^DBACK** routine:

- BACKUP^DBACK
- LISTDIRS^DBACK
- CLRINC^DBACK (Legacy Concurrent External Backup Only)

BACKUP^DBACK

This procedure invokes a backup using the passed arguments to satisfy the questions that are asked during the interactive execution of **^DBACK**.

```
BACKUP^DBACK(argfile,TYPE,DESC,OUTDEV,kiljrn,LOGFILE,MODE,clrjrn,swjrn,
nwjrnfil,quietimeout,taskname)
```

The capitalized arguments are the ones most important to the backup procedure in this release of Caché. Due to changes in the journaling mechanism, you can no longer delete the current journal file, clear the current journal file, or specify a new journal file name in the backup utilities. As a result, this version of Caché ignores many of the arguments as described in the following tables.

Argument	Description
argfile	No longer used; always use a null value.
TYPE	Type of backup (required) Values: I — Incremental C — Cumulative F — Full E — External full backup using a third-party or operating system backup utility
DESC	Description stored in the backup label and in the backup history global. Free form text string that can be empty.

External (type E) backups ignore the arguments in the following table. Whether they are ignored or required for other backup types is also indicated in the table.

Argument	Description
OUTDEV	Output device for the backup; can be a file name or tape device (required).
kiljrn	Indicates whether to switch the journal file: Y — switch the journal file (recommended) N — do not switch (No longer allows deletion of the current journal file)
LOGFILE	Indicates whether or not to log and where to write the output of the function: File name — writes all messages that would be sent to the terminal to this file (recommended) Null value — no log file
MODE	Indicates what output to write to the terminal during the procedure; this does not control what is written to the log file. Values: “NOISY” — write all text on terminal (default) “QUIET” — only write text related to abnormal conditions “NOINPUT” — do not send any output as no terminal is attached to this process; if a read is required, the backup aborts
clrjrn	Indicates whether to switch the journal file: Y — switch the journal file (recommended) N — do not switch (No longer allows clearing of the current journal file)
swjrn	Indicates whether or not to switch the journal file: Y — switch the journal file (recommended) N — do not switch (If the *clrjrn* or *kiljrn* arguments have a value of Y, *swjrn* converts to Y and the journal file is switched)
nwjrnfil	(Ignored)

The following are two optional arguments:

Argument	Description
quietimeout	Number of seconds to wait for the system to quiesce before aborting the backup. A zero (0) or negative value indicates waiting indefinitely (default = 60).
taskname	Task name used internally by Caché for backup types `F`, `I`, and `C`.

Using these external entry points requires read/write access to the ^*SYS* global for all modes. An external (type E) backup deletes the .IND, .INE, and .INF files in the directories in the ^*SYS* global. The routine also records this backup with the description in the backup history global as being the `LASTFULL` backup.

If you set switch 10 when you call this routine, it remains set throughout the backup. If you set switch 13, then the procedure converts it into a switch 10 while the backup performs necessary **Set** and **Kill** commands and then restores it to switch 13 upon exit.

The routine performs the **New** command on variables defined by this entry point, but does not on those defined within the body of the **^DBACK** procedure. As a result, you must be careful to protect all other variables in any calling routine prior to calling **^DBACK**.

Return values:

- `0` — failed; check log file if specified.
- `1` — success
- `1,`*warning message string* — success, but with warnings. The warnings are separated by a tilde character (~) in the string.

LISTDIRS^DBACK

This procedure opens an output file and writes a list to it of database directories included in a backup. If the list is empty, the procedure backs up all database directories (except CACHETEMP). The format of the function is as follows:

```
LISTDIRS^DBACK(file,mode)
```

Argument	Description
file	Name of the output file name to which the function writes the list of database directories in the backup list.
mode	Indicates what output to write to the terminal during the procedure. "QUIET" — Only display return value and text related to abnormal conditions. Any other value — Display the directory list at the terminal.

The following is an example of the output displayed on the terminal:

```
%SYS>w $$LISTDIRS^DBACK("c:\temp\listdirs.txt","NOISY")

List of directories to be backed up
     c:\mycache\mgr\cacheaudit\
     c:\mycache\mgr\cachelib\
     c:\mycache\mgr\
     c:\mycache\mgr\docbook\
     c:\mycache\mgr\samples\
     c:\mycache\mgr\user\
1
%SYS>
```

Notes for this procedure:

- Requires read access to the *^SYS* global.
- Does not set or clear any switches.
- Issues a **New** command on all local variables.

CLRINC^DBACK (Legacy Concurrent External Backup Only)

This procedure is called to clear the incremental backup bitmaps that mark blocks as modified for the databases in the backup list. At the same time it also deletes the backup history as it is no longer possible to do any sort of a backup without first performing a full backup. This procedure is designed to be used prior to doing an external full backup with Caché running. If you perform an external backup with Caché down, then before shutting down the system, call this entry point. The format of the function is as follows:

```
CLRINC^DBACK(mode)
```

Argument	Description
mode	Indicates what output to write to the terminal during the procedure. "QUIET" — Only display return value and text related to abnormal conditions. Any other value — Display the directory names as they are processed.

For example:

```
%SYS>w $$CLRINC^DBACK("QUIET")
1
%SYS>

%SYS>Do CLRINC^DBACK("NOISY")

     Cleared incremental backup bitmap in c:\mycache\mgr\
     Cleared incremental backup bitmap in c:\mycache\mgr\cacheaudit\
     Cleared incremental backup bitmap in c:\mycache\mgr\docbook\
     Cleared incremental backup bitmap in c:\mycache\mgr\samples\
   Cleared incremental backup bitmap in c:\mycache\mgr\user\
%SYS>
```

Notes for this procedure:

- Requires read and write access to the ^*SYS* global.
- Records the state of switch 13, sets it if it is not already, and restores it to its original state upon exit.
- Issues a **New** command on all local variables except those defined in the main backup procedure.
- Leaves directories dismounted if they are dismounted when you call this routine.

3.7.4 Edit/Display List of Directories for Backup Using ^BACKUP

Important: When editing the database list use the database name, not the directory name. This is consistent with the way the backup configuration works in the Management Portal.

Options of the Caché **^BACKUP** utility allow you to backup Caché databases or to restore an already created backup. If you have not created a list of databases, then the utility includes all databases in the backup. When you create a list, it applies to all aspects of the Caché backup system including calls to the entry points of the **^DBACK** utility in scripted backups.

The following examples show sample output from each option on the menu:

1. From the ^BACKUP menu, choose option 4 to display the database backup list maintenance menu:

```
%SYS>Do ^BACKUP


1) Backup
2) Restore ALL
3) Restore Selected or Renamed Directories
4) Edit/Display List of Directories for Backups
5) Abort Backup
6) Display Backup volume information
7) Monitor progress of backup or restore

Option? 4

1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information
```

2. To add a database to the list choose option 1. This example also uses option 3 to show that the database is added to the current list.

```
1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information

Option? 1
Enter database to add? SAMPLES
Enter database to add?
You've selected SAMPLES to be added to the backups
Are you sure you want to do this (yes/no)? y
Completed.

1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information

Option? 3
The following 5 databases are included in backups
     CACHEAUDIT               C:\MyCache\Mgr\cacheaudit\
     CACHESYS                 C:\MyCache\Mgr\
     DOCBOOK                  C:\MyCache\Mgr\Docbook\
     SAMPLES                  C:\MyCache\Mgr\Samples\
     USER                     C:\MyCache\Mgr\User\
```

3. To remove a database to the list choose option 2. This example also uses option 3 to show that the database is removed from the current list.

```
1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information

Option? 2
Enter database to remove (^ when done)?  SAMPLES
Enter database to remove (^ when done)?
You've removed SAMPLES from the backups
Are you sure you want to do this? y
Completed.

1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information

Option? 3
The following 4 databases are included in backups
     CACHEAUDIT               C:\MyCache\Mgr\cacheaudit\
     CACHESYS                 C:\MyCache\Mgr\
     DOCBOOK                  C:\MyCache\Mgr\Docbook\
     USER                     C:\MyCache\Mgr\User\
```

4. To display a list of all databases in this instance choose option 4:

```
1) Add a database to the backup
2) Remove a database from the backup
3) Show current list of databases included in backups
4) Show list of available databases
5) Display last full backup information

Option? 4
The following is a list of all databases in the configuration.
Databases which are part of the backup are marked with (*)
     (*) CACHEAUDIT           c:\MyCache\mgr\cacheaudit\
         CACHELIB (Read Only) c:\MyCache\mgr\cachelib\
         CACHE                c:\MyCache\mgr\cache\
     (*) CACHESYS             c:\MyCache\mgr\
         DOCBOOK              c:\MyCache\mgr\docbook\
         SAMPLES              c:\MyCache\mgr\samples\
     (*) USER                 c:\MyCache\mgr\user\
```

5. To display information about the last full backup choose option 5:

```
    1) Add a database to the backup
    2) Remove a database from the backup
    3) Show current list of databases included in backups
    4) Show list of available databases
    5) Display last full backup information

    Option? 5

    =====Last Full Backup Information=====

    Date: 19 Jan 2011
    Description: Full backup of all databases that are in the backup database list.
    Device: c:\MyCache\mgr\backup\FullDBList_20110119_001.cbk
```

3.7.5 Abort a Running Backup Using ^BACKUP

You can abort a *running* backup using menu option 5 of the **^BACKUP** utility.

```
%SYS>Do ^BACKUP


1) Backup
2) Restore ALL
3) Restore Selected or Renamed Directories
4) Edit/Display List of Directories for Backups
5) Abort Backup
6) Display Backup volume information
7) Monitor progress of backup or restore

Option? 5
Are you sure you want to abort the backup operation ? <No> Yes
Backup abortion succeed.
```

If no backup is running when you choose this option, you receive the following message:

```
No Backup operation is currently running.
```

3.7.6 Display Information About a Backup Volume Using ^BACKUP

When you select option 6 from the **^BACKUP** menu, you are prompted for the pathname of a backup file, as shown in the following:

```
Enter the backup volume you want to display information from
 (Type STOP to exit)
Device: c:\131u1\mgr\backup\FullAllDatabases_20120802_001.cbk =>

This backup volume was created by:
Cache for Windows (x86-64) 2013.1 (Build 257U) Sun Jul 1 2012 23:35:31 EDT

The volume label contains:
   Volume number       1
   Volume backup       AUG 2 2012 12:27PM Full
   Previous backup     AUG 2 2012 11:35AM Cumulative Incremental
   Last FULL backup    AUG 2 2012 11:24AM
   Description         Full backup of ALL databases
   Buffer Count        0
   Mirror name         MIRROR122
   Failover member     NONNIE20/CACHE122A
   Journal File        c:\131u1\mgr\journal\20120802.005
   Log file            c:\131u1\mgr\Backup\FullAllDatabases_20120802_001.log
   Backup Status       Completed

Database                              Size(mb)  Mirror DB Name
--------                              --------  --------------
c:\131u1\mgr\                              191
c:\131u1\mgr\cacheaudit\                    11
c:\131u1\mgr\user\                        2113  MUSER
Total of 3 databases, totaling 2315 mb
Backup volume size is 2365 mb
```

3.7.7 Monitor Backup or Restore Progress Using ^BACKUP

To monitor the progress of a running backup or restore job started using **^BACKUP**, select option 7 from the **^BACKUP** menu. You can start the monitor in a separate Terminal window before starting the backup or restore job.

For a running backup job, the following information is displayed:

```
                              Backup Status
                              -------------

Status:                       Running
Backup process pid:           13052
Backup file:                  c:\132u1\mgr\stcfull.bck
Description:
Log file:                     c:\132u1\mgr\cachebackup.log
Type:                         Full
Start time:                   2012-09-05 14:09:02
Estimated finish time:        2012-09-05 14:09:45
Total time (hh:mm:ss):        00:00:18
Time remaining (hh:mm:ss):    00:00:25
Estimated backup size:        7.68GB
Current backup size:          3.12GB
Current Backup rate:          184MB/sec
% Completed                   41
```

For a running restore job, the following information is displayed:

```
                              Restor Status
                              -------------

Status:                       Running
Restore process pid:          13052
Restore file:                 c:\132u1\mgr\stcfull.bck
Description:
Type:                         Full
Start time:                   2012-09-05 14:10:49
Estimated finish time:        2012-09-05 14:12:07
Total time (hh:mm:ss):        00:00:34
Time remaining (hh:mm:ss):    00:00:44
Restore file size:            7.68GB
Restore size completed:       3.28GB
Current restore rate:         103MB/sec
% Completed                   43
```

Note: When doing an incremental backup, some of the fields are not available and are displayed as `N/A`. This includes `Time remaining`, `Estimated backup size`, and `% Completed`.

A restore job cannot be monitored if switch 10 was set when starting the restore.

Tape backup and restore jobs cannot be monitored using this utility.

3.8 Caché Online Backup Restore Utility

The Caché Online Backup Restore Utility (**^DBREST**) utility performs the following actions:

- Requires you to choose whether to restore all or selected directories.
- Asks if you want to stop all other Caché processes from running during the restore. In most cases, answer Yes.
- Asks for the name of the file that holds the backup you wish to restore. If your backup is on more than one volume, that information is in the volume header. After restoring the first volume, the utility prompts you for the next.
- Displays the header of the volume. The header contains the following information determined during the backup: the date of the backup on this volume, the date of the previous backup, and the date of the last full backup.
- Asks you to verify that this is the backup you wish to restore.

- Lists the directories on the device to restore.
- Allows you to specify another input device that contains the next backup to restore.
- Lets you display information about a backup volume at any time.

The **^DBREST** menu options 1, 2 and 3 are the same as options 2, 3 and 6 from the **^BACKUP** menu.

When you select one of these restore options, the utility asks you to enter the name of the device holding the first backup to restore. The default the first time you enter a restore option is the device to which the last full backup was sent, if there was one.

CAUTION: When you are restoring an incremental or cumulative backup, the target database must be in *exactly* the same state as when you restored it from the last full backup.

You must prevent all updates to the restored databases until you restore all subsequent incremental backups. Failure to heed this warning can result in the incremental or cumulative restore producing a degraded database.

The following is an example of running the utility from the Terminal:

```
%SYS>Do ^DBREST

                      Cache DBREST Utility
         Restore database directories from a backup archive

Restore: 1. All directories
         2. Selected and/or renamed directories
         3. Display backup volume information
         4. Exit the restore program
    1 =>
```

The following sections describe the process of choosing each option:

1. Restore All Databases Using ^DBREST
2. Restore Selected or Renamed Databases Using ^DBREST
3. Display Information About a Backup Volume Using ^BACKUP
4. Restoring Databases Using the Backup History
5. Unattended Restore Using ^DBREST
6. Mirrored Database Considerations

You can also perform these functions non-interactively in a script. See the *External Entry Points of ^DBREST* section for details.

3.8.1 Restore All Databases Using ^DBREST

Choosing 1 from the **^DBREST** menu is equivalent to choosing 2 from the **^BACKUP** menu.

Note: If you are restoring mirrored databases, review Mirrored Database Considerations in this section before you perform this procedure.

The following procedure is an outline of an example that restores all directories. It shows the beginning prompts of the restore process. As it continues, the utility asks you very specific questions while stepping through the restore process.

1. Select to restore all directories from the utility menu. This option restores all directories that are on the backup medium.

```
%SYS>Do ^DBREST

                        Cache DBREST Utility
          Restore database directories from a backup archive

Restore: 1. All directories
         2. Selected and/or renamed directories
         3. Display backup volume information
         4. Exit the restore program
     1 => 1
```

2. Confirm that you want to restore all directories:

```
Proceed with restoring ALL directories Yes=>
```

3. Next you are asked to enter the top path you want all the databases to be restored to. The system prefixed it to the original path of the database to be restored. So all the databases are restored under this directory with their original directory as a subdirectory to this directory. Press **Enter** to ignore it and restore the databases to their original path.

```
Top directory for all Databases to be restored to (? for Help)?
```

4. Indicate whether you want to suspend Caché processes while restoring takes place. InterSystems recommends suspending processes.

```
Do you want to set switch 10 so that other processes will be
prevented from running during the restore? Yes =>
```

5. Specify the first file from which to restore. You can press **Enter** to accept the default file, which is the last full backup.

```
Specify input file for volume 1 of backup 1
 (Type STOP to exit)
Device: c:\mycache\mgr\backup\FullAllDatabases_20110323_001.cbk =>
```

6. Check that the description of the backup is correct and verify that this is the file you want to restore.

```
This backup volume was created by:
   Cache for Windows (Intel) 5.1

The volume label contains:
   Volume number      1
   Volume backup      MAR 23 2011 09:52AM Full
   Previous backup    MAR 22 2011 11:00AM Incremental
   Last FULL backup   MAR 16 2011 11:00AM
   Description        Full backup of ALL databases, whether or not they are in
                      the backup database list.
   Buffer Count       0
Is this the backup you want to start restoring? Yes =>
```

7. The utility tells you which directories it will restore, and the restore proceeds.

```
The following directories will be restored:
c:\mycache\mgr\
c:\mycache\mgr\cacheaudit\
c:\mycache\mgr\samples\
c:\mycache\mgr\test\
c:\mycache\mgr\user\


***Restoring c:\mycache\mgr\ at 10:46:01
146045 blocks restored in 241.3 seconds for this pass, 146045 total restored.
```

8. Specify the input file for the next incremental backup to restore, or enter `stop` if there are no more input files to restore.

```
Specify input file for volume 1 of backup following MAR 23 2011  09:52AM
 (Type STOP to exit)
Device: stop
```

9. Indicate whether you want to restore other backups. When you answer `Yes`, the procedure repeats. When you respond `No`, Caché mounts the databases you have restored.

```
Do you have any more backups to restore? Yes => No
Mounting c:\mycache\mgr\
    c:\mycache\mgr\       ... (Mounted)

Mounting c:\mycache\mgr\cacheaudit\
    c:\mycache\mgr\cacheaudit\  ... (Mounted)

Mounting c:\mycache\mgr\samples\
    c:\mycache\mgr\samples\  ... (Mounted)

Mounting c:\mycache\mgr\test\
    c:\mycache\mgr\test\  ... (Mounted)

Mounting c:\mycache\mgr\user\
    c:\mycache\mgr\user\  ... (Mounted)
```

10. Specify which journal entries you want to apply to the restored databases and the name of the journal file you are restoring. Normally, you select option `1` and apply only those changes that affect the directories you have just restored.

    Note: For information about what happens when you restore all entries for all directories or selected directories and globals, see Restore Globals From Journal Files Using ^JRNRESTO in the "Journaling" chapter of this guide.

```
Restoring a directory restores the globals in it only up to the
date of the backup.  If you have been journaling, you can apply
journal entries to restore any changes that have been made in the
globals since the backup was made.

What journal entries do you wish to apply?

     1. All entries for the directories that you restored
     2. All entries for all directories
     3. Selected directories and globals
     4. No entries

Apply: 1 =>
```

11. Restore from the journal files begins after confirming several pieces of information:

```
We know something about where journaling was at the time of the backup:
0: offset 172940 in c:\mycache\mgr\journal\20110323.002

Use current journal filter (ZJRNFILT)? No
Use journal marker filter (MARKER^ZJRNFILT)? No
Updates will not be replicated

The earliest journal entry since the backup was made is at
offset 172940 in c:\mycache\mgr\journal\20110323.002

Do you want to start from that location? Yes => Yes
Final file to process (name in YYYYMMDD.NNN format): <20110323.003> [?]
      =>

Prompt for name of the next file to process? No => No

Provide or confirm the following configuration settings:

Journal File Prefix: =>

Files to dejournal will be looked for in:
     c:\mycache\mgr\journal\
     c:\journal\altdir\
in addition to any directories you are going to specify below, UNLESS
you enter a minus sign ('-' without quotes) at the prompt below,
in which case ONLY directories given subsequently will be searched

Directory to search: <return when done>
Here is a list of directories in the order they will be searched for files:
     c:\mycache\mgr\journal\
     c:\journal\altdir\
The journal restore includes the current journal file.
You cannot do that unless you stop journaling or switch
     journaling to another file.
```

```
Do you want to switch journaling? Yes => Yes
Journaling switched to c:\mycache\mgr\journal\20110323.004

You may disable journaling the updates for faster restore; on the other hand,
you may not want to do so if a database to restore is being shadowed.
Do you want to disable journaling the updates? Yes => yes
Updates will NOT be journaled
```

12. The utility displays its progress and indicates when it is complete:

```
c:\mycache\mgr\journal\20110323.002
  61.32%  65.03%  68.44%  72.21%  75.86%  79.26%  82.73%  86.08%  89.56%
  92.99%  96.07%  98.87%100.00%
***Journal file finished at 11:03:31


c:\mycache\mgr\journal\20110323.003
  16.17%  17.10%  17.90%  18.90%  20.05%  21.33%  22.58%  23.81%  25.15%
  26.32%  27.65%  28.85%  30.08%  31.37%  32.59%  33.98%  35.16%  36.25%
  37.32%  38.41%  39.55%  40.72%  41.81%  42.83%  43.85%  44.89%  46.00%
  47.15%  48.24%  49.28%  50.32%  51.41%  52.54%  53.71%  54.76%  55.80%
  56.85%  57.97%  59.10%  60.16%  61.17%  62.19%  63.24%  64.32%  65.18%
  66.02%  66.87%  67.71%  68.52%  69.34%  70.14%  70.96%  71.76%  72.60%
  73.58%  74.51%  75.43%  76.35%  77.26%  78.17%  79.07%  79.69%  80.31%
  80.93%  81.56%  82.20%  82.83%  83.47%  84.27%  87.00%  88.57%  91.65%
  93.03%  96.09%  97.44%  99.04%100.00%
***Journal file finished at 11:03:32

Journal reads completed. Applying changes to databases...
  14.29%  28.57%  42.86%  57.14%  71.43%  85.71% 100.00%

[journal operation completed]
```

3.8.2 Restore Selected or Renamed Databases Using ^DBREST

Choosing 2 from the **^DBREST** menu is equivalent to choosing 3 from the **^BACKUP** menu. This option lets you select which directories to restore from the backup medium. It also allows you to restore a database to a different directory name or to a different device.

Note: If you are restoring mirrored databases, review Mirrored Database Considerations in this section before you perform this procedure.

The following example shows how to restore selected or renamed directories.

1. Choose option 2 from the **^DBREST** utility:

```
%SYS>Do ^DBREST

                      Cache DBREST Utility
         Restore database directories from a backup archive

Restore: 1. All directories
         2. Selected and/or renamed directories
         3. Display backup volume information
         4. Exit the restore program
    1 => 2
```

2. Indicate whether you want to suspend Caché processes while restoring takes place. InterSystems recommends suspending processes.

```
Do you want to set switch 10 so that other Cache processes
will be prevented from running during the restore? Yes =>
```

3. Specify the first file from which to restore. You can press **<Enter>** to accept the default file, which is the last full backup.

```
Specify input file for volume 1 of backup 1
 (Type STOP to exit)
Device: c:\mycache\mgr\backup\IncrementalDBList_20110323_001.cbk =>
```

4. Check that the description of the backup is correct and verify that this is the file you want to restore.

```
This backup volume was created by:
   Cache for Windows (Intel) 5.1

The volume label contains:
   Volume number      1
   Volume backup      MAR 23 2011 11:03AM Full
   Previous backup    MAR 23 2011 09:52AM Full
   Last FULL backup   MAR 23 2011 09:52AM
   Description        Incremental backup of all databases that are in the backup
                      database list.
   Buffer Count       0
Is this the backup you want to start restoring? Yes =>
```

5. As the utility prompts you with directory names, specify which databases you want to restore, and in which directories you want to restore them:

```
For each database included in the backup file, you can:

 -- press RETURN to restore it to its original directory;
 -- type X, then press RETURN to skip it and not restore it at all.
 -- type a different directory name.  It will be restored to the directory
    you specify.  (If you specify a directory that already contains a
    database, the data it contains will be lost).

c:\mycache\mgr\ =>
c:\mycache\mgr\cacheaudit\ =>
c:\mycache\mgr\test\ =>
c:\mycache\mgr\user\ =>
```

6. After responding to each directory prompt, you see the prompt:

```
Do you want to change this list of directories? No =>
```

   Answer `Yes` if you want to edit your choices, or press **Enter** to confirm them.

7. The process then continues the same as the procedure for restoring all directories, as specified in the previous section.

You can use this procedure to restore a backup performed on different system from the one on which you are restoring it.

3.8.3 Restoring Databases Using the Backup History

The Caché backup utility maintains a *backup history* to help you restore backups in logical order. The restore utility prompts you for the backups to restore according to their type and order in the backup history.

After restoring the last full backup, the utility uses the information in the *backup history* to suggest the next logical backup for you to restore and continues through the history. It prompts you to restore subsequent backups in the following order:

1. The most recent cumulative backup after the last full backup, if one exists.
2. All incremental backups since the last cumulative backup (or if none exists, since the last full backup). It does so in order from the first to the most recent.

You can override the suggested backups in the restore process. Remember, however, that an incremental or cumulative backup does not represent a complete copy of the databases. You can restore an incremental backup only after restoring a full backup.

Important: On Caché platforms that support access to a database from cluster, you should always back up a given database directory from the same node so that its complete backup history is available if you need to restore the directory.

3.8.4 Unattended Restore Using ^DBREST

The Caché restore utility, **^DBREST**, provides a non-interactive execution option using external entry points.

Note: If you are restoring mirrored databases, review Mirrored Database Considerations in this section before you perform an unattended restore.

You can write a script to implement unattended restores by calling one of these two entry points:

- **EXTALL^DBREST** — Restores all databases from the backup device (the unattended equivalent to the procedure described in Restore All Databases Using ^DBREST). The syntax to use the EXTALL entry point of the **^DBREST** utility is as follows:

  ```
  EXTALL^DBREST(quietmode,allowupd,inpdev,dirlist,jrnopt,jrnfile,jdirglo)
  ```

  Note: If the target directory exists but the CACHE.DAT file does not, **EXTALL^DBREST** restores the CACHE.DAT file.

- **EXTSELCT^DBREST** — Restores selected files from the backup device, or restores to a target directory that is different from the source directory (the unattended equivalent to the procedures described in Restore Selected or Renamed Databases Using ^DBREST). The syntax to use the EXTSELCT entry point of the **^DBREST** utility is as follows:

  ```
  EXTSELCT^DBREST(quietmode,allowupd,inpdev,dirlist,jrnopt,jrnfile,jdirglo)
  ```

Both entry points are functions, which return the status of the call. All arguments are input only. The following table describes the input arguments used for both functions except where indicated.

Argument	Description
allowupd	Indicates whether or not to allow updates during the restore process: 1 — allow updates during the restore process 0 — do not allow updates
dirlist	Name of file containing a list of directories to restore. One record for each directory to be restored has to be present. Ignored for the EXTALL entry point. See *Directory List File Requirements*.
inpdev	Input device that contains the backup. If this device is a tape device, the utility prompts you to specify the device for the next volume.
jdirglo	Only used when *jrnopt* is 3. This is the name of the file containing selection criteria for directories and globals for the journal restore. See *Directory and Global Selection Criteria*.
jrnfile	Journal file. If null, the utility uses the current file, which is stored in the *^%SYS("JOURNAL","CURRENT")* global.
jrnopt	Options to restore the journal: 1 — All directories for which you just restored the backup 2 — All directories in the journal 3 — Selected directories and globals specified by the *jdirglo* argument 4 — No directories
quietmode	A value indicates the operation is in quiet (non-interactive) mode; must be a non-null value for this external call, typically 1.

The following return codes are possible from calling these functions:

Return Code	Description
3	No errors; successful completion
-1	Cannot open input device (device to restore from)
-2	Volume label does not match label in backup history
-3	Backup/Restore already in progress
-4	Invalid device for reading selection criteria (for selective restore of directories)
-5	Invalid journal restore option
-6	Invalid journal file or file for selection criteria for journal restore cannot be opened

3.8.4.1 Directory List File Requirements

Requirements of the file indicated by *dirlist*:

- Contains one record for each directory to restore
- Separate each field with a comma (`,`).
- Format of each record is *SourceDir*,*TargetDir*,*CreateDir*; where:

Argument	Description
SourceDir	Name of the directory to restore.
TargetDir	Name of the directory to which to restore; omit if restoring to same as source directory.
CreateDir	Whether or not to create the target directory if it does not exist. Y — create the target directory N — do not create target directory

For example, assuming you have created a text file (RestoreList.txt) in the C:\Backup\ directory that contains the string:

```
C:\intersystems\cache\mgr\user\,C:\intersystems\cache\mgr\,Y
```

you could execute the following routine in the %SYS namespace to restore the database from `C:\intersystems\cache\mgr\user\` to `C:\intersystems\cache\mgr\`:

```
set SourceDir = "C:\intersystems\cache\mgr\user\"
set BackupDir = "C:\Backup\"
set BackupListFile="C:\Backup\RestoreList.txt"
do EXTSELCT^DBREST(1,0,BackupDir_"back1.cbk",BackupListFile,4,"","")
```

3.8.4.2 Directory and Global Selection Criteria

Requirements of the file indicated by *jdirglo*:

- Contains one record for each directory that you want to restore.
- Separate each field with a comma (`,`).
- Format of each record is *DirName*, *RestAll*, *Globals*; where:

Argument	Description
DirName	Name of the directory on which you want to restore the journal.
Globals	A list of globals separated by commas for journal restore. Only used if RestAll is N. If the list of globals is large, you can enter the remaining global names on the next line but you must specify the directory name again followed by N and then the list of globals.
RestAll	Whether or not to restore journal entries for all globals in the directory; not case-sensitive and required. Y — restore journal on all globals in this directory N — specify globals for journal restore in globals list

Examples of different records:

```
DUA0:[TEST1],Y
DUA1:[TEST2],N,GLO1,GLO2,GLO3
DUA1:[TEST2],N,GLO4,GLO5,GLO6
DUA1:[TEST3],n
```

The last record could also be completely omitted if you do not want to restore the journal for that directory.

3.8.5 Mirrored Database Considerations

When a mirrored database is the target of a Caché backup restore, the source database must match the target (that is, it must be the same mirrored database).

Note: Mirrored database backups created on a failover mirror member can be restored to any member of that mirror (that is, another failover member or an async member). If you are restoring from a backup created on an async mirror member, see Restoring from Async Mirror Members in this section.

In this section, the term "active mirrored database" refers to a database that is currently included in a mirror to which its host system belongs, even if that mirror is not currently in operation. It does not include databases that were once mirrored, or that were included in a mirror to which the host system no longer belongs.

Caché backup restore in a mirror behaves differently depending on whether it is being restored on the mirror member where the backup was created or on a different system:

- Primary Failover Member — **^DBREST** does not let you restore an active mirrored database on the primary failover member. However, if the database must be restored, you can temporarily remove it from the mirror (see Removing Mirrored Databases from Mirrors in the "Mirroring" chapter of the *High Availability Guide*). The database is restored as a mirrored database, however (since it was backed up as one), and the functionality described in the following sections applies, including automatic restore of the journal files required to activate the database in the mirror.

- Backup Failover Member — On the backup failover member, restoring a mirrored database that is active, or newer than the copy in the backup, results in a warning message:
  - For a full backup restore, these databases are skipped.
  - For a selective restore, you are given a chance to overwrite the target if that is what you really want to do.

3.8.5.1 Full Backup Restores of Mirrored Databases

The following differences apply to full backup restores of mirrored databases, whether they are done interactively via the **^DBREST** utility (see Caché Online Backup Restore Utility in this chapter) or non-interactively via the EXTALL entry point (see External Entry Points of ^DBREST in this chapter):

- A full backup restore on the system that created the backup works as it does for non-mirrored systems, *except* that local mirrored databases that are newer than the copies in the backup — which, by definition, includes databases that are currently active — are skipped; all databases are restored to their original locations.
- A full backup restore on a mirror member other than the system that created the backup restores only mirrored databases that already exist on the system. However, mirrored databases on the target system that are newer than those in the backup are skipped.
- A full backup restore that specifies a new top-level directory restores all the databases in the backup after generating the new path for the databases. If the user ends up with two copies of any of the mirrored databases after the restore, they are warned that the copy being restored cannot be activated because it is already active on the system.

Following a full backup restore the system attempts to:

1. Activate the mirrored databases by restoring the required journal files.
2. Link the mirrored databases into the active mirror (if the mirror exists and a copy of the database does not already exist).

3.8.5.2 Selective Backup Restores of Mirrored Databases

The following differences apply to selective backup restores of mirrored databases:

- When a backup is being restored, you are asked whether to limit the restore to only mirrored databases; depending on your response:
  - yes — Only the mirrored databases are displayed.
  - no — You are prompted to select destinations for all the databases in the backup.
- During the database selection phase of a selective restore, you are presented with a list of the databases in the backup and asked to specify a path where each should be restored. You can specify a path, enter an "X" or "x" to skip that directory, or press **Enter** to restore it to the path stored in the backup, if you are restoring on the source system.

  When restoring a backup on a mirror member other than the system that created the backup, the result of pressing **Enter** for a mirrored database directory varies depending on whether an active mirrored database with the same mirror database name exists on that system.
  - If the database exists, pressing **Enter** overwrites the existing database with the restored database, regardless of the existing database's directory path, including whether that path is the same as on the source system.
  - If the database does not exist, the database is restored to the directory path it has on the source machine. If that path does not exist, you must confirm the creation of the needed directories. If a database already exists at that location, no action is taken.
- If a selective backup restore detects that the restore is going to overwrite a mirrored databases in the following situations, you are warned that continuing will destroy the target database and asked if you want to continue:
  - The database from the backup is older than the current database.
  - The database from the backup is not a copy of the current database (for example, it could be a non-mirrored database or a database of the same mirror database name from a different mirror).

  If you continue, the target database is removed from the mirror and overwritten as requested.

- If a selective backup restore is being done non-interactively via the EXTSELECT entry point (see External Entry Points of ^DBREST in this chapter), the database selection file the source and target paths can be specified with either path names or mirror databases names.

  For a mirrored database, if the target is left blank it means restore to the corresponding mirror database on the local machine. If the target is not blank, then the database must be one of the following:

  – The correct, local copy of that mirrored database.

  – If a local copy of that mirrored database does not already exist on the system, a non-mirrored database (or a directory/database that does not exist) .

  – A mirror database name.

  Note: If the target is a mirror database name (or blank) and that mirrored database does not exist on the local system (or is dismounted), the target is created only if the backup is being restored on the system that created it. If the backup is being restored on another mirror member the database is skipped.

As with full backup restores, following a selective backup restore, the system attempts to:

1. Activate the mirrored databases by restoring the required journal files.
2. Link the mirrored databases into the active mirror (if the mirror exists).

3.8.5.3 Restoring from Async Mirror Members

The following considerations apply when restoring from mirrored databases created on async mirror members:

- Mirrored databases backups created on a async mirror member can be restored to any async member of the same mirror.
- Mirrored databases backups created on a async mirror member can only be used to create new copies of the mirrored database (that is, they cannot be restored over an existing mirrored database on a failover member).

4

Journaling

Global journaling records all global update operations performed on a database, and used in conjunction with backup makes it possible to restore a database to its state immediately before a failure or crash.

While backup is the cornerstone of physical recovery, it is not the complete answer. Restoring a database from backup does not recover global updates made since that backup, which may have been created a number of hours before the point at which physical integrity was lost. These post-backup updates can be restored to the database from journal files after the database is restored from backup, bringing the database up to date. Any transactions open at the time of the failure are rolled back to ensure transaction integrity.

This chapter discusses the following topics:

4.1 Journaling Overview

Each instance of Caché keeps a journal, a set of files that keeps a time-sequenced log of updates that have been made to databases since the last backup. The process is redundant and logical and does not use the Caché write daemon. Caché transaction processing works with journaling to maintain the logical integrity of data following a failure.

Together, backup and journaling allow you to recreate your database. If a failure renders your database corrupt, inaccessible or unusable, you can restore the most recent backup and then apply the changes in the journal to recreate your database to the point of failure. This method of recovering from a loss of physical integrity is known as "roll forward" recovery. The journal is also used for rolling back incomplete transactions.

The journaling state is a property of the database, not individual globals. A database can have only one of two global journaling states: *Yes* or *No*. By default, all databases you create are journaled (the **Global Journal State** is *Yes*). In newly installed Caché instances, the CACHEAUDIT, CACHESYS, and USER databases are journaled; the CACHELIB, CACHETEMP, DOCBOOK, CACHE, and SAMPLES databases are not. Operations to globals in CACHETEMP are never journaled; map temporary globals to the Caché temporary database, CACHETEMP.

Important: Be sure to read Consequences of Not Journaling Databases for important information about limits to the recovery of non-journaled databases.

When Caché starts, it reapplies all journal entries since the last write daemon pass. Since user processes update the journal concurrently, rather than through the write daemon, this approach provides added assurance that updates prior to a crash are preserved.

In addition to recording all updates to journaled databases, the journal contains all updates to non-journaled databases that are part of transactions (primarily **Set** and **Kill** operations). This greatly improves the reliability of the system, avoiding post-recovery inconsistencies due to updates to globals that may or may not be journaled, and that may or may not be involved in transactions. (**Set** and **Kill** operations on local and process-private variables are not journaled.)

Journaling global operations in databases mounted on a cluster depends on the database setting. The local Caché instance does not journal transaction operations to globals on remote nodes. In a network configuration, journaling is the responsibility of the node on which the global actually resides, not the one that requests the **Set** or **Kill**. Thus, if node B performs a **Set** at the request of node A, the journal entry appears in the journal on node B, not node A.

Note: If you need to journal restore a privately mounted database on a Caché cluster node from a point prior to the last crash/start/restart of the node, use the cluster journal restore procedure (see Cluster Journal Restore in the "Cluster Journaling" chapter of this guide) instead of noncluster journal restore because open ECP transactions from the crashed node are transferred to the surviving node rather than rolled back. InterSystems recommends that you back up the databases after a cluster failover recovery, which makes it unnecessary to start journal restore from a pre-failover point.

The following topics provide greater detail of how journaling works:

- Differences Between Journaling and Write Image Journaling
- Protecting Database Integrity
- Automatic Journaling of Transactions
- Rolling Back Incomplete Transactions
- Consequences of Not Journaling Databases
- The Journal Write Cycle
- Journal Files and Journal History Log
- Using Temporary Globals and CACHETEMP
- Journal Management Classes and Globals

4.1.1 Differences Between Journaling and Write Image Journaling

In this chapter, "the journal" refers to the journal file; "journaling" refers to the writing of global update operations to the journal file. Do not confuse the Caché journal described in this chapter with write image journaling, which is described in the "Write Image Journaling and Recovery" chapter of this guide. Journaling and write image journaling have different functions, as follows:

- Journaling provides a complete record of all database modifications (as long as you have journaling enabled for the database). In the event that some modifications are lost, for example because they occurred after the most recent backup of a recovered database, you restore them to the database by restoring the contents of the journal file.
- Write image journaling provides a record of all database modifications that have been made in memory but not yet written to the database. When a system crash occurs, the system automatically writes the contents of the write image journal to the database when it restarts.

4.1.2 Protecting Database Integrity

The Caché recovery process is designed to provide maximal protection:

- It uses the "roll forward" approach. If a system crash occurs, the recovery mechanism completes the updates that were in progress. By contrast, other systems employ a "roll back" approach, undoing updates to recover. While both approaches protect internal integrity, the roll forward approach used by Caché does so with reduced data loss.
- It protects the sequence of updates; if an update is present in the database following recovery, all preceding updates are also present. Other systems which do not correctly preserve update sequence may yield a database that is internally consistent but logically invalid.
- It protects the incremental backup file structures, as well as the database. You can run a valid incremental backup following recovery from a crash.

4.1.3 Automatic Journaling of Transactions

In a Caché application, you can define a unit of work, called a transaction. Caché transaction processing uses the journal to store transactions. Caché journals any global update that is part of a transaction regardless of the global journal state setting for the database in which the affected global resides.

You use commands to:

- Indicate the beginning of a transaction.
- Commit the transaction, if the transaction completes normally.
- Roll back the transaction, if an error is encountered during the transaction.

Caché supports many SQL transaction processing commands. See the "Transaction Processing" chapter of *Using Caché ObjectScript* for details on these commands.

4.1.4 Rolling Back Incomplete Transactions

If a transaction does not complete, Caché rolls back the transaction using the journal entries, returning the globals involved to their pre-transaction values. As part of updating the database, Caché rolls back incomplete transactions by applying the changes in the journal, that is, by performing a journal restore. This happens in the following situations:

- During recovery, which occurs as part of Caché startup after a system crash.
- When you halt your process while transactions are in progress.
- When you use the **Terminate** option to terminate a process from the Process page of the Management Portal (**System Operation** > **Process**). If you terminate a process initiated by the **Job** command, the system automatically rolls back any incomplete transactions in it. If you terminate a user process, the system sends a message to the user asking whether it should commit or roll back incomplete transactions.

You can write roll back code into your applications. The application itself may detect a problem and request a rollback. Often this is done from an error-handling routine following an application-level error.

See the Managing Transactions Within Applications section of the "Transaction Processing" chapter of *Using Caché ObjectScript* for more information.

4.1.5 Consequences of Not Journaling Databases

Databases that are not journaled do not participate in journal recovery and transaction rollback at Caché startup. As a consequence, the following conditions apply after a failure, backup restore, and restart:

- The most recent updates to non-journaled databases can be lost; the data in these databases will represent an earlier moment in time than the data in journaled databases.
- Transactions or portions of transactions in non-journaled databases can be left partially committed. The durability provided by synchronous-commit transactions does not apply to databases that are not journaled.
- While updates to non-journaled databases that are part of transactions are journaled, these journal records are used only to roll back transactions during normal operation. These journal records do not provide a means to roll back transactions at startup, nor can they be used to recover data at startup or following a backup restore.

4.1.6 The Journal Write Cycle

The operation that writes the contents of the journal buffer to the journal file is called a *journal sync*. A journal sync is guaranteed to write all operations currently in the journal buffer to the current journal file.

The frequency with which the journal is synced depends on the operating circumstances of the Caché instance involved. A journal sync can be triggered:

- Once every two (2) seconds if the system is idle.
- In an ECP configuration, by the data server when responding to specific requests (for example, **$Increment**) from the application servers to guarantee ECP semantics.
- By a **TCOMMIT** (in synchronous commit mode, which causes the data involved in that transaction to be flushed to disk) if you are using Caché transactions.
- As part of every database write cycle by the write daemon.
- When the journal buffer is full.

4.1.7 Journal Files and Journal History Log

Journal files are stored in the primary journal directory (*install-dir*\Mgr\journal by default) and are logged in the journal history log file, *install-dir*\Mgr\journal.log, which contains a list of all journal files maintained by the instance. The log is used by all journal-related functions, utilities, and APIs to locate journal files.

The journal history log file is updated as follows:

- Entries are added to the log when a new journal file is created.
- Entries are purged periodically, starting from the beginning of the file, *if* the corresponding journal file identified by the entry no longer exists and the entry is 30 days old or older. Purging stops when an entry is reached that does not satisfy both criteria.

CAUTION: Do not modify the journal.log file. If the file is modified outside of the journal utilities, it may be viewed as being corrupt, which may disable journaling. If the file is corrupt, contact the InterSystems Worldwide Response Center (WRC) for guidance. If journaling is disabled (that is, Caché is not able to update the journal.log file), rename the corrupt log file and restart journaling.

InterSystems recommends that you include the journal.log file in your backup strategy to ensure that it is available when needed for a journal restore following a backup restore; for information about backup and restore strategies and procedures, see the "Backup and Restore" chapter in this guide.

If the journal.log file is missing (for example, if you renamed the file because it is corrupt), the system creates a new one when a new journal file is created, but information about previous journal files is lost because the log file only lists journal files created since it was created. Unlisted journal files are not available for journal-related functions, utilities, and APIs that use the journal.log file. However, for journal restores, if the journal.log file is missing or you do not want to use the existing log file, you can specify the journal files manually (see the "Restore Globals from Journal Files Using ^JRNRESTO" section in this chapter).

In addition, you can use the journal.log file to migrate/restore journal files to different locations, as follows:

1. Copy the journal files and journal.log file to a location *other than* the *install-dir*\Mgr directory on the target Caché instance.
2. On the target system, run the **^JRNRESTO** routine, and enter No in response to the following prompt:

   ```
   Are journal files created by this Cache instance and located in their original
   paths? (Uses journal.log to locate journals)?
   ```

3. When prompted, specify the locations (on the target system) of the copied journal files and journal.log file; **^JRNRESTO** uses the log file to validate the range of journal files you want to migrate/restore to the target system.
4. Complete the process as described in the "Restore Globals from Journal Files Using ^JRNRESTO" section in this chapter.

Note: When a Caché instance becomes a member of a mirror, the following journaling changes to support mirroring occur:

- On becoming primary, a journal switch is triggered to a new journal file prefixed with MIRROR-*mirror_name*, for example MIRROR-MIR21-20120921.001. From that point, all journal files are written as mirror journal files and logged to the mirrorjrn-*mirror_name*.log, for example mirrorjrn-MIR21-20120921.log, as well as to journal.log.
- On becoming backup or async, mirror journal files received from the primary are written to the configured journal directory along with the local instance's standard journal files, and a copy of the primary's mirror journal log (mirrorjrn-*mirror_name*.log) is created in *install-dir*\Mgr and continuously updated.

For more information about the role of journal files in mirroring, see Mirror Synchronization in the "Mirroring" chapter of the *Caché High Availability Guide*.

4.1.8 Using Temporary Globals and CACHETEMP

Nothing mapped to the CACHETEMP database is ever journaled.

Since the globals in a namespace may be mapped to different databases, some may be journaled and some may not be. It is the journal property for the database to which the global is mapped that determines if Caché journals the global operation. The difference between CACHETEMP and a database with the journal property set to **No** is that nothing in CACHETEMP, not even transactional updates, are journaled.

Note: A database configured with the **Journal globals** property set to **No** (see the Create Local Databases in the "Configuring Caché" chapter of the *Caché System Administration Guide*) continues to journal global **Set**/**Kill** operations in journal transactions, which can cause the journal file to become very large. CACHETEMP, however, does not journal **Set**/**Kill** operations, even when they are in a journal transaction.

If you need to exclude new z/Z* globals from journaling, map the globals to a database with the journal property set to **No**. To always exclude z/Z* globals from journaling, you must map them in every namespace to the CACHETEMP database.

Caché does not journal temporary globals. Some of the system globals designated by Caché as temporary and contained in CACHETEMP are:

- *^%cspSession*
- *^CacheTemp**
- *^mtemp**

4.1.9 Journal Management Classes and Globals

See the class documentation for %SYS.Journal.System in the *InterSystems Class Reference* for information on available journaling methods and queries. It is part of the %SYS.Journal package.

Also, Caché uses the *^%SYS("JOURNAL")* global node to store information about the journal file. For example:

- *^%SYS("JOURNAL","ALTDIR")* stores the name of the alternate journal directory.
- *^%SYS("JOURNAL","CURDIR")* stores the name of the current journal directory.
- *^%SYS("JOURNAL","CURRENT")* stores journal status and the journal file name.

You can view this information from the **System Explorer** > **Globals** page of the Management Portal.

4.2 Configuring Journaling

There are a number of factors to consider in planning and configuring journaling. Topics in this section include the following:

- Enabling Journaling
- Journal File Naming
- Journaling Best Practices
- Configuring Journal Settings

4.2.1 Enabling Journaling

By default, journaling is enabled for the Caché databases CACHESYS, CACHEAUDIT, and USER. You can enable or disable journaling on each database from the **Local Databases** page of the Management Portal (**System Administration** > **Configuration** > **System Configuration** > **Local Databases**). Click **Edit** on the row corresponding to the database and click **Yes** or **No** in the **Global Journal State** box.

The default setting of the journal state for new databases is *Yes*. When you first mount a database from an earlier release of Caché, the value is set to *Yes*, regardless of the previous setting for new globals and regardless of the previous settings of individual globals within that database.

You can change the global journal setting for a database on a running system. If you do this, Caché warns you of the potential consequences and audits the change if auditing is enabled.

4.2.2 Journal File Naming

A journal file's name consists of an optional user-defined prefix, a base name consisting of the date and time it is created in the format *yyyymmdd.nnn*, and a suffix *nnn* used to incrementally number the journal files created during one calendar day. When a journal file fills, the system automatically switches to a new one with the same prefix and base name but with the suffic increased by one. The base name changes only if a new calendar day begins while the journal file is in use.

For example, if the first journal file that is active on April 27, 2014 is named 20140427.001. When it fills, the system starts a new one called 20140427.002. If midnight passes and the date changes while the journal file is in use, however, it is renamed 20140428.001.

4.2.3 Journaling Best Practices

The following are some important points to consider when planning and configuring journaling:

- Journal files are stored in both a primary journal directory and an alternate journal directory (for use if the primary directory becomes unwriteable for any reason).

  In the interests of both performance and recoverability, InterSystems recommends placing the primary and alternate journal directories on storage devices that are separated from the devices used by databases and the write image journal (WIJ), as well as separated from each other. For practical reasons, these different devices may be different logical unit numbers (LUNs) on the same storage area network (SAN), but the general rule is the more separation the better, with separate sets of physical drives highly recommended. The major benefits of this separation between database/WIJ storage and primary and alternate journal storage include the following:

  – Isolating journal directories from failures that may compromise the databases or WIJ ensures that journal files will be available for use in restoring the database after such a failure.

  – Separating primary and alternate journal directories ensures that when an outage occurs on the device on which the primary directory is located, journaling can continue.

  – Separating journal I/O paths is a key factor in achieving the I/O concurrency that most applications require.

  For simplicity and convenience, Caché installation creates the directory *install-dir*\Mgr\journal, configures it as both the primary and alternate journal directory, and creates in it the first journal file for the default journaled databases. InterSystems recommends, however, that you identify and prepare separate storage devices for the primary and alternate journal directories and reconfigure these settings (as described in Configuring Journal Settings) as soon as possible after installation.

  Note: Journal files should always be backed up along with database files, as described in the "Backup and Restore" chapter of this guide. Consider replicating journal files offsite for disaster recovery purposes, enabling recovery from a failure involving multiple storage devices at the primary data center. Caché mirroring, Caché shadowing, disk-level replication or file system shadowing can be used for this purpose.

- Verify that journaling is enabled for all databases (other than those that contain only transient data).

  Important: Be sure to read Consequences of Not Journaling Databases for important information about limits to the recovery of non-journaled databases.

- Consider setting the journal **Freeze on error** option to *Yes*. If a failure causes the system to be unable to write to both the primary and the alternate journal devices, this setting causes the system to freeze, making it unavailable to users and ensuring that no data is lost. Alternatively, you can set **Freeze on error** option to *No*, which lets the system continue and leads to journaling being disabled, keeping the system available but compromising data integrity and recoverability. See Journal I/O Errors for more information about **Freeze on error**.

- Do not purge a journal file unless it was closed prior to the last known good backup, as determined by the backup validation procedure. Set the number of days and the number of successful backups after which to keep journal files appropriately.
- To ensure optimal performance during a journal restore, consider increasing the size of the generic memory heap (**gmheap**); see Restore Journal Files for more information.

4.2.4 Configuring Journal Settings

To configure Caché journaling, navigate to the **System Administration** > **Configuration** > **System Configuration** > **Journal Settings** page of the Management Portal.

You can edit the following settings:

- **Primary journal directory** — Enter the name of a directory in which to store the journal files. The directory name may be up to 214 characters long.
- **Secondary journal directory** — Enter the name of an alternate directory for journaling to use if the current directory becomes unwritable for any reason. (You can also manually switch journal directories, as described in Switch Journal Directories.) The directory name may be up to 214 characters long.

  Important: InterSystems recommends placing the primary and alternate journal directories on storage devices that are separated from the devices used by databases and the write image journal (WIJ), as well as separated from each other; see Journaling Best Practices for more information.

- **Start new journal file every** — Enter the number of megabytes for the maximum size of the journal file after which the journal file switches. The default size is 1024 MB; the maximum size is 4079 MB.
- **Journal File Prefix** (optional) — Enter an alphanumeric prefix for journal file names.
- **When to purge journal files** — You can set either or both of the following two options. If you enter nonzero values for both settings, purging occurs when a journal file meets whichever of the two conditions occurs first. If you set `0` (zero) for one and not the other, purging is determined by the nonzero setting. If both are `0`, the automatic purging of journal files (and journal history) is disabled.
  - **After this many days** — Enter the number of days after which to purge (valid values: 0-100).

    Note: When you set the number of days after which to purge journal files, the last journal file from the day before the purge limit is also retained. For example, if **After this many days** is set to `1` and the purge is run at midnight on April 1, the last journal file created on March 30 is retained along with those created on March 31.

  - **After this many successive successful backups** — Enter the number of consecutive successful backups after which to purge (valid values: 0-10).

    This includes Caché online backups, an external backup using **$$BACKUP^DBACK("","E")**, or the use of the **Backup.General.ExternalSetHistory()** method to add to the backup history.

  Note: If **After this many days** is set to `0` (no time-based purge) and **After this many successive successful backups** is set to `1`, journal files are not purged until there have been two successful backups; that is, there must be two successful backups for the "successive" criterion to be met.

  You can also update these settings using the **^JRNOPTS** routine or by selecting option 7, *Edit Journal Properties*, from the **^JOURNAL** routine menu. See Update Journal Settings Using ^JRNOPTS for details.

Note: Journal files are sometimes retained even if they meet the criteria of the purge setting. When this happens, the event is recorded in the console log and the reason (for example, that the journal file contains open transactions) is provided.

- **Freeze on error** — Select Yes or No. This setting controls the behavior when an error occurs in writing to the journal. The default is No. See the Journal I/O Errors section for a detailed explanation of this setting.

  Note: When a Caché instance is the primary failover member of a mirror (see the "Mirroring" chapter of the *Caché High Availability Guide*), the instance's **Freeze on error** configuration is automatically overridden to freeze all journaled global updates when a journal I/O error occurs, regardless of the current setting. If the current setting is No, behavior reverts to this setting when the instance is no longer a primary failover member.

- **Journal CSP Session** — Select Yes or No. This setting controls whether or not Caché Server Page (CSP) session journaling is enabled. The default is No.
- **Write image journal entry** — Enter the location of the write image journal (WIJ) file. See the Write Image Journaling section of the "Write Image Journaling and Recovery" chapter of this guide for a detailed explanation of this setting.
- **Target size for the wij (MB) (0=not set)** — Enter the target size for the WIJ file.

Note: All of the settings on this page are included in the instance's cache.cpf file. For information about the journal settings, see [Journal] in the *Caché Parameter File Reference*; for information about the WIJ settings, see Write Image Journal (WIJ) File in the "Write Image Journaling and Recovery" chapter of this guide, as well as targwijsiz and wijdir in the [config] section of the *Caché Parameter File Reference*).

You are not required to restart Caché after changing most of these settings (except where indicated), but any change causes a new journal file to begin.

There are two additional configuration settings affecting journaling, as follows:

- **jrnbufs** — Specifies the amount of memory allocated to journal buffers; the default is 64 MB, the maximum is 1024 MB, and the minimum is 16 MB for Unicode instances and 8 MB for 8-bit instances. Increasing this setting means increasing the amount of journal data that can be held in memory, which improves journaling performance, but increases the maximum amount of journal data that could be lost in the event of a system failure because it was written to the buffer after the last journal sync (see The Journal Write Cycle).

  To change the **jrnbufs** setting, navigate to the **Advanced Memory Settings** page of the management portal (**System Administration** > **Configuration** > **Additional Settings** > **Advanced memory**). The **jrnbufs** setting can also be changed by editing the cache.cpf file; for more information, see jrnbufs in the *Caché Parameter File Reference*.

- **SynchCommit** — Specifies when the **TCOMMIT** command requests that journal data involved in a transaction be flushed to disk: when this setting is **true**, **TCOMMIT** does not complete until the journal data write operation completes; when it is **false** (the default), **TCOMMIT** does not wait for the write operation to complete.

  To change the **SynchCommit** setting, navigate to the **Compatibility Settings** page of the management portal (**System Administration** > **Configuration** > **Additional Settings** > **Compatibility**). For more information on **SynchCommit**, see SynchCommit in *Caché Additional Configuration Settings Reference* and **TCOMMIT** in *Caché ObjectScript Reference*.

4.3 Journaling Operation Tasks

Once journaling is configured there are several tasks you can perform:

- Start journaling

- Stop journaling
- View journal files
- Switch journal files
- Switch journal directories
- Display journal file profiles
- Check journal file integrity
- View journal file summaries
- Purge journal files
- Restore journal files

4.3.1 Start Journaling

If journaling is stopped, you can start it using the **^JRNSTART** routine or by selecting option 1, *Begin Journaling*, from the **^JOURNAL** routine menu. See Start Journaling Using ^JRNSTART for details.

Note: You cannot start journaling from the Management Portal.

When you start journaling, Caché audits the change if auditing is enabled.

4.3.2 Stop Journaling

Stopping journaling system wide has a number of generally undesirable consequences, as described in the Journal Freeze on Error Setting is No section. Both shadowing and transaction processing are affected.

When you stop journaling, transaction processing ceases. If a transaction is in progress when you stop journaling, the complete transaction may not be entered in the journal. To avoid this problem, it is best to make sure all users are off the system before stopping journaling.

If you stop journaling and Caché crashes, the startup recovery process does not roll back incomplete transactions started before journaling stopped since the transaction may have been committed but not journaled.

In contrast, transactions are not affected in any adverse way by switching journal files. Rollback correctly handles transactions spanning multiple journal files created by journal switching; so, if possible, it is better to switch journal files than to stop journaling.

You can stop journaling using the **^JRNSTOP** routine or by selecting option 2, *Stop Journaling*, from the **^JOURNAL** routine menu. See Stop Journaling Using ^JRNSTOP for details.

Note: You cannot stop journaling from the Management Portal.

When you stop journaling, Caché audits the change if auditing is enabled.

4.3.3 View Journal Files

You can view a journal file on the **Home** > **Journals** > **View Journal** page of the Management Portal.

1. Click **Journals** from the **System Operations** menu of the home page to list the instance's journal files. Use the **Filter** box to shorten the list if necessary.
2. If the instance is configured as a mirror member, all journal files including mirrored and nonmirrored are displayed by default. Optionally click the link containing the mirror name, for example **Mirror Journal Files Of 'MUNDANE'**, to

display a list of mirror journal files only. If the instance is configured as a reporting async member of multiple mirrors, there is a separate link for the journal files from each mirror. To return to displaying all journal files, click the **All Journal Files** link.

Note: For information about mirror journal files, see Journal Files and Journal History Log in this chapter and Mirror Synchronization in the "Mirroring" chapter of the *Caché High Availability Guide*.

3. To view a journal file, click **View** in the row of the journal file you want to see.
4. The journal file is displayed record by record on the **System Operation** > **Journals** > **View Journal** page. You can:
   a. Click in the **Offset** column of a record to view a dialog box containing its details.
   b. Choose whether to color code the records by the time of entry, the process that performed the operation recorded in the journal, the type of operation, whether the operation was part of a transaction, the global involved in the operation, or the database involved in the operation.

      Note: The Journal File Operations table in the section Display Journal Records Using ^JRNDUMP provides information about the values that appear in the **Type** column of the display to indicate the type of operation represented by each record.

   c. Search for a particular record set of records using the **Match** boxes and the **Search** button.
      1. For a manual search, set the first drop-down to the column you want to search by, select an operator such as "equal to" or "not equal to", and enter the value you want to match in the right-most box, then click **Search**.
      2. To match a particular cell in one of the columns, just double-click in that cell. For example, to find all journal records containing KILL operations, double-click in any cell in the **Type** column containing **KILL**. The operator drop-down is automatically set to "equal to" but you can change that before pressing **Search**.

You can also use the **^JRNDUMP** utility to display the entire journal and the **SELECT^JRNDUMP** entry point to display selected entries. See Display Journal Records Using ^JRNDUMP for details.

4.3.4 Switch Journal Files

The system automatically switches the journal file in the following situations:

- After a successful backup of a Caché database.
- When the current journal file grows to the maximum file size allowed (configurable on the Journal Settings page).
- When the journal directory becomes unavailable and you specified an alternate directory.
- After updating settings on the Journal Settings page.

Switching the journal file is preferable to stopping and starting journaling because during the latter process, any global operations that occur after stopping but before restarting are not journaled.

To manually switch journal files:

1. Navigate to the **System Operation** > **Journals** page of the Management Portal.
2. Click **Switch Journal** above the list of database journal files.
3. Confirm the journal switch by clicking **OK**.

You can also switch journal files using the **^JRNSWTCH** routine or by selecting option 3, *Switch Journal File* from the **^JOURNAL** routine menu. See Switch Journal Files Using ^JRNSWTCH for details.

4.3.5 Switch Journal Directories

As described in Configuring Journal Settings, journaling automatically switches to the secondary journaling directory (assuming it is configured) if the primary directory becomes unwritable for any reason. To manually switch journaling directories, do the following:

1. Navigate to the **System Operation** > **Journals** page of the Management Portal.
2. Click **Switch Directory** above the list of database journal files.
3. Confirm the journal switch by clicking **OK**.

You can also switch journal directories by selecting option 13, *Switch Journaling to Secondary/Primary Directory* from the **^JOURNAL** routine menu. See Switch Journaling Directories Using SWDIR^JOURNAL for details.

4.3.6 Display Journal File Profiles

You can display the global profile of a journal file, showing the globals that appear in the file's records and the number of records each appears in, on the Journal Profile page (**Home** > **Journals** > **Journal Profile**).

1. Click **Journals** from the **System Operations** menu of the home page to list the instance's journal files. Use the **Filter** box to shorten the list if necessary.
2. To display a journal file profile, click **Profile** in the row of the appropriate journal file. The Journal Profile page displays with the profile on it. If the journal file has a large number of records, it may take a little while to build the profile.
3. You can sort the journal profile by global or by the cumulative size, in bytes, of all the records in which each global appears.
4. If the journal file is the current one, you can use the **Recalculate** button build the profile again after some time has passed.

4.3.7 Check Journal File Integrity

You can check the integrity of a journal file on the Journals page. This operation verifies that the journal file ends where it is expected to end, which verifies that there are no records missing from the end of the file.

1. Click **Journals** from the **System Operations** menu of the home page to list the instance's journal files. Use the **Filter** box to shorten the list if necessary.
2. To run an integrity check on a journal file, click **Integrity Check** in the row of the appropriate journal file. The Journal Integrity Check page displays.
3. Select **Check Details** to scan the journal file record by record from the beginning to detect potential missing records.
4. Once you have clicked **OK**, a link to the **System Operation** > **Background Tasks** page appears, letting you view the status and results of the integrity check.

4.3.8 View Journal File Summaries

You can view information about a journal file on the Journal Profile page. For example, you can find out whether the journal file is encrypted, and what databases are affected by the operations recorded in the journal file.

1. Click **Journals** from the **System Operations** menu of the homepage to list the instance's journal files. Use the **Filter** box to shorten the list if necessary.

2. To view information about a journal file, click **Summary** in the row of the appropriate journal file. The Journal File Summary page displays.

4.3.9 Purge Journal Files

You can schedule a task to run regularly that purges obsolete journal files. A new Caché instance contains a pre-scheduled *Purge Journal* task that is scheduled to run after the daily *Switch Journal* task that runs at midnight. For information about purging mirror journal files, see Purging Mirror Journal Files.

The purge process deletes journal files based on the **When to purge journal files** setting on the Journal Settings page; for information, see Configure Journal Settings in this chapter.

Note: Journal files are sometimes retained even if they meet the criteria of the purge setting. When this happens, the event is recorded in the console log and the reason (for example, that the journal file contains open transactions) is provided.

You can also purge journal files using the **PURGE^JOURNAL** routine or by selecting option 6, *Purge Journal Files* from the **^JOURNAL** routine menu. See Purge Journal Files Using PURGE^JOURNAL for details.

Note: The configured journal purge settings can be overridden by the **%ZJRNPURGE** routine; for more information, contact the InterSystems Worldwide Response Center (WRC).

4.3.10 Purging Mirror Journal Files

Mirror journal files are subject to additional purge criteria because they must be successfully distributed by the primary failover member to the other mirror members and dejournaled on each to synchronize the mirrored databases (see Mirror Synchronization in the "Mirroring" chapter of the *Caché High Availability Guide* for a full description of this process). Transmission of the files to the backup is synchronous and always rapid when the mirror is operating normally, but transmission to asynchronous (async) members may take longer and may be delayed when an async is disconnected from the mirror. Backup and DR async members must also follow the same policy as the primary, since they are eligible to become primary in failover or disaster recovery situations. Mirror journal files are therefore purged as follows:

- On the primary failover member, a file is purged when the local journal file purge criteria have been met (see Configure Journal Settings) and when it has been received by the backup (if there is one) and all async members, whichever takes longer. If an async has been disconnected from the mirror for more than 14 days, however, files are purged even if that async has not yet received them.
- On the backup failover member (if there is one) and any disaster recovery (DR) async members, a file is purged when it has been fully dejournaled on that member, when local journal file purge criteria have been met, and when it has been received by all async members, with the same exception for asyncs that have been disconnected for more than 14 days.
- On reporting async members, mirror journal files are purged immediately after they have been dejournaled by default, to ensure that async mirror members do not run out of space (particularly when they are receiving journal files from multiple mirrors). You can optionally configure a reporting async to instead retain the files and purge them according to local journal file purge criteria; see Editing or Removing an Async Member in the "Mirroring" chapter.

No mirror journal file containing a currently open transaction is ever purged on any mirror member.

Note: When a mirror journal file is retained longer than would be dictated by local journal file purge criteria, this is recorded in the member's console log and the reason is provided.

You can modify the defaults for purging mirror journal files with the **SYS.Mirror.JrnPurgeDefaultWait()** method.

4.3.11 Restore Journal Files

After a system crash or disk hardware failure, recreate your database by restoring your backup copies. If you have been journaling and your journal file is still accessible, you can further restore your databases by applying changes since the last backup recorded in the journal files to the databases.

To restore the journal files:

1. First confirm that all users exit Caché.
2. Stop journaling if it is enabled.
3. Restore the latest backup of your database. See the "Backup and Restore" chapter of this guide for more information.
4. Run the journal restore utility. See the Restore Globals From Journal Files Using ^JRNRESTO section for details.
5. Restart journaling if it is disabled.

Note: You cannot run the journal restore process from the Management Portal.

4.4 Journaling Utilities

Caché provides several utilities to perform journaling tasks. The **^JOURNAL** utility provides menu choices to run some common journaling utilities, which you can also run independently. There are also several other journaling utilities, which you run from the %SYS namespace.

The following sections describe the journaling utilities in detail:

- Perform Journaling Tasks Using ^JOURNAL
- Recover from Startup Errors Using ^STURECOV
- Convert Journal Files Using ^JCONVERT and ^%JREAD
- Set Journal Markers Using ^JRNMARK
- Manipulate Journal Files Using ^JRNUTIL
- Manage Journaling at the Process Level Using %NOJRN

In the following sections the sample procedures show C:\MyCache as the Caché installation directory.

4.4.1 Perform Journaling Tasks Using ^JOURNAL

The following example shows the menu available by invoking the ^**JOURNAL** routine; the full menu is not repeated in subsequent examples.

```
%SYS>Do ^JOURNAL

 1) Begin Journaling (^JRNSTART)
 2) Stop Journaling (^JRNSTOP)
 3) Switch Journal File (^JRNSWTCH)
 4) Restore Globals From Journal (^JRNRESTO)
 5) Display Journal File (^JRNDUMP)
 6) Purge Journal Files (PURGE^JOURNAL)
 7) Edit Journal Properties (^JRNOPTS)
 8) Activate or Deactivate Journal Encryption (ENCRYPT^JOURNAL())
 9) Display Journal status (Status^JOURNAL)
10) -not available-
11) -not available-
12) Journal catch-up for mirrored databases (MirrorCatchup^JRNRESTO)
13) Switch Journaling to Secondary Directory (SWDIR^JOURNAL)

Option?
```

Note: The `-not available-` text for options 10 and 11 is replaced as follows:

- Option `10) Cluster Journal Restore (CLUMENU^JRNRESTO)` is displayed only on a cluster node; for information, see Restore Cluster Journal Using CLUMENU^JRNRESTO in this section.
- Option `11) Manage pending or in progress transaction rollback (Manage^JRNROLL)` is displayed if pending or in-progress transaction rollbacks are encountered when you run the ^**STURECOV** (at system startup) or ^**MIRROR** (at primary mirror member startup) routine; for more information, see Manage Transaction Rollback Using Manage^JRNROLL in this section.

Enter the appropriate menu number option to start that particular routine. Press **Enter** without entering an option number to exit the utility. The following subsections describe the options available through the ^JOURNAL utility:

- Start Journaling Using ^JRNSTART
- Stop Journaling Using ^JRNSTOP
- Switch Journal Files Using ^JRNSWTCH
- Switch Journaling Directories Using SWDIR^JOURNAL
- Restore Globals From Journal Files Using ^JRNRESTO
- Display Journal Records Using ^JRNDUMP
- Purge Journal Files Using PURGE^JOURNAL
- Update Journal Settings Using ^JRNOPTS
- Journal Encryption Using ENCRYPT^JOURNAL
- Display Journal Status Using Status^JOURNAL
- Restore Cluster Journal Using CLUMENU^JRNRESTO
- Manage Transaction Rollback Using Manage^JRNROLL
- Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO

4.4.1.1 Start Journaling Using ^JRNSTART

To start journaling, run ^**JRNSTART** or enter `1` at the `Option` prompt of the ^**JOURNAL** menu, as shown in the following examples.

Example of running ^**JRNSTART** directly:

```
%SYS>Do ^JRNSTART
```

Example of starting journaling from the ^**JOURNAL** menu:

```
%SYS>Do ^JOURNAL

 1) Begin Journaling (^JRNSTART)
 ...
Option? 1
```

If journaling is running when you select this option, a message similar to the following is displayed:

```
Already journaling to C:\MyCache\mgr\journal\20151113.001
```

4.4.1.2 Stop Journaling Using ^JRNSTOP

To stop journaling, run ^**JRNSTOP** or enter `2` at the `Option` prompt of the ^**JOURNAL** menu, as shown in the following examples.

Note: When the **Freeze on error** flag (see Configure Journal Settings in this chapter) is set to "Yes," stopping journaling is allowed (although there is a risk of data loss) and does not cause the instance to freeze.

Example of running ^**JRNSTOP** directly:

```
%SYS>Do ^JRNSTOP

Stop journaling now? No => Yes
```

Example of stopping journaling from the ^**JOURNAL** menu:

```
%SYS>Do ^JOURNAL

 ...
 2) Stop Journaling (^JRNSTOP)
 ...
Option? 2
Stop journaling now? No => Yes
```

If journaling is not running when you select this option, you see a message similar to the following:

```
Not journaling now.
```

4.4.1.3 Switch Journal Files Using ^JRNSWTCH

To switch the journal file, run ^**JRNSWTCH** or enter `3` at the `Option` prompt of the ^**JOURNAL** menu, as shown in the following example:

```
%SYS>Do ^JOURNAL

 ...
 3) Switch Journal File (^JRNSWTCH)
 ...
Option? 3
Switching from: C:\MyCache\mgr\journal\20151113.002
To:             C:\MyCache\mgr\journal\20151113.003
```

The utility displays the name of the previous and current journal files.

4.4.1.4 Switch Journaling Directories Using SWDIR^JOURNAL

To switch journaling directories, assuming a secondary directory is configured as described in Configuring Journal Settings, run **SWDIR^JOURNAL** or enter `13` at the `Option` prompt of the **^JOURNAL** menu, as shown in the following example:

```
%SYS>Do ^JOURNAL

 ...
13) Switch Journaling to Secondary Directory (SWDIR^JOURNAL)

Option? 3
Option? 13
Journaling to \\remote\MyCache\journal_secondary\MIRROR-MIRRORONE-20150720.007
```

The utility displays the name of the current journaling directory and journal file following the switch.

4.4.1.5 Restore Globals From Journal Files Using ^JRNRESTO

The Caché **^JRNRESTO** routine is used after a database is restored from backup to return it to its state immediately prior to a failure by applying updates from journal files. This is called a journal restore, and the process of applying the changes is called dejournaling. A journal restore dejournals all journal records created between the creation of the backup and the failure. For example, if the database was backed up early Tuesday morning and crashed on Wednesday afternoon, after you restore the Tuesday backup, you can restore updates from the journal files created on Tuesday and Wednesday.

If there are sufficient computing and memory resources available, and you do not choose either to abort the operation due to both database-related problems and journal-related problems or to enable journaling of the updates during the restore, up to four jobs can perform the updates to separate databases in parallel within a journal restore operation. This is called *parallel dejournaling* and increases the performance of the operation.

Parallel dejournaling is used only when the host system has at least eight CPUs and the Caché instance involved has enough generic memory available to allocate for this purpose. In practice, parallel dejournaling will not be used in journal restores on most Caché instances unless generic memory is increased. The number of parallel dejournaling jobs can never exceed the size of the generic memory heap divided by 200; for example, to support four dejournaling jobs running in parallel, the generic heap must be greater than or equal to 800 MB. (Even if you do not have enough memory available to support parallel dejournaling, dejournaling throughput may improve if you increase the size of the generic memory heap from the default.)

Note: To change the size of the generic memory heap or gmheap (sometimes known as the shared memory heap or SMH), navigate to the **Advanced Memory Setting** page (**System Administration** > **Configuration** > **Additional Settings** > **Advanced Memory**); see Advanced Memory Settings in the "Caché Additional Configuration Settings" chapter of the *Caché Additional Configuration Settings Reference* for more information.

Parallel dejournaling is also used by Caché mirroring; for more information, see Configuring Parallel Dejournaling in the "Mirroring" chapter of the *Caché High Availability Guide*.

^JRNRESTO restores only to databases whose journal state is *Yes* at the time of the journal restore. The first time it encounters each database, the routine checks and records its journal state. The restore process skips journal records for databases whose journal state is *No*. If no databases are marked as being journaled, the routine asks if you wish to terminate the restore; you can then change the database journal state to *Yes* on specific databases and restart **^JRNRESTO**.

Note: The journal state of a database *at the time of restore* determines what action is taken; Caché stores nothing in the journal about the current journal state of the database when a given journal record is written. This means that changes to databases whose journal state is *Yes* are durable, but changes to other databases may not be. Caché ensures physical consistency, but not necessarily application consistency, if transactions involve databases whose journal state is *No*.

^JRNRESTO lets you make several decisions about the journal restore. Using **^JRNRESTO**, you can do the following:

- Restore global updates to databases in the Caché instance in which you are running the routine, or to databases in another Caché instance. You can choose to restore updates for all globals to all databases in the current instance, or to select individual databases in the current or another instance and optionally specify which globals to restore to each.
- Restore mirror journal files to a mirrored database (catch up a mirrored database) or to a non-mirrored database. On a mirror member, you are prompted to indicate whether you are catching up a mirrored database, as noted in the following procedure; if so, the procedure is redirected to the **MirrorCatchup^** entry point to **^JRNRESTO** (see Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO).
- Apply existing journal filters (see Filter Journal Records Using ^ZJRNFILT) to the restore.
- Select a range of journal files to restore from.
- Disable journaling of updates during the restore to make the operation faster.

CAUTION: If you use journal restore scripts based on prompts, you should update the scripts because some prompts may have changed since the last release.

To restore global updates from journal files:

1. Run the **^JRNRESTO** routine in the %SYS namespace, then press **<Enter>** at the `Restore the Journal?` prompt to continue.
2. If you are running the routine on a mirror member, the following prompt is displayed.

   ```
   Catch-up mirrored databases? No =>
   ```

   - If you are restoring mirror journal files to a mirrored database *in the same mirror in which the mirror journal files were created*, enter `yes`; the procedure is redirected to the **MirrorCatchup^** entry point to **^JRNRESTO** (see Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO).
   - If you are restoring mirror journal files to a non-mirrored database, or are not restoring mirror journal files, enter `no` or **<Enter>** and continue to use the procedure described here.

3. If you have existing journal filters (see Filter Journal Records Using ^ZJRNFILT), specify whether you want to use them:

   ```
   Use current journal filter (ZJRNFILT)?
   Use journal marker filter (MARKER^ZJRNFILT)?
   ```

4. Choose whether you want to restore all journaled globals to all databases in the current Caché instance, or to specify one or more databases and optionally specify which globals to restore to each.

   ```
   Process all journaled globals in all directories?
   ```

   - Enter `Yes` if you want to restore all globals to all databases in the current instance.
   - Enter `No` or **<Enter>** if you want to restore only selected databases in the current or another instance. Then do the following:
     - Indicate whether the journal files were created under a different operating system from that of the current system. This is important because the directory paths you specify for the databases you want to restore must exactly match the paths in the journal files, which are in canonical form. If you respond with `No`, **^JRNRESTO** puts the directory paths you enter into canonical form for the current operating system so they will match those in the journal files. If you respond with `Yes`, **^JRNRESTO** does not canonicalize the paths you enter, because the canonical form in the journal files is different from the canonical form on the current system. In the latter case, you must take care to enter directory paths in the canonical form appropriate to the operating system of the journal files to ensure that they will match.

For example:

- if you are working on a Windows system and enter `No` at this prompt, then enter the path `c:\intersystems\cache\mgr\user`, **^JRNRESTO** automatically canonicalizes this to **c:\intersystems\cache\user** to match journal files created on a Windows system.
- if you are working on a Unix system and enter `Yes` at this prompt because the journal files were created on a Windows system, you must be sure to enter the canonical form of the path, `c:\intersystems\cache\mgr\user\`, to ensure that it matches the journal files, because **^JRNRESTO** cannot canonicalize it for you.

– Specify each database you want to restore by entering its directory path; this indicates the source database from which the journal records were taken. Press **<Enter>** at the `Redirect to directory` prompt to indicate that source and target are the same and restore global updates to the source database. If you are restoring to a different database, for example because you have restored the source database from backup to a different system, enter the directory path of the target database.

  If you are restoring mirror journal files to a non-mirrored database, at the `Directory to restore` prompt, you can do either of the following:

  - Enter directory path of the source database and then either **<Enter>** or the directory path of the target non-mirrored database, as described in the foregoing.
  - Enter the full, case-sensitive mirror database name of the source mirrored database, for example, `:mirror:JLAP:MIRRORDB`, which can be found using the **List mirrored databases** option on the **Mirror Status** menu of the **^MIRROR** utility, and then specify the directory path of the target non-mirrored database.

  Note: If you are restoring mirror journal files to a mirrored database, you will not have reached this point in the procedure; see Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO.

– For each database you specify, either confirm that you want to restore updates for all globals or enter one or more globals to restore.

– When you have entered all the databases, press **<Enter>** at the `Directory to restore` prompt, then confirm the list of specified databases and globals.

For example:

```
Process all journaled globals in all directories? no
Are journal files imported from a different operating system? No => No

Directory to restore [? for help]: c:\intersystems\cache23\mgr\user\
Redirect to Directory: c:\intersystems\cache23\mgr\user\
 => --> c:\intersystems\cache23\mgr\user\
Process all globals in c:\intersystems\cache23\mgr\user\? No => yes

Directory to restore [? for help]: c:\intersystems\cache23\mgr\samples\
 Redirect to Directory: c:\intersystems\cache23\mgr\samples\
 => --> c:\intersystems\cache23\mgr\samples\
Process all globals in c:\intersystems\cache23\mgr\samples\? No => no

Global ^Aviation.AircraftD
Global^

Directory to restore [? for help]:

Processing globals from the following datasets:
 1. c:\intersystems\cache23\mgr\user\   All Globals
 2. c:\intersystems\cache23\mgr\Samples\   Selected Globals:
          ^Aviation.AircraftD

Specifications correct? Yes =>  Yes
```

Note: If you are redirecting two or more databases to the same directory, you must make the same global selection — that is, either enter **yes** to process all globals, or **no** and then the same list of globals to process – for all of these databases. If you try to restore multiple databases to a single directory and the global selections are not all the same, the utility gives you the opportunity to either change your database redirection and global selections or cancel the operation.

5. Specify the journal files to restore from, which should be from the same Caché instance as the source databases you are restoring, by specifying the correct journal history log (see Journal History Log in this chapter).

```
Are journal files created by this Cache instance and located in their original
paths? (Uses journal.log to locate journals)?
```

   - Enter `yes` or **<Enter>** at the prompt to use the journal history log of the current Caché instance to identify the journal files to process. For example, if you entered `yes` at the `Process all journaled globals in all directories?` prompt at the start of the process, enter `yes` here to restore all databases in the current instance from the current instance's journal files.
   - If you entered `no` at the `Process all journaled globals in all directories?` prompt and then specified databases in another Caché instance, enter `no` here to specify the journal history log and journal file directory path of that instance, or files copied from that instance, so that the databases can be restored from that instance's journal files.

     Important: If you are using a journal history log from another Caché instance, you must use a copy of the file, not the actual log.

     For example,

```
Are journal files created by this Cache instance and located in their original
paths? (Uses journal.log to locate journals)? no
If you have a copy of the journal history log file from the Cache
instance where the journal files were created, enter its full path below;
otherwise, press ENTER and continue.
Journal history log: c:\cache23_journals\journal.log

Specify the location of the journal files to be processed
Directory of the journal files:  c:\cache23_journals\journal\
Directory of the journal files:
```

6. Specify the range of journal files you want to process. Bear in mind the following:

   - If Caché switched to multiple journal files since the restored backup, you must restore the journal files in order from the oldest to the most recent. For example, if you have three journal files to restore, 20130214.001, 20130214.002, and 20130215.001, you must restore them in the following order:

     20130214.001

     20130214.002

     20130215.001

   - When you back up with Caché online backup, information about the oldest journal file required for transaction rollback during restore is displayed at the beginning of the third and final pass and stored in the backup log. See the "Backup and Restore" chapter of this guide for more information.

   Respond to the prompts as follows:

   - If you entered `yes` at the `Are journal files created by this Cache instance` prompt, or answered `no` and then specified the journal history log and journal file location of another instance, you can enter the path-

names of the first and last journal files to process. You can also enter ? at either prompt to see a numbered list of the files in the specified location, then enter the numbers of the files, for example:

```
Specify range of files to process
Enter ? for a list of journal files to select the first and last files from
First file to process:  ?

1) c:\intersystems\cache2\mgr\journal\20130212.001
2) c:\intersystems\cache2\mgr\journal\20130213.001
3) c:\intersystems\cache2\mgr\journal\20130214.001
4) c:\intersystems\cache2\mgr\journal\20130214.002
5) c:\intersystems\cache2\mgr\journal\20130215.001
6) c:\intersystems\cache2\mgr\journal\20130216.001
7) c:\intersystems\cache2\mgr\journal\20130217.001
8) c:\intersystems\cache2\mgr\journal\20130217.002

First file to process:  5 c:\intersystems\cache2\mgr\journal\20130215.001
Final file to process:
  c:\intersystems\20141316mar14\mgr\journal\20130217.002 =>

Prompt for name of the next file to process? No => no
```

- If you entered no at the `Are journal files created by this Cache instance` prompt and did not specify a journal history log, processing continues with prompts that attempt to identify the specific journal files you want to process. For example:

```
Journal history log:
Specify range of files to process (names in YYYYMMDD.NNN format)

from:     <20130212.001> [?] => 20130215.001

through:  <20130217.001> [?] => 20130217.002

Prompt for name of the next file to process? No => no

Provide or confirm the following configuration settings:

Journal File Prefix: [?] =>

Files to dejournal will be looked for in:
     c:\intersystems\cache\mgr\journal\
in addition to any directories you are going to specify below, UNLESS
you enter a minus sign ('-' without quotes) at the prompt below,
in which case ONLY directories given subsequently will be searched

Directory to search: {return when done} -
     [Directory search list is emptied]
Directory to search: {return when done} c:\intersystems\cache2\mgr\journal
Directory to search: {return when done}
Here is a list of directories in the order they will be searched for files:
     c:\intersystems\cache2\mgr\journal\
```

7. Process the journal files:

```
Prompt for name of the next file to process? No => No
The following actions will be performed if you answer YES below:

* Listing journal files in the order they will be processed
* Checking for any missing journal file on the list ("a broken chain")

The basic assumption is that the files to be processed are all
currently accessible. If that is not the case, e.g., if you plan to
load journal files from tapes on demand, you should answer NO below.
Check for missing journal files? Yes => Yes
```

8. If one or more journal files within the range you specified are missing, you are given the opportunity to abort the operation. If you do not, or if no files are missing, the process proceeds with an opportunity to check journal integrity before starting the restore:

```
Journal files in the order they will be processed:
1. c:\intersystems\cache2\mgr\journal\20130215.001
2. c:\intersystems\cache2\mgr\journal\20130216.001
3. c:\intersystems\cache2\mgr\journal\20130217.001
4. c:\intersystems\cache2\mgr\journal\20130217.002

While the actual journal restore will detect a journal integrity problem
when running into it, you have the option to check the integrity now
before performing the journal restore. The integrity checker works by
scanning journal files, which may take a while depending on file sizes.
Check journal integrity? No => No
```

9. If the current journal file is included in the restore, you must switch journaling to another file, and are prompted to do so:

```
The journal restore includes the current journal file.
You cannot do that unless you stop journaling or switch
     journaling to another file.
Do you want to switch journaling? Yes => yes
Journaling switched to c:\intersystems\cache2\mgr\journal\20150217.003
```

10. Next, choose whether to disable journaling of updates during the restore to make the operation faster.

```
You may disable journaling of updates for faster restore for all
databases other than mirrored databases. You may not want to do this
if a database to restore is being shadowed as the shadow will not
receive the updates.
Do you want to disable journaling the updates? Yes => yes
Updates will NOT be journaled
```

Important: If you do not disable journaling of updates during the restore, parallel dejournaling will not be used to increase performance, as described at the beginning of this section.

If journaling is disabled but database updates continue, you cannot use the last good journals to do a manual restore unless you can assure either of the following:

- You know exactly what will be updated and can control what is restored to the satisfaction of the application.
- You have restored the database(s) involved from the last backup and accept that after applying the journals you will have lost the data written when journaling was off.

InterSystems recommends that you run the following commands after completing the journal restore under these circumstances to verify that the object IDs are not out of sync; only IDs that are found to be out of sync are reported in the array, *errors*:

```
Do CheckIDCounters^%apiOBJ(.errors)
zwrite errors
```

11. After confirming or changing the default options for the restore job, confirm the restore to begin:

```
Before we job off restore daemons, you may tailor the behavior of a
restore daemon in certain events by choosing from the options below:

     DEFAULT:    Continue despite database-related problems (e.g., a target
     database is not journaled, cannot be mounted, etc.), skipping affected
     updates

     ALTERNATE:  Abort if an update would have to be skipped due to a
     database-related problem (e.g., a target database is not journaled,
     cannot be mounted, etc.)

     DEFAULT:    Abort if an update would have to be skipped due to a
     journal-related problem (e.g., journal corruption, some cases of missing
     journal files, etc.)
```

```
     ALTERNATE:  Continue despite journal-related problems (e.g., journal
     corruption, some missing journal files, etc.), skipping affected updates

Would you like to change the default actions? No => No

Start the restore? Yes =>
```

Important: If you choose to abort due to both database-related problems and journal-related problems, parallel dejournaling will not be used to increase performance, as described at the beginning of this section.

12. The progress of the journal restore is displayed at intervals, and when the job is complete a list of databases updated by the restore is displayed:

```
c:\MyCache1\mgr\journal\20150406.001
 35.73%  70.61% 100.00%
c:\MyCache1\mgr\journal\20150406.002
 35.73%  70.61% 100.00%
c:\MyCache1\mgr\journal\20150406.003
 100.00%
[Journal restore completed at 20150407 02:25:31]

The following databases have been updated:

1. c:\MyCache1\mgr\source22\
2. c:\MyCache1\mgr\source23\
3. c:\MyCache1\mgr\cache\
4. c:\MyCache1\mgr\cachelib\
5. c:\MyCache1\mgr\cachetemp\

The following databases have been skipped:

1. /bench/user/cache/162/
2. /scratch1/user/cache/p750.162/mgr/
3. /scratch1/user/cache/p750.162/mgr/cache/
4. /scratch1/user/cache/p750.162/mgr/cachelib/
5. /scratch1/user/cache/p750.162/mgr/cachetemp/
6. /scratch1/user/cache/p750.162/mgr/user/
```

Rolling Back Incomplete Transactions

Restoring the journal also rolls back incomplete transactions. Ensure that users have completed all transactions so that the restore does not attempt to roll back active processes.

To ensure that transactions are all complete before you restore your backup and clear the journal file, InterSystems strongly recommends the following:

- If you need to roll back transactions for your own process, the process must halt or use the **TROLLBACK** command.
- If you need to roll back transactions system-wide, shut down Caché and restart it to ensure that no users are on the system.

Restoring Mirror Journal Files

You can restore mirror journal files to either mirrored or non-mirrored databases. If you are restoring to a mirrored database, see step 2 of the procedure in Restore Globals From Journal Files Using ^JRNRESTO and the section Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO. If you are restoring to a non-mirrored database, see step 4 of the procedure in Restore Globals From Journal Files Using ^JRNRESTO.

Filter Journal Records Using ^ZJRNFILT

InterSystems provides a journal filter mechanism to manipulate the journal file. The journal filter program is a user-written routine called **^ZJRNFILT** whose format is shown below. This is called by the Caché journal restore program, **^JRNRESTO**, and ensures that only selected records are restored. Create the **^ZJRNFILT** routine using the following format:

```
ZJRNFILT(jid,dir,glo,type,restmode,addr,time)
```

Argument	Type	Description
jid	input	Job ID which you can use to identify the PID that generated the journal
dir	input	Full pathname of the directory containing the CACHE.DAT file to be restored, as specified in the journal record
glo	input	Global in journal record
type	input	Command type in journal record (S for **Set**, K for **Kill**)
addr	input	Address of the journal record
time	input	Time stamp of the record (in **$horolog** format). This is the time the journal buffer is created, not when the **Set** or **Kill** operation occurs, so it represents the earliest this particular operation could have happened.
restmode	output	0 - do not restore record 1 - restore record

^ZJRNFILT Considerations

Consider the following when using ^**ZJRNFILT**:

- If the startup routine (^**STU**) calls ^**JRNRESTO**, it does not call the filter routine under any circumstances.
- Journal restore only calls the journal filter (^**ZJRNFILT**) if it exists. If it does exist, the restore procedure prompts you to confirm the use of the filter in the restore process.
- If you answer yes to use the journal filter, for every record in the journal file to restore, the routine calls the journal filter ^**ZJRNFILT** with the indicated input arguments to determine whether to restore the current record.
- You can use any logic in your ^**ZJRNFILT** routine to determine whether or not to restore the record. Return confirmation through the output *restmode* argument.
- If you are using the directory name, *dir*, in the ^**ZJRNFILT** routine logic, specify the full directory pathname.
- The entire global reference is passed to ^**ZJRNFILT** for use in program logic.
- When the journal restore process completes, it prompts you to confirm whether to rename the ^**ZJRNFILT** routine or delete it. If you choose to rename the filter, the utility renames it ^**XJRNFILT** and deletes the original ^**ZJRNFILT**.
- The restore process aborts with an appropriate error message if any errors occur in the ^**ZJRNFILT** routine.

^ZJRNFILT Examples

Two globals, ^*ABC* and ^*XYZ*, are journaled. While journaling is turned on, the following code is executed, and the journal file records the **Set** and **Kill** operations for these globals:

ObjectScript

```
For I=1:1:500 Set ^ABC(I)=""
For I=1:1:500 Set ^XYZ(I)=""
For I=1:1:100 Kill ^ABC(I)
```

1. To restore all records for ^*ABC* only, the ^**ZJRNFILT** routine looks like this:

   #### ObjectScript

   ```
   ZJRNFILT(jid,dir,glo,type,restmode,addr,time)    /*Filter*/
    Set restmode=1                                  /*Return 1 for restore*/
    If glo["XYZ" Set restmode=0                     /*except when it is ^XYZ*/
    Quit
    ;
   ```

2. To restore all records except the kill on *^ABC*, the **^ZJRNFILT** routine looks like this:

   ObjectScript

```
ZJRNFILT(jid,dir,glo,type,restmode,addr,time)    /*Filter*/
 Set restmode=1                                  /*Return 1 for restore*/
 If glo["^ABC",type="K" Set restmode=0           /*except if a kill on ^ABC*/
 Quit
 ;
```

3. In some cases (for example, when the *jid* is a PID or on a mirror member), *remsysid* is *not* the actual ECP system ID. In these cases, use the **%SYS.Journal.Record.GetRealPIDSYSinFilter** method to return the real ECP system ID as well as the real PID.

   To pull the real PID (and ECP system PID) in a filter, the **^ZJRNFILT** routine looks like this:

   ObjectScript

```
ZJRNFILT(jidsys,dir,glo,type,restmode,addr,time) ;
 SET restmode=0 ;test only
 SET pid=##class(%SYS.Journal.Record).GetRealPIDSYSinFilter(jidsys,.ecpsysid)
 DO
##class(%SYS.System).WriteToConsoleLog($SELECT(pid="":"jid="_+jidsys,1:"pid="_pid)_",ecpsysid="_ecpsysid)

 QUIT
```

   Note: The *jidsys* argument in **^ZJRNFILT** contains two components: *jid* and *remsysid*, separated by a comma.

4. To restore all records after a specific time, the **^ZJRNFILT** routine looks like this:

   ObjectScript

```
ZJRNFILT(jid,dir,glo,type,restmode,addr,time)              /*Filter*/
 Set restmode=1                                           /*Return 1 for restore*/
 If time<$zdatetimeh("08/14/2015 14:18:31") Set restmode=0  /*except if before Aug 14 2015 2:18.31
 pm*/
 Quit
 ;
```

4.4.1.6 Display Journal Records Using ^JRNDUMP

To display the records in the journal file, enter `5` at the `Option` prompt of the **^JOURNAL** menu or run **^JRNDUMP** as shown in the following example:

1. Run the **^JRNDUMP** utility from the system manager namespace by entering:

```
%SYS>DO ^JRNDUMP

   Journal                Directory & prefix

   20151113.001           C:\MyCache\Mgr\Journal\
   20151113.002 [JRNSTART] C:\MyCache\mgr\journal\
   20151113.003           C:\MyCache\mgr\journal\
   20151113.004           C:\MyCache\mgr\journal\
   20151114.001           C:\MyCache\mgr\journal\
   20151115.001           C:\MyCache\mgr\journal\
   20151115.002           C:\MyCache\mgr\journal\
   20151115.003           C:\MyCache\mgr\journal\
>  20151115.004           C:\MyCache\mgr\journal\
```

2. The routine displays a list of journal files. A greater-than sign (>) appears to the left of the currently selected file followed by a prompt:

```
Pg(D)n,Pg(U)p,(N)ext,(P)rev,(G)oto,(E)xamine,(Q)uit =>
```

   Use these options to navigate to the journal file you wish to locate:

- If the instance is a mirror member, enter M to limit the list to mirror journal files only. (For information about mirror journal files, see Journal Files and Journal History Log in this chapter and Mirror Synchronization in the "Mirroring" chapter of the *Caché High Availability Guide*.)
- Enter D or U to page through the list of journal files.
- Enter N or P to move the > to the desired journal file.
- Enter G to directly enter the full pathname of the journal file to display.
- Enter E to display the contents of the selected journal file.
- Enter I to display information about the selected journal file and, optionally, a list of databases from the journal.
- Enter Q or **<Enter>** to quit the routine.

3. When you enter I, when you accept the currently selected journal file or specify a different one, information like the following is displayed:

```
Journal: C:\MyCache\mgr\journal\20151113.003
File GUID: 97734819-CA75-4CB1-9C3E-74D294784D23
Max Size: 1073741824
Time Created: 2015-11-13 10:44:52
File Count: 22
Min Trans: 22,3497948
Prev File: C:\MyCache\mgr\journal\20151113.002
Prev File GUID: 8C5D3476-F12C-4258-BF6C-7423876653A4
Prev File End: 0
Next File: C:\MyCache\mgr\journal\20151113.004
Next File GUID: 4F4D20B1-D38C-473E-8CF0-4D04C6AF90B0

(D)atabases,(Q)uit =>
```

   Min Trans is the file count and offset of the minimal transaction position, that is, any open transaction must have started at or later than that point.

   If the selected file is a mirror journal file, addition information is displayed.

   Entering Q at the prompt at the bottom returns you to the journal file list. Enter D to display database information like the following:

```
Journal: C:\MyCache\mgr\journal\20151113.003
  sfn  Directory or Mirror DB Name
================================================================================
    0  C:\MyCache\mgr\
    1  C:\MyCache\mgr\cachelib\
    2  C:\MyCache\mgr\cachetemp\
    3  :mirror:MIR:MIRTEST
    5  C:\MyCache\mgr\cache\
    6  C:\MyCache\mgr\user\

(P)rev,(N)ext,(Q)uit =>
```

   Enter Q to return to the journal file information display.

4. After you enter G or E, the utility displays the journal file name and begins listing the contents of the file by offset address. For example:

```
Journal: C:\MyCache\mgr\journal\20150330.002
   Address   Proc ID Op Directory        Global & Value
================================================================================
    131088      2980 S  C:\MyCache\mgr\  SYS("shdwcli","doctest","remend") = 1+
    131156      2980 S  C:\MyCache\mgr\  SYS("shdwcli","doctest","end") = 1013+
    131220      2980 S  C:\MyCache\mgr\  SYS("shdwcli","doctest","jrnend") = 1+
...
```

5. At the bottom of the current listing page is information about the journal file and another prompt:

```
Last record:     573004;   Max size: 1073741824
(N)ext,(P)rev,(G)oto,(F)ind,(E)xamine,(Q)uit =>
```

Use these options to navigate to the journal record you wish to display:

- Enter `N` or `P` to display the next or previous page of addresses.
- Enter `G` to move the display to a particular address.
- Enter `F` to search for a particular string within the journal file.
- Enter `E` to enter the address of a journal record and display its contents.
- Enter `Q` to return to the list of journal files.

6. After entering `E` or `G`, enter an address at the prompt. The `E` option displays the contents of the journal record at or near the address you entered; the `G` option displays the page of journal records starting at that location.

   For either option, the utility locates the record that is the closest to the offset address you specify; it does not need to be a valid address of a journal record. Also, you may enter `0` (zero) to go to the beginning of the journal file, or enter `-1` to go to the end of the journal file.

7. You may browse through a display of the journal records using `N` or `P` to display the next or previous journal record contents, respectively. When you are finished displaying records, enter `Q` at the prompt to return to the list of journal records.

There are different types of journal records:

- The journal header is 8192 bytes long. It appears once at the start of every journal file. The **^JRNDUMP** utility does not display the journal header record.
- Journal data records.
- Journal markers

The following is a sample journal file data record as displayed by **^JRNDUMP**. The example shows how a **Set** command is recorded. The new value is recorded, but not the old value, because the **Set** occurred outside a transaction:

```
Journal: C:\MyCache\mgr\journal\20150119.004

Address:                233028
Type:                   Set
In transaction:         No
Process ID:             4836
ECP system ID:          0
Time stamp:             60284,53240
Collation sequence:     5
Prev address:           232984
Next address:           0

Global:    ^["^^C:\MyCache\mgr\"]ABC
New Value: 2


(N)ext,(P)rev,(Q)uit =>
```

In a transaction, the old value is also recorded, to allow transaction rollback, as seen in this second example:

```
Journal: C:\MyCache\mgr\journal\20151115.004

Address:                204292
Type:                   Set
In transaction:         Yes
Process ID:             458772
ECP system ID:          0
Time stamp:             60584,52579 - 11/15/2015 14:36:19
Collation sequence:     5
Prev address:           204224
Next address:           204372

Global:    ^["^^C:\MyCache\mgr\"]ABC
New Value: 5
Old Value: 2


(N)ext,(P)rev,(Q)uit =>
```

The following is an example of a journal marker record created by an incremental backup:

```
Journal: C:\MyCache\mgr\journal\20151115.004

Address:                210848
Type:                   JrnMark
Marker ID:              -1
Marker text:            NOV 15 2015;03:14PM;Incremental
Marker seq number:      1
Prev marker address:    0
Time stamp:             60584,52579 - 11/15/2015 14:36:19
Prev address:           210744
Next address:           210940


(N)ext,(P)rev,(Q)uit =>
```

The following table describes each field in the journal data record.

Table 4–1: Journal Data Record Fields Displayed by ^JRNDUMP

Field	Description
Address	Location of this record in number of bytes from beginning of file. This is the only field where you enter a value to select a record.
Type	The type of operation recorded in this journal record entry. See the Journal File Operations table for possible types.
In transaction	Whether or not the update occurred in a transaction.
Process ID	Process ID number for the process issuing the command.
ECP system ID	ECP system ID number (0 if a local process).
Time stamp	Creation time of the journal buffer, in **$HOROLOG** and human-readable format. This is not the time the **Set** or **Kill** operation occurs, so it represents the earliest this particular operation could have happened.
Collation sequence	Collation sequence of the global being updated.
Prev address	Location of previous record (0 indicates this is the first record).
Next address	Location of next record (0 indicates this is the last record).
Cluster sequence #	Sequencing for globals in cluster-mounted databases. During cluster failover, journal entries from different nodes are updated in order of this cluster time sequencing.

Field	Description
Mirror Database Name	If a mirror journal file, the mirror name for the database on which the operation occurred.
Global	Extended reference of global being updated.
New Value	For a **Set** operation, the value assigned to the global.
Old Value	For a **Set** or **Kill** operation in a transaction, the value that was in the global before the operation.

The following table lists and describes the journal operations displayed in the **Op** column of a ^**JRNDUMP** journal file display and the **Type** field of a ^**JRNDUMP** journal record listing. For example, in the previous example of a journal file display, **S** in the **Op** column represents a **JRNSET** operation, while in the examples of journal record displays, **Set** appears in the **Type** field to indicate a **JRNSET** operation. Note that the **Type** column of the journal record display in the management portal (see View Journal Files) differs for some operations from the **Type** field of the ^**JRNDUMP** listing; for example, a **JRNSET** operation is indicated by **RemoteSET** in the portal and by **NSet** in ^**JRNDUMP** output. These differences are shown in the table.

The table also shows the codes that can be specified to filter journal records by operation when using the **SELECT^JRNDUMP** function.

Table 4–2: Journal File Operations

Operation	Description	Op in file listing	Type in ^JRNDUMP record listing	Type in Management Portal record listing	Numeric SELECT code	Alpha SELECT code
JRNSET	set a node, local	S[1]	Set	SET	6	s
JRNNSET	set a node, remote	S[1]	NSet	RemoteSET	10	s
JRNMIRSET	internal mirror operation[2]	S[1]	Mirror Set	MirrorSET	19	s
JRNBITSET	set a specified bit position in a node	b[1]	BitSet	BitSET	14	bs
JRNKILL	kill a node, local	K[1]	KillNode	KILL	7	k
JRNNKILL	kill a node, remote	K[1]	NKill	RemoteKILL	11	k
JRNKILLDES	kill a descendant node	k[1]	KillDesc	KILLdes	8	k
JRNMIRKILL	internal mirror operations[2]	k[1]	Mirror Kill	MirrorKILL	20	k
JRNZKILL	kill a node without killing subordinate nodes, local	k[1]	ZKill	ZKILL	9	zk

Operation	Description	Op in file listing	Type in ^JRNDUMP record listing	Type in Management Portal record listing	Numeric SELECT code	Alpha SELECT code
JRNNZKILL	kill a node without killing subordinate nodes, remote	k[1]	NZKill	RemoteZKILL	12	zk
JRNBEGTRANS	begin a transaction	BT	BeginTrans	BeginTrans	4	--
JRNTBEGINLEVEL	begin transaction level	BTL	BeginTrans with Level	BeginTrans with level	16	--
JRNCOMMIT	commit a transaction	CT	CommitTrans	CommitTrans	5	--
JRNTCOM-MITLEVEL	commit isolated transaction level	CTL	CommitTrans with Level	CommitTrans with level	18	--
JRNTCOMMIT-PENDLEVEL	commit pending transaction level	PTL	CommitTrans Pending with Level	CommitTrans Pending with level	17	--
JRNMARK	journal marker	M	JrnMark	Marker	13	--
JRNBIGNET	ECP networking	NN	NetReq	netsyn	15	--
JRNTROLEVEL	roll back a transac-tion	RB	Rollback	Rollback	21	--

[1] **T** is appended when the operation occurs within a transaction, for example **ST** for a **Set** operation within a transaction or **kT** for a **ZKill** operation within a transaction.

[2] Operation is ignored during journal restore.

Select Journal Records to Dump

The function **SELECT^JRNDUMP** lets you display any or all of the records in the journal file. Caché dumps selected records from the journal file, starting from the beginning of the file, based on the arguments passed to the function.

The syntax to use the `SELECT` entry point of the **^JRNDUMP** utility is as follows:

```
SELECT^JRNDUMP(%jfile,%pid,%dir,%glo,%gloall,%operation,%remsysid)
```

Argument	Description
%jfile	Journal file name. Default is the current journal file. You must specify the fully qualified path of the journal file.
%pid	Process ID in the journal record. Default is any process.
%dir	Directory in the journal record. Default is any directory.
%glo	Global reference in the journal record. Default is any global.

Argument	Description
%gloall	Global indicator whether to list entries related to all global nodes containing the name represented by *glo*: 0 — Exact match of global reference with the name specified in *glo*, 1 — Partial match; all records with a global reference that contains the name specified in *glo*. Default is 0.
%operation	Operation type of the journal record. Default is any operation. Use the numeric or alphabetic codes listed in the Journal File Operations table in the previous section.
%remsysid	ECP system ID of journal record. Default is any system.[1]

[1] If *%pid* is specified, then *%remsysid* defaults to local system (0); otherwise, it defaults to any system, the same as if it is specified as 0. That is, you *cannot* select journal entries only from the local system.

You may pass the null string for any argument, in which case the routine uses the defaults.

As with other terminal functions, you can use the `Device:` prompt to direct the output of **SELECT^JRNDUMP** to a device other than the terminal, or to a file. (See the chapter "I/O Devices and Commands" in the *Caché I/O Device Guide* for information on user device selection.) If you direct the output to a file, you are prompted for parameters. You must include `R` and `W` when writing to a file; if it is an existing file, include `A` to append output to the existing content rather than overwriting it; if a new file, you must include `N`. Enter **?** at the `Parameters?` prompt to display all possible choices.

Note: If the file you are overwriting is longer than the current output, the excess lines from the original file may not be removed from the updated file.

SELECT^JRNDUMP Examples

The following examples show different ways to select specific journal records.

To select all records in a journal file with the process ID `1203` and send the output to a new file called JRNDUMP.OUT:

```
%SYS>Do SELECT^JRNDUMP("C:\MyCache\mgr\journal\120020507.009","1203")

Device: SYS$LOGIN:JRNDUMP.OUT
Parameters: "RWN"=>
```

To select all records in the journal file that contain the global reference *^ABC*:

```
DO SELECT^JRNDUMP("C:\MyCache\mgr\journal\20050327.001","","","^ABC",1)
```

To select only records that have an exact match to the global reference *^ABC*:

```
DO SELECT^JRNDUMP("C:\MyCache\mgr\journal\20050327.001","","","^ABC",0)
```

Note: Records that are not an exact match, such as *^ABC(1)* or *^ABC(100)*, are not selected.

To select only records for local **Set** operations of global *^ABC*:

```
DO SELECT^JRNDUMP("C:\MyCache\mgr\journal\20050327.001","","","^ABC","","6")
```

To select only records for local and remote **Set** operations of global *^ABC*:

```
DO SELECT^JRNDUMP("C:\MyCache\mgr\journal\20050327.001","","","^ABC","","s")
```

4.4.1.7 Purge Journal Files Using PURGE^JOURNAL

To purge files, use the **PURGE^JOURNAL** routine or enter `6` at the `Option` prompt of the **^JOURNAL** menu, as shown in the following examples.

Example of running **PURGE^JOURNAL** directly:

```
zn "%SYS"
%SYS>Do PURGE^JOURNAL
```

Example of starting journaling from the **^JOURNAL** menu:

```
%SYS>Do ^JOURNAL

 ...
 6) Purge Journal Files (PURGE^JOURNAL)
 ...
Option? 6

1) Purge any journal NOT required for transaction rollback or crash recovery
2) Purge journals based on existing criteria (2 days or 2 backups)

Option?
```

The routine reports on the action taken in response to the option you specify. For example:

```
Option? 1

The following files have been purged (listed from latest to oldest):

3. c:\intersystems\cache\mgr\journal\20150714.001
2. c:\intersystems\cache\mgr\journal\20150713.001
1. c:\intersystems\cache\mgr\journal\20150710.003
```

If no files are purged, the following message is displayed:

```
None purged
```

4.4.1.8 Update Journal Settings Using ^JRNOPTS

As an alternative to using the Journal Settings page of the Management Portal, you can update the basic journal configuration settings using the **^JRNOPTS** routine or by entering 7 at the `Option` prompt of the **^JOURNAL** menu. To change the setting, type the new value at the prompt and press **Enter**. For example:

```
SYS>Do ^JRNOPTS

1) Primary Journal Directory: C:\MyCache\Mgr\Journal\
2) Alternate Journal Directory: D:\cachesys\altjournal\
3) Journal File Size Limit (MB) 1024
4) Journal File Prefix:
5) Journal Purge Options: 2 days OR 2 backups, whichever comes first
```

Entering a question mark (?) displays Help. For example:

```
Journal File Prefix: ?
  Enter an alphanumeric string ('_' allowed) or . to reset prefix to null
```

If you change any of the settings, then press **Enter** at a `Change Property?` prompt, you are given the option to activate the changes:

```
Change Property?
Save and activate changes? Yes =>
*** Journal options updated.
```

If you do not change any settings, you see the following message:

```
*** Nothing changed
```

4.4.1.9 Journal Encryption Using ENCRYPT^JOURNAL

For information on option `8) Activate or Deactivate Journal Encryption (ENCRYPT^JOURNAL)`, see the Configuring Caché Database Encryption Startup Settings section of the "Managed Key Encryption" chapter of the *Caché Security Administration Guide*, which describes details on journal file encryption.

4.4.1.10 Display Journal Status Using Status^JOURNAL

Choosing option `9) Display Journal status` displays a concise overview of journal status information including the following:

- Current journal directory and its remaining space
- Alternate journal directory (if different) and its remaining space
- Current journal file, its maximum size, and space used
- Journaling state, which can be one of the following:
  - Enabled
  - Disabled (stopped)
  - Disabled due to I/O error (suspended)
  - Frozen due to I/O error
  - Journal switch in progress (paused)

  Though suspended and frozen due to I/O error are the same journal state, the system takes different action; when frozen, it discards journal data.
- If applicable, the process IDs of any process running **^JRNSTART**, **^JRNSTOP**, or **^JRNSWTCH**

For example:

```
%SYS>Do ^JOURNAL

 ...
 9) Display Journal status (Status^JOURNAL)
 ...
Option? 9

Current journal directory:  C:\MyCache\Mgr\Journal\
Current journal directory free space (KB):  53503904
Alternate journal directory:  C:\MyCache\Mgr\
Alternate journal directory free space (KB): 53503904
Current journal file:  C:\MyCache\mgr\journal\20151129.001
Current journal file maximum size:    1073741824
Current journal file space used:         1979276
Journaling is enabled.
```

4.4.1.11 Restore Cluster Journal Using CLUMENU^JRNRESTO

Option `10) Cluster Journal Restore (CLUMENU^JRNRESTO)` is displayed only on a cluster node; for information, see Cluster Journal Restore in the "Cluster Journaling" chapter of this guide.

4.4.1.12 Manage Transaction Rollback Using Manage^JRNROLL

Caché provides the **^JRNROLL** utility to roll back partially completed transactions for records in the journal; use the **Manage** entry point (**Manage^JRNROLL**) when transaction rollbacks are pending or in progress at system startup or primary mirror member startup.

To start managing transaction rollback, run **Manage^JRNROLL** or enter `11` at the `Option` prompt of the **^JOURNAL** menu, as shown in the following example:

```
%SYS>Do ^JOURNAL

 ...
 11) Manage pending or in progress transaction rollback (Manage^JRNROLL)
 ...
Option? 11
```

Choosing option `11) Manage pending or in progress transaction rollback (Manage^JRNROLL)` displays a message similar to the following:

```
Transaction rollback is pending or in progress
Do you wish to run Manage^JRNROLL? Yes => Yes
Rollback operations currently in progress
     ID   Phase     MB Remaining   Current Open Transaction Count
     1    scan      307            2
          Rollback at system startup at 11/29/2015 15:54:35 (578MB)
          20151129.004 has 2 open transaction(s) starting at offset 11303
          2 file(s) remaining to process

1) Restart pending rollback
2) Interrupt transaction rollback
3) Redisplay rollback information

Option?
```

This option displays the state of transaction rollbacks, including what phase (scanning or rollback) it's in, the amount of data (MBs) remaining to be processed, the number of open transactions it has found, etc.

In addition, it lists sub-options that let you manage the listed transaction rollbacks. For example, you can interrupt the operation, in which case it is queued as a "pending" operation; then you can restart pending rollbacks.

Note: On a mirror, transaction rollback is executed twice: once for non-mirrored databases (at system startup); then for mirrored databases (when a system becomes the primary mirror member). As a result, when starting the primary mirror member, it may be necessary to interrupt the rollback twice, resulting in two pending operations. Restarting the pending operations performs the non-mirror and mirror rollbacks separately.

During rollback, messages are written to the console log (cconsole.log) every 10% of the way through (more or less) indicating how much space is left to process and how many open transactions are listed.

When journal files are purged, files required for pending transaction rollback are retained (for example, if they would otherwise have been deleted).

4.4.1.13 Restore Journal to Mirrored Database Using MirrorCatchup^JRNRESTO

You can restore mirror journal files to mirrored databases by entering `12` at the `Option` prompt of the **^JOURNAL** menu, or by answering `yes` at the `Catch-up mirrored databases?` prompt when using Option 4, `Restore Globals From Journal (^JRNRESTO)`. For example:

```
%SYS>Do ^JOURNAL

 ...
 12) Journal catch-up for mirrored databases (MirrorCatchup^JRNRESTO)
 ...
Option? Option? 12

Specify the list of mirrored databases you want to catch-up.
Enter database, * for all, ? for a list or to end list? *
Enter database or to end list?
Starting catch-up for the following mirrored database(s):
     sfn #6: c:\intersystems\cache20121x\mgr\mirrordb3\
Catch-up succeeded.
```

To catch up mirrored databases, journaling need not be running, but it must have been started at least once to ensure that the current journal directory is available from memory.

4.4.2 Recover from Startup Errors Using ^STURECOV

During the Caché startup procedure if the journal or transaction restore process encounters errors, such as <FILEFULL> or <DATABASE>, the procedure logs the errors in the console log (cconsole.log) and starts the system in single-user mode.

Caché provides a utility, **^STURECOV**, to help you recover from the errors and start Caché in multiuser mode. The routine has several options which you can use to retry the failed operation and bring the system up, or ignore the errors and bring the system up. The journal restore phase tries to do as much work as possible before it aborts. If a database triggers more than three errors, it aborts the recovery of that database and leaves the database dismounted.

Note: The **^STURECOV** utility does not work on a mirror member on which transaction rollback is pending or in progress because the system does not activate a mirrored database read/write until transaction rollback has been completed. In this case, Caché lets you run the **Manage^JRNROLL** routine, which provides a way to force the system to come up and store transaction rollback information which can be used to roll back transactions after the system is up and running. For more information, see Manage Transaction Rollback Using Manage^JRNROLL in this section.

During transaction rollback, the first error in a database causes the rollback process to skip that database in the future. The process does not fully replay transactions that reference that database; it stores them for rollback during the recovery process.

When Caché encounters a problem during the dejournaling phase of startup it generates a series of console log messages similar to the following:

```
08/10-11:19:47:024 ( 2240) System Initialized.
08/10-11:19:47:054 ( 2256) Write daemon started.
08/10-11:19:48:316 ( 1836) Performing Journal Recovery
08/10-11:19:49:417 ( 1836) Error in JRNRESTB: <DATABASE>restore+49^JRNRESTB
     C:\MyCache\mgr\journal\20150810.004 addr=977220
     ^["^^C:\MyCache\mgr\jo1666\"]test(4,3,28)
08/10-11:19:49:427 ( 1836) Error in JRNRESTB: <DATABASE>restore+49^JRNRESTB
     C:\MyCache\mgr\journal\20150810.004 addr=977268
     ^["^^C:\MyCache\mgr\test\"]test(4,3,27)
08/10-11:19:49:437 ( 1836) Error in JRNRESTB: <DATABASE>restore+49^JRNRESTB
     C:\MyCache\mgr\journal\20150810.004 addr=977316
     ^["^^C:\MyCache\mgr\test\"]test(4,3,26)
08/10-11:19:49:447 ( 1836) Error in JRNRESTB: <DATABASE>restore+42^JRNRESTB
     C:\MyCache\mgr\journal\20150810.004 addr=977748
     ^["^^C:\MyCache\mgr\test\"]test(4,2,70)
08/10-11:19:50:459 ( 1836) Too many errors restoring to C:\MyCache\mgr\test\.
 Dismounting and skipping subsequent records
08/10-11:19:50:539 ( 1836) 4 errors during journal restore,
see console.log file for details.
Startup aborted, entering single user mode.
```

If the errors are from transaction rollback, then the output looks similar to this:

```
08/11-08:55:08:732 ( 428) System Initialized.
08/11-08:55:08:752 ( 1512) Write daemon started.
08/11-08:55:10:444 ( 2224) Performing Journal Recovery
08/11-08:55:11:165 ( 2224) Performing Transaction Rollback
08/11-08:55:11:736 ( 2224) Max Journal Size: 1073741824
08/11-08:55:11:746 ( 2224) START: C:\MyCache\mgr\journal\20150811.011
08/11-08:55:12:487 ( 2224) Journaling selected globals to
     C:\MyCache\mgr\journal\20150811.011 started.
08/11-08:55:12:487 ( 2224) Rolling back transactions ...
08/11-08:55:12:798 ( 2224) Error in %ROLLBACK: <DATABASE>set+2^%ROLLBACK
     C:\MyCache\mgr\journal\20150811.010 addr=984744
     ^["^^C:\MyCache\mgr\test\"]test(4,1,80)
08/11-08:55:12:798 ( 2224) Rollback of transaction for process id #2148
 aborted at offset 984744 in C:\MyCache\mgr\journal\20150811.010.
08/11-08:55:13:809 ( 2224) C:\MyCache\mgr\test\ dismounted -
      Subsequent records will not be restored
08/11-08:55:13:809 ( 2224) Rollback of transaction for process id #924
 aborted at offset 983464 in C:\MyCache\mgr\journal\20150811.010.
08/11-08:55:14:089 ( 2224) STOP: C:\MyCache\mgr\journal\20150811.011
08/11-08:55:14:180 ( 2224) 1 errors during journal rollback,
see console.log file for details.
Startup aborted, entering single user mode.
```

Both output listings end with the same instructions:

```
Enter Cache' with
     C:\MyCache\bin\cache -sC:\MyCache\mgr -B
and D ^STURECOV for help recovering from the errors.
```

When Caché cannot start properly, it starts in *single-user* mode. While in this mode, execute the special commands indicated in these instructions to enter Caché. For example, for a Windows installation, enter the following:

```
C:\MyCache\bin\>cache -sC:\MyCache\mgr -B
```

UNIX®/Linux systems have a slightly different syntax.

This runs the Caché executable from the Caché installation bin directory (*install-dir*\bin) indicating the pathname (by using the `-s` argument) of the system manager's directory (*install-dir*\mgr) and inhibits all logins except one emergency login (by using the `-B` argument).

You are now in the manager's namespace and can run the startup recovery routine, ^**STURECOV**:

```
Do ^STURECOV
```

The ^**STURECOV** journal recovery menu appears as follows:

```
Journal recovery options
------------------------------------------------------------
1) Display the list of errors from startup
2) Run the journal restore again
3) Bring down the system prior to a normal startup
4) Dismount a database
5) Mount a database
6) Database Repair Utility
7) Check Database Integrity
8) Reset system so journal is not restored at startup
9) Display instructions on how to shut down the system
10) Display Journaling Menu (^JOURNAL)
------------------------------------------------------------
H) Display Help
E) Exit this utility
------------------------------------------------------------

Enter choice (1-10) or [Q]uit/[H]elp?
```

Only UNIX®/Linux systems contain option `9` on the menu.

Before starting the system in *multiuser* mode, correct the errors that prevented the journal restore or transaction rollback from completing. You have several options regarding what to do:

- *Option 1* — The journal restore and transaction rollback procedure tries to save the list of errors in the *^%SYS()* global. This is not always possible depending on what is wrong with the system. If this information is available, this option displays the errors.
- *Option 2* — This option performs the same journal restore and transaction rollback which was performed when the system was started. The amount of data is small so it should not be necessary to try and restart from where the error occurred.
- *Option 3* — When you are satisfied that the system is ready for use, use this option to bring the instance down prior to restarting it in a normal fashion.
- *Option 4* — This option lets you dismount a database. Generally, use this option if you want to let users back on a system but you want to prevent them from accessing a database which still has problems (^**DISMOUNT** utility).
- *Option 5* — This option lets you mount a database (^**MOUNT** utility).
- *Option 6* — This option lets you edit the database structure (^**REPAIR** utility).
- *Option 7* — This option lets you validate the database structure (^**INTEGRIT** utility).

- *Option 8* — This updates the system so that it does not attempt journal restore or transaction rollback at startup. This applies only to the next time the startup process is run. Use this in situations where you cannot get journal recovery to complete and you need to allow users back on the system. Consider dismounting the databases which have not been recovered. This operation is not reversible. You can perform journal restore manually using the **^JRNRESTO** utility.
- *Option 9* — It is not possible to shut down the system from this utility, but this option displays instructions on how to shut the system down from the UNIX® command line.
- *Option 10* — This option brings up the journaling menu which allows you to browse and restore journal files. There are options which start and stop journaling but these are not generally of interest when resolving problems with journaling at startup.

Take whatever corrective action is necessary to resolve the problem. This may involve using the **^DATABASE** routine to extend the maximum size of the database, or it may require freeing space on the file system or using the **^INTEGRIT** and **^REPAIR** utilities to find and correct database degradation. As you do this work, you can use *Option 2* of the **^STURECOV** utility to retry the journal replay/transaction rollback as many times as necessary. You can display any errors you encounter, including those from when the system started, using *Option 1*. When you correct all the problems, and run *Option 2* without any errors, use *Option 3* to bring the system up in multiuser mode.

If you find that you cannot resolve the problem, but you still want to bring the system up, use *Option 8* to clear the information in the Caché image journal (.wij file) that triggers journal restore and transaction rollback at startup. The option also logs the current information in the console log. Once this completes, use *Option 3* to start the system. Use this facility with care, as it is not reversible.

If Caché was unable to store the errors during startup in the *^%SYS()* global for **^STURECOV** to display, you may get an initial message before the menu that looks like this:

```
There is no record of any errors during the prior startup
This could be because there was a problem writing the data
Do you want to continue ? No => yes
Enter error type (? for list) [^] => ?

Supported error types are:
     JRN - Journal and transaction rollback

Enter error type (? for list) [^] => JRN
```

Journaling errors are one type of error that this utility tries to handle and that is the scope of this chapter. Other error types are discussed in the appropriate sections of the documentation.

CAUTION: Only use the **^STURECOV** utility when the system is in single-user mode following an error during startup. Using it while the system is in any other state (for example, up running normally) can cause serious damage to your data as it restores journal information if you ask it to and this information may not be the most current data. The **^STURECOV** utility warns you, but it lets you force it to run.

4.4.3 Convert Journal Files Using ^JCONVERT and ^%JREAD

The **^JCONVERT** routine is a utility that reads journal files and converts them to a common file in variable record format. The **^%JREAD** utility can then read this file and apply journal transactions to databases on a different system. The **^JCONVERT** utility exists on older InterSystems database products as well as all versions of Caché. Use these utilities to move journal data between different system versions that do not have compatible journal files.

For example, if you are converting to a new version of Caché and need to minimize downtime, perform the following steps:

1. Enable journaling on the old system.
2. Run a backup on the old system; this switches to a new journal file on the old system.
3. Continue journaling on the old system.

4. Restore the backup of the old system on the new system and perform any necessary conversions.
5. Stop the old system and run **^JCONVERT** on the journal files created on the old system since the backup.
6. Apply the transactions from the old system to the new system using the file created from **^JCONVERT** as input to **^%JREAD** on the new system.

The **^JCONVERT** utility uses the same process as the journal restore utility to select and filter the journal files for processing. You can include a range of journal files as input and create one output file. See the Restore Globals From Journal Files Using ^JRNRESTO section for details on selecting and filtering journal files.

The converted file is in variable record format. The default character encoding is UTF8, which is compatible with the current **^%JREAD** utility on all platforms, and can be moved among platforms with binary FTP. If you answer NO at the **Use UTF8 character translation?** prompt, no character encoding is applied.

Globals in the journal file are stored with a specific directory reference appended to the global reference. You can choose either to include the directory reference in the converted file, or exclude it. If you include it, you can always filter it out or change it later during the **^%JREAD** procedure.

The directory reference determines where **^%JREAD** sets the global on the target system. If you do not include the directory reference, **^%JREAD** makes all sets in the current directory. If you do include the directory reference, the utility makes sets in the same directory as on the source system unless translated by a **^%ZJREAD** program you supply. If the target system is on a different operating system or the databases reside in different directories on the target system, you *must* supply a **^%ZJREAD** routine to translate the directory reference.

The **^%JREAD** routine reads a common journal file format and applies the journal transactions to the databases on the target system. During the import of records, if a **^%ZJREAD** routine exists, the utility calls it for each journal transaction allowing you to manipulate the journal records. You can reference the following variables in your **^%ZJREAD** routine:

```
type    - Transaction type
gref    - Global reference
value   - Global value
%ZJREAD - 1:Apply transaction, 0:Do not apply transaction
```

If you decide not to apply a transaction, set the variable *%ZJREAD* to 0 (zero) to skip the record. You can also modify the other variables. For example, you can change the directory specification by modifying *gref*.

The following is an example **^%ZJREAD** routine. It looks for transactions that contain updates to *%SYS("JOURNAL"*, and prevents them from being applied. You can copy this and modify it to suit your needs:

ObjectScript

```
%ZJREAD;
 /*The following variables are defined; you can modify them
   before the transaction gets applied

        type - Transaction type
        gref - Global reference
        value - Global value
        %ZJREAD - 1:Apply transaction, 0:Do not apply transaction
 */
 If gref["SYS(""JOURNAL""" Set %ZJREAD=0
 Quit
```

Sample Run of ^JCONVERT

The following is a sample run of the **^JCONVERT** utility:

```
%SYS>Do ^JCONVERT

Journal Conversion Utility  [ Cache Format --> Common Format ]


The converted file will be in variable record format.
The default character translation UTF8 is compatible with current ^%JREAD
on all platforms and can be moved among platforms with binary FTP.
If you answer NO, no character translation will be applied.

Use UTF8 character translation? <Yes>


Globals in the journal file are stored with a specific directory reference
appended to the global reference. You can choose either to include
the directory reference in the converted file, or exclude it. Note that
if you include it, you can always filter it out or change it later during
the %JREAD procedure.  The directory reference determines where ^%JREAD sets
the global on the target system.  If the directory reference is not included,
all sets are made to the current directory.  If the directory reference is
included, sets will be made to the same directory as on the source system
unless translated by a ^%ZJREAD program you supply.  If the target system
is on a different operating system or the databases reside in different
directories on the target system, the ^%ZJREAD program must be used to
translate the directory reference.

Include the directory reference? <Yes>

Enter common journal file name:  common.jrn

Common journal file: common.jrn
Record separator: Variable
Directory reference: Yes

Use current journal filter (ZJRNFILT)? no
Use journal marker filter (MARKER^ZJRNFILT)? no
Process all journaled globals in all directories?   enter Yes or No, please
Process all journaled globals in all directories? yes
Specify range of files to process (names in YYYYMMDD.NNN format)

from:     <20151201.001> [?] => 20151202.001

through:  <20151204.001> [?] =>

Prompt for name of the next file to process? No => No


Provide or confirm the following configuration settings:

Journal File Prefix: =>

Files to dejournal will be looked for in:
     C:\MyCache\mgr\journal\
     C:\MyCache\mgr\
in addition to any directories you are going to specify below, UNLESS
you enter a minus sign ('-' without quotes) at the prompt below,
in which case ONLY directories given subsequently will be searched

Directory to search: <return when done>
Here is a list of directories in the order they will be searched for files:
     C:\MyCache\mgr\journal\
     C:\MyCache\mgr\


You may tailor the response to errors by choosing between the alternative
actions described below.  Otherwise you will be asked to select an action
at the time an error actually occurs.

     Either Continue despite database-related problems (e.g., a target
     database is not journaled, cannot be mounted, etc.), skipping affected
     updates

     or     Abort if an update would have to be skipped due to a
     database-related problem (e.g., a target database is not journaled,
     cannot be mounted, etc.)

     Either Abort if an update would have to be skipped due to a
     journal-related problem (e.g., journal corruption, some cases of missing
     journal files, etc.)
```

```
     or     Continue despite journal-related problems (e.g., journal
     corruption, some missing journal files, etc.), skipping affected updates

     Either Apply sorted updates to databases before aborting

     or     Discard sorted, not-yet-applied updates before aborting (faster)

Would you like to specify error actions now? No => yes


     1.  Continue despite database-related problems (e.g., a target database
     is not journaled, cannot be mounted, etc.), skipping affected updates

     2.  Abort if an update would have to be skipped due to a database-related
     problem (e.g., a target database is not journaled, cannot be mounted,
     etc.)

Select option [1 or 2]:  1

     1.  Abort if an update would have to be skipped due to a journal-related
     problem (e.g., journal corruption, some cases of missing journal files,
     etc.)

     2.  Continue despite journal-related problems (e.g., journal corruption,
     some missing journal files, etc.), skipping affected updates

Select option [1 or 2]:  2

     1.  Apply sorted updates to databases before aborting

     2.  Discard sorted, not-yet-applied updates before aborting (faster)

Select option [1 or 2]:  2
Based on your selection, this restore will

** Continue despite database-related problems (e.g., a target database is not
journaled, cannot be mounted, etc.), skipping affected updates

** Continue despite journal-related problems (e.g., journal corruption, some
missing journal files, etc.), skipping affected updates

** Discard sorted, not-yet-applied updates before aborting (faster)



C:\MyCache\mgr\journal\20151202.001
  13.98%  14.93%  15.95%  17.14%  18.25%  19.27%  20.49%  21.63%  22.65%  23.84%
  24.99%  25.97%  27.10%  28.25%  29.31%  30.50%  31.72%  32.84%  33.84%  34.84%
  35.84%  36.85%  37.91%  38.99%  40.10%  41.08%  42.03%  42.97%  43.93%  44.94%
  45.95%  47.05%  48.11%  49.07%  50.04%  51.02%  52.03%  53.07%  54.14%  55.25%
  56.21%  57.17%  58.15%  59.14%  60.18%  61.24%  62.33%  63.28%  64.20%  65.15%
  66.10%  67.11%  68.13%  69.05%  69.94%  70.83%  71.61%  72.41%  73.09%  73.85%
  74.59%  75.32%  76.06%  76.75%  77.73%  78.70%  79.65%  80.59%  81.53%  82.46%
  83.40%  84.33%  85.27%  86.05%  86.59%  87.13%  87.67%  88.23%  88.78%  89.34%
  89.89%  90.61%  93.28%  94.38%  97.12%  98.21%  99.93%100.00%
***Journal file finished at 11:31:36


C:\MyCache\mgr\journal\20151203.001
  14.01%  14.96%  15.98%  17.18%  18.29%  19.31%  20.53%  21.67%  22.69%  23.88%
  25.03%  26.01%  27.15%  28.30%  29.36%  30.55%  31.78%  32.90%  33.90%  34.90%
  35.91%  36.92%  37.99%  39.06%  40.17%  41.16%  42.11%  43.05%  44.01%  45.03%
  46.04%  47.14%  48.20%  49.17%  50.14%  51.11%  52.13%  53.17%  54.25%  55.36%
  56.33%  57.29%  58.27%  59.26%  60.30%  61.36%  62.46%  63.40%  64.33%  65.28%
  66.23%  67.24%  68.26%  69.19%  70.08%  70.97%  71.76%  72.56%  73.25%  74.01%
  74.75%  75.47%  76.22%  76.91%  77.89%  78.87%  79.83%  80.77%  81.70%  82.64%
  83.58%  84.52%  85.46%  86.24%  86.78%  87.32%  87.87%  88.42%  88.98%  89.53%
  90.09%  90.81%  93.49%  94.59%  97.33%  98.42%100.00%
***Journal file finished at 11:31:37


C:\MyCache\mgr\journal\20151204.001
  13.97%  14.92%  15.93%  17.12%  18.24%  19.25%  20.47%  21.61%  22.62%  23.82%
  24.96%  25.94%  27.07%  28.22%  29.28%  30.46%  31.69%  32.80%  33.80%  34.80%
  35.80%  36.81%  37.87%  38.94%  40.05%  41.04%  41.98%  42.92%  43.88%  44.89%
  45.90%  47.00%  48.06%  49.02%  49.98%  50.96%  51.97%  53.01%  54.08%  55.19%
  56.15%  57.11%  58.08%  59.07%  60.12%  61.17%  62.26%  63.20%  64.13%  65.07%
  66.02%  67.03%  68.05%  68.97%  69.86%  70.75%  71.53%  72.33%  73.01%  73.77%
  74.51%  75.23%  75.98%  76.67%  77.64%  78.61%  79.56%  80.50%  81.43%  82.37%
  83.30%  84.24%  85.17%  85.95%  86.49%  87.03%  87.57%  88.13%  88.68%  89.23%
  89.79%  90.51%  93.18%  94.27%  97.01%  98.10%  99.81%100.00%
***Journal file finished at 11:31:38
```

```
[journal operation completed]
Converted 26364 journal records
```

4.4.4 Set Journal Markers Using ^JRNMARK

To set a journal marker in a journal file, use the following routine:

```
SET rc=$$ADD^JRNMARK(id,text)
```

Argument	Description
id	Marker ID (for example, -1 for backup)
text	Marker text of any string up to 256 characters (for example, “timestamp” for backup)
rc	Journal location of the marker (journal offset and journal file name, delimited by a comma) or, if the operation failed, a negative error code followed by a comma and a message describing the error. Note that a journal offset must be a positive number.

4.4.5 Manipulate Journal Files Using ^JRNUTIL

InterSystems provides several functions in the **^JRNUTIL** routine. You can use these functions for writing site-specific routines to manipulate journal records and files.

The following table lists the functions available in the routine.

Table 4–3: Functions Available in ^JRNUTIL

Journaling Task	Function Syntax
Close a journal file	$$CLOSEJRN^JRNUTIL(*jrnfile*)
Delete a journal file	$$DELFILE^JRNUTIL(*jrnfile*)
Read a record from a journal file into a local array	$$GETREC^JRNUTIL(*addr,jrnode*)
Switch to a different journal file directory	$$JRNSWCH^JRNUTIL(*newdir*)
Open a journal file	$$OPENJRN^JRNUTIL(*jrnfile*)
Use an opened journal file	$$USEJRN^JRNUTIL(*jrnfile*)

Important: The **DELFILE^JRNUTIL** function does not check for open transactions before deleting the journal file.

The following table describes the arguments used in the utility.

Argument	Description
addr	Address of the journal record.
jrnfile	Name of journal file.
newdir	New journal file directory.
jrnode	Local variable passed by reference to return journal record information.

4.4.6 Manage Journaling at the Process Level Using %NOJRN

If journaling is enabled system-wide, you can stop journaling for **Set** and **Kill** operations on globals within a particular process by issuing a call to the **^%NOJRN** utility from within an application or from programmer mode as follows:

```
%SYS>DO DISABLE^%NOJRN
```

Journaling remains disabled until one of the following events occurs:

- The process halts.
- The process issues the following call to reactivate journaling:

  ```
  %SYS>DO ENABLE^%NOJRN
  ```

Note: Disabling journaling using **DISABLE^%NOJRN** does not affect mirrored databases.

You must have at least read access to the **%Admin_Manage** resource to use **DISABLE^%NOJRN**.

4.5 Journal I/O Errors

When Caché encounters a journal file I/O error, the response depends on the **Freeze on error** journal setting, which is on the Journal Settings page of the Management Portal (**System Administration** > **Configuration** > **System Configuration** > **Journal Settings**). The **Freeze on error** setting works as follows:

- When the **Freeze on error** setting is **No** (the default), the journal daemon retries the failed operation until it succeeds or until one of several conditions is met, at which point all journaling is disabled. This approach keeps the system available, but disabling journaling compromises data integrity and recoverability.
- When **Freeze on error** is set to **Yes**, all journaled global updates are frozen. This protects data integrity at the expense of system availability.

The **Freeze on error** setting also affects application behavior when a local transaction rollback fails.

InterSystems recommends you review your business needs and determine the best approach for your environment. The following sections describe the impact of each choice:

- Journal Freeze on Error Setting is No
- Journal Freeze on Error Setting is Yes
- Impact of Journal Freeze on Error Setting on Transaction Rollback with TROLLBACK

4.5.1 Journal Freeze on Error Setting is No

If you configure Caché *not* to freeze on a journal file I/O error, the journal daemon retries the failed operation periodically (typically at one second intervals) until either it succeeds or one of the following conditions is met:

- The daemon has been retrying the operation for a predetermined time period (typically 150 seconds)
- The system cannot buffer any further journaled updates

When one of these conditions is met, journaling is disabled and database updates are no longer journaled. As a result, the journal is no longer a reliable source from which to recover databases if the system crashes. The following conditions exist when journaling is disabled:

- Transaction rollback fails, generating <ROLLFAIL> errors and leaving transactions partly committed.
- Shadowing becomes undependable once updates to the source databases are no longer journaled because it relies on journaling of the source databases.
- Crash recovery of uncommitted data is nonexistent.
- Full recovery no longer exists. You are able to recover only to the last backup.
- ECP lock and transaction recoverability guarantees are compromised.
- If the system crashes, Caché startup recovery does not attempt to roll back incomplete transactions started before it disabled journaling because the transactions may have been committed, but not journaled.

What to do if journaling is disabled?

To summarize, if journaling is disabled, perform the following steps:

1. *Resolve the problem* — As soon as possible, resolve the problem that disabled journaling.
2. *Switch the journal file* — The Journal daemon retries the failed I/O operation periodically in an attempt to preserve the journal data accumulated prior to the disabling. If necessary, you can switch the journal file to a new directory to resolve the error; however, Caché does *not* re-enable journaling automatically even if it succeeds with the failed I/O operation and switches journaling to a new file. It also does *not* re-enable journaling if you switch the journal file manually.
3. *Back up the databases* — on the main server (the backup automatically re-enables journaling if you have not done so).

   InterSystems strongly recommends backing up your databases as soon as possible after the error to avoid potential data loss. In fact, performing a Caché online backup when journaling is disabled due to an I/O error restarts journaling automatically, provided that the error condition that resulted in the disabling of journaling has been resolved and you have sufficient privileges to do so. You can also enable journaling by running **^JRNSTART**.

   When a successful backup operation restarts journaling, Caché discards any pending journal I/O, since any database updates covered by the pending journal I/O are included in the backup.

   Important: Starting journaling requires higher privileges than running a backup.

4. *Restore shadow databases* — If using shadowing, restore the backup to the shadow(s) to synchronize the databases, and restart the shadow from the new journal file started since the backup.

4.5.2 Journal Freeze on Error Setting is Yes

If you configure Caché to freeze on a journal file I/O error, all journaled global updates are frozen immediately upon such an error. This prevents the loss of journal data at the expense of system availability. Global updates are also frozen if the journal daemon has been unable to complete a journal write for at least 30 seconds.

The journal daemon retries the failed I/O operation and unfreezes global updates after it succeeds. Meanwhile, the freezing of global updates causes other jobs to hang. The typical outcome is that Caché hangs until you resolve the journaling problem, with the system appearing to be down to operational end-users. While Caché is hung you can take corrective measures, such as freeing up disk space, switching the journal to a different disk, or correcting a hardware failure.

The advantage to this option is that once the problem is resolved and Caché resumes normal operation, no journal data has been lost. The disadvantage is that the system is less available or unavailable while the problem is being solved.

Caché posts alerts (severity 3) to the cconsole.log file periodically while the journal daemon is retrying the failed I/O operation.

4.5.3 Impact of Journal Freeze on Error Setting on Transaction Rollback with TROLLBACK

It is important to be aware that the **Freeze on error** setting you choose can have significant implications for application behavior unrelated to journaling. When an application attempts to roll back an open transaction using the **TROLLBACK** command (see TROLLBACK in the *Caché ObjectScript Reference*) and the attempt fails, the same tradeoff presents itself as is faced when a journal I/O error is encountered: that of data integrity versus availability. Like journaling, **TROLLBACK** uses the **Freeze on error** setting to determine the appropriate behavior, as follows:

- When the **Freeze on error** setting is **No** (the default), the process initiating the transaction and the **TROLLBACK** receives an error, the transaction is closed, and the locks retained for the transaction are released. This approach keeps the application available, but compromises data integrity and recoverability.
- When **Freeze on error** is set to **Yes**, the initiating process halts and CLNDMN makes repeated attempts to roll back the open transaction. During the CLNDMN retry period, locks retained for the transaction remain intact, and as a result the application might hang. This protects data integrity at the expense of application availability.

If CLNDMN repeatedly tries and fails to roll back an open transaction for a dead job (as reported in the console log), you can use the **Manage^CLNDMN** utility to manually close the transaction.

Note: The **Freeze on error** setting affects local (non-ECP) transaction rollback only.

4.6 Special Considerations for Journaling

Review the following special considerations when using Caché journaling:

- Performance
- UNIX® File System Recommendations
- System Clock Recommendations
- Disabling Journaling for Filing Operations

4.6.1 Performance

While journaling is crucial to ensuring the integrity of your database, it can consume disk space and slow performance, depending on the number of global updates being journaled.

Journaling affects performance because updates result in double processing, as the change is recorded in both the database and the journal file. Caché uses a flat file journaling scheme to minimize the adverse effect on performance.

4.6.2 UNIX® File System Recommendations

The "Supported File Systems" table in the "Supported Technologies" section of the online *InterSystems Supported Platforms* document for this release, which outlines file systems recommended and supported by InterSystems on UNIX®/Linux platforms, includes notes about mount options for optimum journaling performance.

Note: When you configure the primary or alternate journal directory on a file system that does not have the recommended mount option, a message like the following is entered in the console log:

The device for the new journal file was not mounted with a recommended option (cio).

4.6.3 System Clock Recommendations

All operating systems supported by Caché provide Network Time Protocol (NTP) clients, which keep the system clock synchronized to a reference system, as well as facilities that automatically adjust the system clock between daylight saving time and standard time.

It is recommended that you rely on the automatic clock management features of the operating system to keep the system clock synchronized and regulated rather than adjust the system clock manually.

If you must make manual time adjustments for tasks such as testing, be sure to use a test environment (rather than the production environment) when performing such tasks. Furthermore, manual adjustments should be made with care because non-chronological events – such as adjusting the clock forward or backward – may cause issues for some utilities.

4.6.4 Disabling Journaling for Filing Operations

Under certain circumstances, it may be useful or necessary to disable journaling for filing operations, such as object saves and deletes. There are two ways to do this:

- When you open an object (typically with **%OpenId** or **%Open**), specify a concurrency value of 0. However, if the object is already open with a higher concurrency value, then specifying a concurrency of 0 is not effective.
- Suspend object filer transaction processing for the current process. To do this, call **$system.OBJ.SetTransactionMode(0)** (which is the **SetTransactionMode** method of the %SYSTEM.OBJ class; you can invoke it through the special $system object). The **SetTransactionMode** method takes a value of 0 or 1: 0 turns off object filer transactions and 1 turns them on. Note that this setting affects the entire process, not just the current filing operation.

Important: While certain circumstances call for disabling journaling, make sure that this is necessary before doing it. Otherwise, there may be a journal that does not include all the data required, which can result in the permanent loss of data.

5

Shadowing

Shadowing enables secondary computers to maintain a "shadow" copy of selected databases as they are updated on a primary machine. By continually transferring journal information from the primary machine to the secondary machines, shadowing enables recovery to a system which is typically within only a few transactions of the source database.

You can use shadowing for many purposes, each with its own set of important considerations depending on your system environment. Some of the most common objectives satisfied by shadowing include the following:

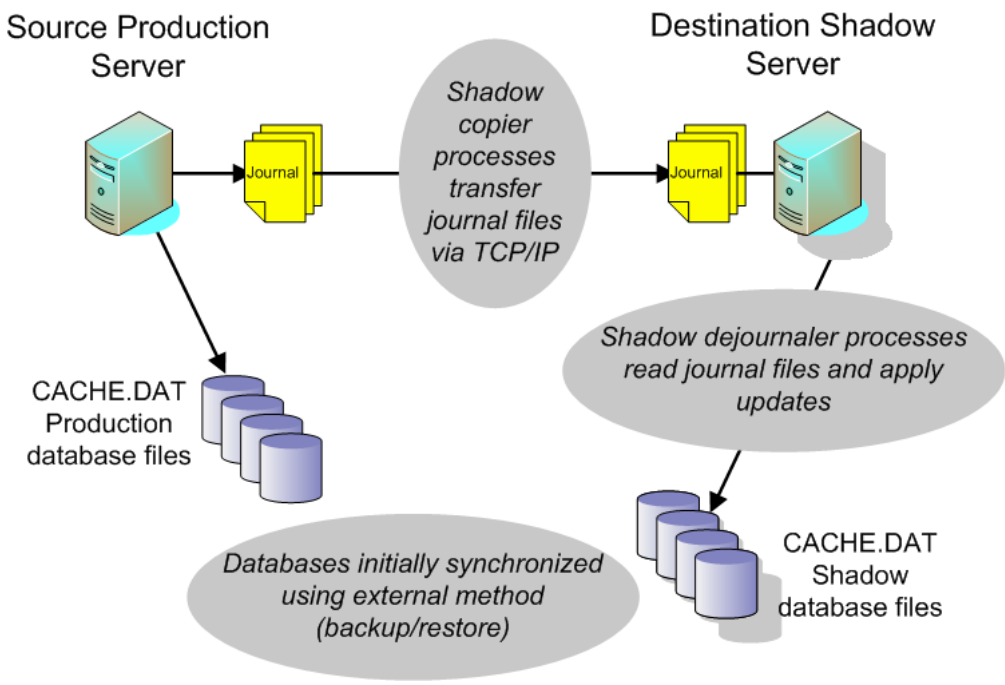
- *Disaster recovery*, the most common use; it is simple and inexpensive.
- *Read-only report server* where ad hoc reporting tasks can operate on current data without affecting production.
- *Low-budget replication* where the databases are replicated on the shadow instance using journaling.

This chapter discusses the following topics:

5.1 Shadowing Overview

A primary Caché instance may have one or more shadows. Shadowing monitors database activity on a primary system, the source, and causes the same activity to occur on a secondary system, the destination. It does this through a shadow client service running on the destination that continually requests journal file details from a shadow service running on the source. The shadow service responds by sending the details of the actual **Set**, **Kill**, and **$Bit** journal record entries to the destination shadow over a TCP connection.

Figure 5–1: Shadowing Overview



Important: The source and destination servers can be of different hardware, operating system, or CPU chipset. To avoid data incompatibility, however, the destination shadow Caché instance must use the same character width (8-bit or Unicode; see Caché Character Width in the *Caché Installation Guide*) and the same locale (see Using the NLS Settings Page of the Management Portal in the "Configuring Caché" chapter of the *Caché System Administration Guide*) as the source Caché instance. The one exception to this requirement is that an 8-bit instance using a locale based on the ISO 8859 Latin-1 character set is compatible with a Unicode instance using the corresponding wide character locale. For example, an 8-bit instance using the **enu8** locale can be configured as the shadow source database server with a Unicode instance using the **enuw** locale as the destination shadow.

All shadowing uses a fast transmission method which allows more efficient performance by sending the compacted journal file block by block. The shadow applies all transactions to the local databases. The transmission mode requires the data to be written to the journal file, which may introduce a delay of a few seconds. The shadow establishes a TCP connection to the server and receives the journal file. As the journal file downloads, another shadow process applies the journal entries to the local destination copy of the database.

Upon connecting to the data source server, the destination shadow sends the server the name of the journal file and the starting point. The shadow checks for new records periodically. If it does not have the latest records, the shadow downloads them and updates the databases. During these processes, Caché continually stores checkpoints in a shadow global to facilitate rollback and restart capabilities.

Caché purges the destination shadow copies of source journal files automatically. You can configure how long to keep files that are eligible for purging, that is, ones that have been dejournaled and do not contain any open transactions.

5.2 Configuring Shadowing

This section explains how to configure and set up shadowing in Caché and includes the following procedures:

- Configuring the Source Database Server

- Configuring the Destination Shadow

Important: If you want to configure both mirroring and shadowing for the same databases, bear in mind the following guidelines:

- If the mirror has only one failover member, you can configure the failover member as the shadow source; you can also configure an async member as the source.
- If the mirror has two failover members, you must configure an async member as the shadow source; you cannot configure a failover member as the source.
- To shadow both mirrored and non-mirrored databases on a mirror member, you must configure separate mirrored and non-mirrored shadows.
- To shadow databases from two or more different mirrors on an async member, you must configure a separate shadow for each mirror.

See the "Mirroring" chapter of the *Caché High Availability Guide* for information about configuring mirroring.

5.2.1 Configuring the Source Database Server

Before enabling shadowing on a source database server, ensure that the destination system can make a TCP connection to the source system. If you plan to secure this connection using SSL, a `%SuperServer` SSL/TLS configuration must exist on the source. See Configuring the Caché Superserver to Use SSL/TLS in the "Using SSL/TLS with Caché" chapter of the *Caché Security Administration Guide* for details.

Important: A shadow service cannot run on a system with a single-server license.

Use the Management Portal from the Caché instance running on the source system to enable the shadow service, restrict connections, and enable global journaling for the databases you are shadowing. These procedures are described in the following topics:

- Enable the Shadowing Service
- Enable Journaling

For information on methods and queries available for interfacing with the data source of a shadow without using the Management Portal, see the SYS.Shadowing.DataSource class documentation in the *InterSystems Class Reference*.

Also see the Important Journaling Considerations section for issues that may pertain to your environment.

5.2.1.1 Enable the Shadowing Service

To use shadowing, you must enable the shadowing service using the Security Management portion of the Management Portal. You may also restrict shadowing access by entering the IP addresses of allowed connections:

1. Navigate to the Services page of the Management Portal (**System Administration** > **Security** > **Services**).
2. Click **%Service_Shadow** in the list of service names to edit the shadow service properties.
3. Select the **Service enabled** check box. Before clicking **Save**, you may want to first restrict what IP addresses can connect to this database source. If so, perform the next step, and then click **Save**.
4. In the **Allowed Incoming Connections** box, any previously entered server addresses are displayed in the **IP Address** list. Click **Add** to add an **IP Address**. Repeat this step until you have entered all permissible addresses.

   You may delete any of these addresses individually by clicking **Delete** in the appropriate row, or click **Delete All** to remove all addresses, therefore allowing connections from any address.

5.2.1.2 Enable Journaling

Verify that you are journaling each database that you wish to shadow.

1. Navigate to the Local Databases page of the Management Portal (**System Administration** > **Configuration** > **System Configuration** > **Local Databases**) and view the **Journal** column for each database you wish to shadow.
2. To change the journal state from `No` to `Yes`, click **Edit** in the row of the appropriate database to edit the database properties.
3. In the **Global Journal State** list, click **Yes** and then click **Save**.

By default, the CACHELIB, CACHE, DOCBOOK, and SAMPLES databases are not journaled and, as a result, you cannot shadow them — CACHETEMP is never journaled.

5.2.1.3 Important Journaling Considerations

Review the following sections for conditions that may affect your system:

- Managing Source Journal File Purging
- Responding to Disabled Source Journaling
- Shadowing Class Compiles

Managing Source Journal File Purging

Caché does not do any special handling of journal file purging on the source for shadowing; therefore, it is your responsibility to configure the journal settings on the source to ensure the journal files that the shadow requires remain available until they are transmitted to the destination shadow.

This is usually only a concern if the shadow falls seriously behind the source; for example, if you suspend the shadow or stop it for a prolonged period of time.

Responding to Disabled Source Journaling

If journaling is disabled on the source, you must determine the best course of action to maintain a valid shadow. Most likely, you will have to resynchronize the shadow with the source after you resolve the condition that caused Caché to disable journaling.

See the *Journal I/O Errors* section of the "Journaling" chapter of this guide for more details.

Shadowing Class Compiles

Caché journals the database that contains the globals used for compiling classes. If you use shadowing and rely on the compile on the source to update the application code on the shadow, ensure that the default qualifier (compile with `/journal=1`) is not changed, so that each class compile is journaled and the updates transferred to the shadow database. If class compiles are not journaled, you cannot use the shadow for disaster recovery unless you recompile all your classes.

5.2.2 Configuring the Destination Shadow

To configure shadowing on a destination shadow server, first ensure that the destination system can make a TCP connection to the source system.

If you plan to use SSL, an SSL/TLS client configuration must exist on the destination. See *A Note on Caché Client Applications Using SSL/TLS* in the "Using SSL/TLS with Caché" chapter of the *Caché Security Administration Guide* for details.

Use the Management Portal from the Caché instance running on the destination system to configure the destination shadow properties, map and synchronize the databases in the shadow, and start shadowing. These procedures are described in the following sections:

- Define the Shadow

- Map the Databases
- Synchronize the Databases
- Start Shadowing

For information on methods and queries available for interfacing with the shadow destination without using the Management Portal, see the SYS.Shadowing.Shadow class documentation in the *InterSystems Class Reference*.

5.2.2.1 Define the Shadow

Navigate to the Shadow Server Settings page of the Management Portal (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**) and perform the following steps to define the shadow properties:

1. Click **Create Shadow Server** to define a shadow on this destination server.

   Important: You can shadow a failover mirror member only if it is the only failover member in the mirror; if a mirror has two failover members, you must shadow an async member instead. See Configuring Shadowing for more information about shadowing mirrored databases..

2. Enter an identifying name for this shadow in the **Name of the shadow** box. This value is also referred to as the *shadow ID*. The system uses this name to distinguish between shadow instances that may be running on the same system.

   Note: Do not use the tilde (~) character in the shadow name; it is used in internal shadow processing.

3. Enter the TCP/IP address or host name (DNS) of the source database server you are shadowing in the **DNS name or IP address of the source** box.
4. Enter the superserver port number of the source Caché instance you are shadowing in the **Port number of the source** box.

   Important: If you change the IP address or the port number on a suspended shadow, it is your responsibility to ensure the shadow can resume properly.

5. Click **Advanced** to enter the following optional fields:

   - **Journal file directory** — Enter the full name, including the path, of the journal file directory on the destination shadow system. Click **Browse** for help in finding the proper directory. Consider the following when updating this entry:
     - If you are shadowing more than one source instance on this destination, ensure you use a unique journal file directory for each instance.
     - If you change the journal file directory on a cluster shadow, the change only takes affect for journal files from new cluster nodes until you stop and restart the shadow.
   - **SSL Configuration** — Choose from the list of existing client configurations; leave this entry blank if you do not wish to use SSL for the shadow connection. The shadow connection to the source fails if either of the following two conditions exist:
     - The source does not support SSL, but you choose an SSL configuration.
     - The source requires SSL, but you do not choose an SSL configuration.
   - **Filter routine** — Enter the name (omit the leading ^) of an optional filter routine the shadow uses to filter journal records before dejournaling them on the shadow. The routine must be in the %SYS namespace and should take the following format

     ```
     MyShadowFilter(pid,dir,glo,type,addr,time)
     ```

Argument	Description
pid	Process ID of the record (If the record has a nontrivial remote system ID, the *pid* contains two fields delimited by a comma (,): the first field is the process ID and the second is the remote system ID.
dir	Source (not shadow) database directory
glo	Global reference in the form of global(subscripts), without the leading ^
type	Type of the record; valid values are: "S" (SET), "s" (BITSET), "K"(KILL), "k" (ZKILL)
addr	Offset of the record in the journal file
time	Timestamp of the record

In the filter routine logic, return 0 for the dejournaling process to skip the record; otherwise the shadow dejournals the record.

CAUTION: Perform the **New** command on any local variable in the filter routine to avoid accidentally overwriting the variables used in the shadow routine.

In the following example, the filter routine skips journal records for globals beginning with *X* (that is, ^*X**, where * is the wildcard) during the dejournaling process and logs each record that is dejournaled; then, if the journal record includes an *oldvalue* and *newvalue*, and they are identical, it skips the journal record.

```
MyShadowFilter(pid,dir,glo,type,addr,time) { ;Sample Shadow Filter Routine
   // Disclaimer: This routine is for illustration purpose only.

   If $Extract($Piece(glo,"(",1),1)="X" {
        Do ##class(%Library.Device).Broadcast("","Skip updating "_glo) ;log
        Quit 0
   }
   Do ##class(%Library.Device).Broadcast("",pid_","_dir_","_glo_","_type_","_addr_","_time)
;log
   // To retrieve more information about the record than provided --
   Set jrecoref=##class(%SYS.Journal.Record).%OpenId(addr) ;open the record
   If jrecoref.%IsA("%SYS.Journal.SetKillRecord") && (jrecoref.NumberOfValues=2)
&& (jrecoref.NewValue=jrecoref.OldValue) {
        Do ##class(%Library.Device).Broadcast("","Skip applying the SET") ;log
        Quit 0
   }
   Quit 1
 }
```

Important: Your shadow filter function should be simple and quick to prevent shadow latency. Ideally your shadow filter function will be a "pure" function, that is its results will depend only upon its inputs. The preceding example shows how to open and inspect the corresponding journal. This or any other deviation from a "pure" function should be taken with care, from both a performance and error handling perspective. You should take the following precautions:

- Use a *new* frame to avoid changing any local variable belonging to your calling frames, and assure that all your local variables are killed when your function exits.
- *Do not* use the **TSTART**, **TCOMMIT**, or **TROLLBACK** commands, or any command that indirectly involves transactions; for this reason, do not use dynamic SQL.
- Avoid **XECUTE** and indirection, and do not use **XECUTE** or indirection on untrusted strings.
- Do not call any routines or use any classes that do not follow these precautions.

Note: If you use dot syntax when referring to a global in your filter routine, you must use the leading ^.

You can specify the use of a filter routine in any of the following ways:

– From the Shadow Server Settings page (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**) when you choose to **Add a New Server** or **Edit** an existing shadow, enter the name in the **Filter routine** box in the **Advanced** settings.

– Via ^*SHADOW*, edit a shadow.

- **Days of old copied journals to keep** — Enter the number of days to keep the shadow copies of the source journal files. By default, the Caché destination purges its copy of a journal file as soon as it finishes dejournaling as long as it does not contain open transactions. You can keep the shadow copies of the journal files on the destination longer by entering a value in this field.

  For example, if you enter `3`, the shadow copy of a source journal file is eligible for purging if the source journal file is at least three days old, is completely dejournaled, and does not contain open transactions. The completion date of the source journal file determines its age.

- **Maximum error messages to keep** — Enter the number of shadowing errors from `0` to `200` which Caché should retain. The default is `10`.

- **Disable journaling of shadow updates** — To prevent local journaling of the updates that this shadow applies to the shadow databases, regardless of the journal settings on the databases themselves, change the default setting of **No** to **Yes**.

  InterSystems recommends that you journal all databases that are the destination of shadowing. This results in the destination shadow maintaining a journal of the shadow updates applied, which provides an additional level of redundancy.

  Important: Be careful not to place local journal files in the same directory as the journal files coming from the shadow source.

  CAUTION: If you choose not to journal shadow updates, the recommended method is to set **Disable journaling of shadow updates** to **Yes**, rather than disabling journaling for one or more of the destination shadow databases. If you do the latter and do not disable journaling on the destination's CACHESYS database, there is the possibility that if the shadow crashes and is restarted, the journal checkpoint stored in the ^*SYS* global in CACHESYS could cause the shadow to be recovered to a point *later* in the journal stream than the last record committed to the shadow databases, effectively skipping some shadow updates.

6. Click **Save** to return to the Shadow Server Settings page, where the new shadow is now listed. When you click the **Edit** link to edit the settings, the **Add database mapping** link is now included on the Edit Shadow Server page; see Map the Databases for details.

5.2.2.2 Map the Databases

After you successfully save the configuration settings, you can add or delete database mappings from the source to the shadow:

1. Next to **Database mapping for this shadow** click **Add** to associate the database on the source system with the directory on the destination system using the **Add Shadow Mapping** dialog box.
2. In the **Source database directory** box, enter the physical pathname of the source database file—the CACHE.DAT file. Enter the pathname of its corresponding destination shadow database file in the **Shadow database directory** box, and then click **Save**.
3. Verify any pre-filled mappings and click **Delete** next to any invalid or unwanted mappings. Shadowing requires at least one database mapping to start.

4. Click **Close** to return to the Shadow Server Settings page (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**).

If the source database server is part of a cluster, the configuration settings for the destination shadow differ slightly. For information on shadowing a clustered system, see the Cluster Shadowing section of the "Cluster Journaling" chapter of this guide.

Note: You can use the CACHESYS database as a source database of shadowing, provided that the target (shadow database) is not the CACHESYS database on the shadow. Currently the only way to add a database mapping containing the source manager's directory (CACHESYS) to a shadow configuration is by using the SYS.Shadowing.Shadow class API. For example:

ObjectScript

```
Set ShadowOref=##class(SYS.Shadowing.Shadow).%OpenId("MyShadow")
Do ShadowOref.SetDatabaseToShadow("C:\MyCache\Mgr","D:\MyCacheShdw\Shdwsys")
Set rc=ShadowOref.%Save()
```

Where C:\MyCache\Mgr is the source manager's directory for the CACHESYS database and D:\MyCacheShdw\Shdwsys is the directory for a database that is not the CACHESYS database on the destination. See the SYS.Shadowing.Shadow entry in the *InterSystems Class Reference* for details.

5.2.2.3 Synchronize the Databases

Before you start shadowing, synchronize the databases on the shadow destination with the source databases. Use an external backup on the source data server and restore the databases on the destination shadow. See the "Backup and Restore" chapter of this guide for more information.

Important: If the source and destination instances are on systems of different endianness (see "Platform Endianness" in the "Supported Technologies" section of the online *InterSystems Supported Platforms* document for this release), the database restored to the destination shadow must be converted to the endianness of the source before being used; see Considering Endianness in the "Backup and Restore" chapter for procedures.

5.3 Managing and Monitoring Shadowing

Caché provides an interface to shadow processing through the Management Portal. You can also configure and manage shadow processing using the **^SHADOW** utility or the SYS.Shadowing API classes, SYS.Shadowing.DataSource and SYS.Shadowing.Shadow. This document describes the procedures using the Management Portal and gives some examples of using the shadowing APIs.

A shadow can be in one of three states; depending on the state, you can perform different actions on the shadow. The following sections describe each state and action including the interrelationships among them.

5.3.1 Shadow States and Actions

A shadow can be in one of these states at any given time:

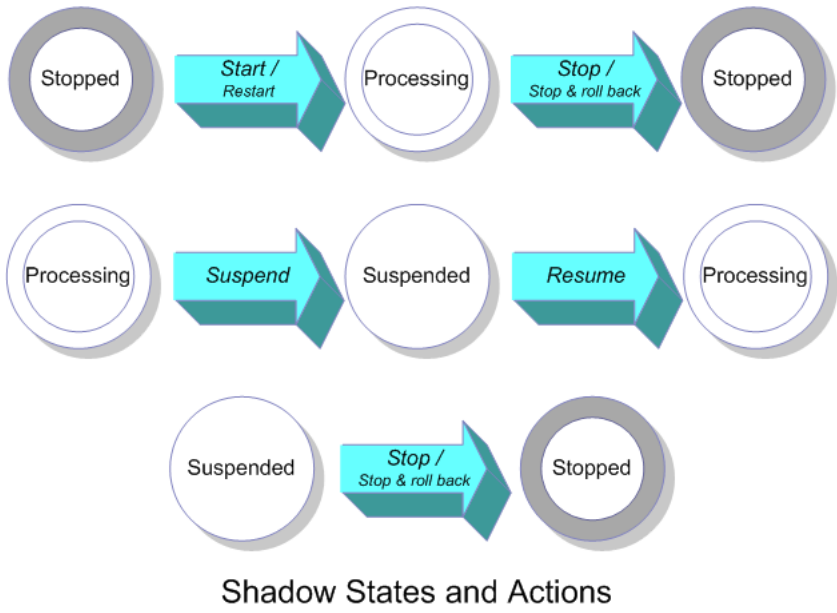
- *Stopped* — When a shadow is stopped, you can modify its properties. This is the initial state of a newly created shadow.
- *Processing* — When a shadow is running, it applies database updates and you cannot modify its properties.
- *Suspended* — When a shadow is suspended, it does not apply database updates but retains checkpoints (see *Shadow Checkpoints*). You can modify its properties, though some changes may not take effect immediately.

There are four types of allowable actions you can perform on a shadow, depending on its current state and your user privileges:

- *Start / Restart* — Starts a stopped shadow from the starting point specified using **Select Source Event** or, in the case of a restart, from the appropriate checkpoint.
- *Stop (with or without rollback)* — Stops a processing or suspended shadow. When you stop shadow processing, Caché offers you the choice whether or not to roll back any open transactions.
- *Suspend* — Suspends a processing shadow. Contrary to stopping a shadow, when you suspend a shadow, Caché maintains its open transactions, journal files, and checkpoints.
- *Resume* — Resumes a suspended shadow from where it left off, or from a selected checkpoint. When a fatal error occurs, a shadow aborts, entering the *suspended* state.

The following diagram shows the permissible actions on a shadow in each state. It indicates the shadow states with circles and shows the actions you can perform on these states with arrows.

Figure 5–2: Relationships of Shadow States and Permissible Actions



5.3.2 Shadow Processing Considerations

Keep the following conditions in mind when deciding when and how to change the state of a shadow:

- A stopped shadow does not start or restart automatically with a Caché restart; you must start or restart it explicitly as described in this chapter. Conversely, on Caché startup, a shadow that was *not* in stopped state in the previous Caché session resumes automatically.
- You cannot start a suspended shadow; you must either resume processing or stop the shadow and then start it.
- Avoid choosing to restart a shadow after you stop it with rollback. The shadow databases may be in an undetermined state until the shadow reaches the journal location of the last stop.
- When an individual database in the shadow destination fails or is dismounted, shadowing for other databases in the shadow continues. When this happens, therefore, you must catch up the affected database after restoring or mounting it using the procedures in Synchronizing or Resynchronizing a Destination Database.

There are two places in the Management Portal that you can perform tasks on a defined shadow:

- The *Shadow Administration Tasks* require system manager privileges. To get to the Create Shadow Server page, select **System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**.
- The *Shadow Operation Tasks* require operator privileges. To get to the Shadows page, select **System Operation** > **Shadow Servers** > **System as Shadow Server**.

See the individual method descriptions in the SYS.Shadowing.Shadow entry of the *InterSystems Class Reference* for details on performing these shadowing tasks programmatically.

5.3.3 Shadow Checkpoints

The system creates checkpoints periodically throughout the shadowing process. A *checkpoint* for the shadow is a location in the shadow copy of a source journal with the following implications:

1. All records at and prior to it are presumed to have been applied to the shadow databases.
2. It is safe, as far as database integrity is concerned, for the shadow to resume from the checkpoint after being suspended.

You can retrieve checkpoint information using the **CheckPointInfo** method of the SYS.Shadowing.Shadow class.

5.3.4 Shadow Administration Tasks

The Shadow Server Settings page (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**) lists each defined shadow with the name, status, source name and port, start point, filter, and choices for performing actions on the shadow configuration. Click the following options to perform the indicated task:

- **Edit** — Allows updates to the fields you entered when you added a new shadow. See *Define the Shadow* for descriptions of these settings. You cannot save edits if the shadow is processing.
- **Start** — Starts shadow processing from a start point you select; option available if the shadow is stopped. See *Start Shadowing* for details.
- **Restart** — Starts shadow processing from the appropriate checkpoint depending on whether or not you rolled back any open transactions when you stopped the shadow. See *Restart Shadowing* for details.
- **Stop** — Stops shadow processing; option available if the shadow is processing or suspended. Select the **Roll back open transactions** check box if you want to roll back any open transactions. See *Stop Shadowing* for details.
- **Delete** — Deletes the entire shadow definition; you must stop the shadow before deleting the shadow definition.

5.3.4.1 Start Shadowing

Once you add a shadow definition it appears in the list of shadows on the Shadow Server Settings page. You can start a shadow from this page:

1. Before starting the shadowing process, verify you have synchronized the databases you are shadowing on the source and destination and mapped the source databases to the corresponding destination databases.
2. Click **Start** in the row for the shadow name you want to start.
3. After verifying the location information for the source instance, click **Select Source Event** to choose where to begin shadowing. A page displays the available source events from the source journal file directory. You must select a source event before you can start shadowing. See Select a Source Event for details.

From the Management Portal, if you attempt to start a shadow that had been processing, you may see the following warning:

```
*** WARNING ***
There is a checkpoint from previous shadowing session.
You might want to RESTART the shadow from that
checkpoint instead.
If you do START, the checkpoint and any remaining
shadow copies of journal files from previous shadowing
session will be deleted.
Are you sure you want to start shadowing?
```

As this warning states, if you start a previously processing shadow, Caché clears all checkpoints and stores the start point you select as the first checkpoint. This process also purges any remaining shadow copies of the source journal files and fetches new copies form the source regardless of a possible overlap between the files. Ensure your new start point coincides with the state of the shadow databases.

If you run multiple shadows on an instance of Caché, see the *Generic Memory Heap Considerations* section for details on adjustments you may have to make.

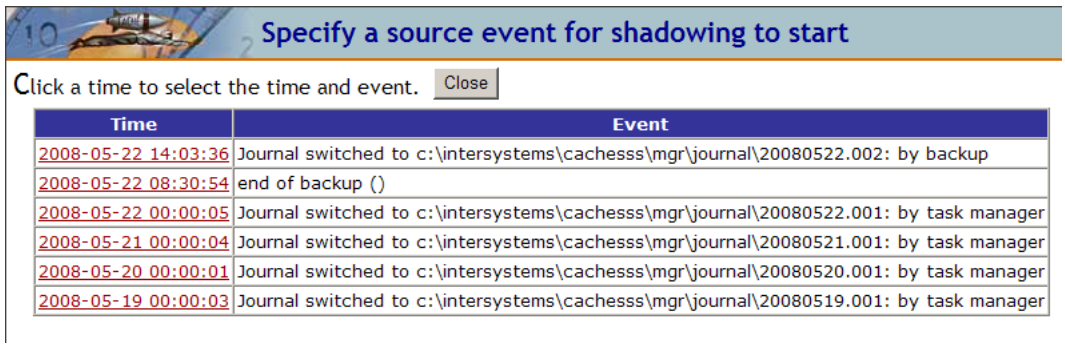
Important: Dismounting a database that is part of a running shadow (one that is in the **Processing** state) does not interrupt the shadow, meaning that other databases continue to be updated as the dismounted database falls behind. A dismounted shadow database causes a severe message to be posted to the console log and must be resynchronized with the shadow after being remounted, as described in Synchronizing or Resynchronizing a Destination Database.

Select a Source Event

While starting (or resuming) a destination shadow, you must select a source event from the journal files on the data source server where shadowing of the journaled databases should begin.

Click **Select Source Event** to display a list of journal events on the source database that are valid starting points for shadowing. From this list, click the time to specify at which source event shadowing starts. Choose the starting point after which you synchronized the databases on the source and destination.

For example, the system automatically switches the journal file after a successful backup. Before starting the shadowing process, synchronize the databases by restoring the successful backup file from the source on the destination shadow databases. On the shadow, click **Select Source Event** from the Shadow Server Settings page to see events listed similar to those in the following display:



For this example, to start shadowing at the point when the source backup ended successfully (the point of database synchronization), click the **Time** (`2008-05-22 08:30:54`), of the **Event** displaying `end of backup ()`.

Generic Memory Heap Considerations

The journal reader and database update processes on the shadow destination communicate via shared memory allocated from the generic memory heap size (also known as **gmheap**). Several processes in Caché use this memory and how Caché allocates it involves very complex interactions; therefore, Caché silently increases **gmheap** allocation during startup when necessary. In most cases, you should not have to manually adjust the allocation.

If you start multiple shadows at or near the same time while Caché is running, you may receive a **gmheap** allocation error. You can improve the allocation by starting the shadows as a group. If you start (or resume) multiple shadows one by one

consecutively, the first shadow to start uses about half of the free **gmheap** memory; the second, half of what remains; and so on. In contrast, if you start multiple shadows as a group, every shadow in the group uses 1/(*N*+1) of the free **gmheap** memory, where *N* is the number of the shadows in the group. Thus, starting multiple shadows as a group not only avoids the possible error allocating memory from **gmheap**, but also allocates memory evenly among the shadows.

See the **StartGroup** method in the SYS.Shadowing.Shadow entry of the *InterSystems Class Reference* for more information.

You can adjust the **gmheap** size from the Advanced Memory Settings page of the Management Portal (**System Administration** > **Configuration** > **Additional Settings** > **Advanced Memory**).

5.3.4.2 Stop Shadowing

When you stop shadowing you can choose to roll back or not to roll back any open transactions by selecting or clearing the **Roll back open transactions** check box.

Stop without Rollback

If you choose not to roll back, it is similar to suspending a shadow, but requires more privileges. You may choose this option if you want to maintain the current checkpoint, but you do not want Caché to automatically resume the shadow at restart, which does happen to a suspended shadow. For example, if you have to make configuration changes that require a Caché restart and additional changes after Caché is up, but before the shadow should start, use this option.

Stop with Rollback

This option is mainly for disaster recovery. The rollback sets the shadow databases to a logically consistent state, though out of sync with the source. Avoid restarting a shadow that you stopped and rolled back. You have the option open to you so that you can recover if it was a mistake to choose the rollback option; thus avoiding the need to resynchronize the shadow with the source.

CAUTION: Restarting a shadow that you stopped with rollback may leave the shadow in an indeterministic state until the shadow has progressed beyond the point of the pre-rollback state.

5.3.4.3 Restart Shadowing

If you stopped shadowing, in addition to the choice of starting the shadow, you can restart the shadow. Processing starts from the last checkpoint taken before you stopped the shadow if you chose not to roll back open transactions. If you chose to roll back when you stopped the shadow, shadow processing begins at the checkpoint prior to the earliest open transaction when the shadow stopped. A shadow restart reuses the journal files retained from the last time you stopped the shadow.

5.3.5 Shadow Operations Tasks

You can monitor the shadowing operation status from both the source and destination servers of the shadow. In the Management Portal, navigate to the **Shadow Servers** page (**System Operation** > **Shadow Servers**). On this page you choose the appropriate option depending on whether you are monitoring the shadowing process from the shadow side or the data-source side. The following sections detail the contents of each side:

- Managing the Destination Shadow
- Monitoring the Data Source
- Synchronizing a Destination Database

5.3.5.1 Managing the Destination Shadow

You can monitor and manage the shadow process from the destination shadow. The Shadows page (**System Operation** > **Shadow Servers** > **System as Shadow Server**) to display a list of source servers for this shadow machine and associated actions you can perform on each item, as described in the following table.

Field	Description
Name	Name of the shadow.
Status	One of three shadowing states described previously in this section: *stopped*, *processing*, *suspended*. You may see *trying to connect* as status when you initiate processing.
Checkpoint	Offset location in the shadow copy of the journal where it is safe to resume processing.
Errors	Number of errors reported on the shadow destination.
Open Transactions	Indicates whether or not there are open transactions on the shadow and, if so, how many.
Latency	Estimated time for the shadow to process the journal records that it copied from the source but has not yet applied to the shadow databases.

Click the following options to perform the indicated task:

- **Details** — displays selected details of this shadowing configuration.
- **Resume** — resumes shadow processing; option available if you have previously suspended shadow processing.
- **Suspend** — suspends shadow processing; option available if the shadow is processing.
- **Errors** — displays a list of errors occurring on the destination shadow. Caché retains the details of the number of errors you indicate in the configuration of the shadow. Click **Delete** to clear these errors and return to the list of shadows.

5.3.5.2 Monitoring the Data Source

You can also monitor the shadow process on the source system from the **Data Source** column of the **Shadow Servers** page (**System Operation** > **Shadow Servers**):

- Click **This System as Data Source** to display a list of shadows defined for this data source. The **Data Source** page (**System Operation** > **Shadow Servers** > **System as Data Source**) lists each defined shadow with the details described in the following table.

Field	Description
Port	Superserver port number of the Caché source instance (also shows process ID).
Shadow IP	IP address of the shadow destination machine.
Journal	Full directory path and file name of journal file currently being copied.
PID	Process ID number of the journal copying process.
Latency	Estimated time for the shadow to catch up copying the source journal file. This is not the latency of shadow dejournaling, which is available on the destination side.
Shadowing Rate	Rate in KBs per second that the shadow copies the source journal files.

- Display the **Data Source Errors** page (**System Operation** > **Shadow Servers** > **Error Log**), which lists errors reported on this data source.

You can also obtain this information programmatically. See the SYS.Shadowing.DataSource entry of the *InterSystems Class Reference* for details.

5.3.5.3 Synchronizing or Resynchronizing a Destination Database

For a variety of reasons, you may need to resynchronize a database that has fallen behind the other databases in the shadow (for example, because the destination database was dismounted for maintenance, or has been restored from backup). In addition, when you add a database to an existing shadow configuration, you must ensure that it is synchronized with the journal files being dejournaled by shadowing.

There are several options from which to choose depending on your needs.

Synchronizing Using a New Copy of a New or Existing Source Database

The simplest way to synchronize a shadow database is to suspend shadowing before making a backup of the source database that require synchronization. A database synchronized in this way will be in an inconsistent state on the shadow destination until the shadow catches up to the journal location corresponding to the creation of the backup; only the database being synchronized is affected. The disadvantage of this approach is that you may need to wait until an appropriate time to perform the backup of the source databases.

To synchronize a database in this way, use the following procedure:

1. Suspend (do not stop) the shadow: navigate to the Shadows page (**System Operation** > **Shadow Servers** > **System as Shadow Server**) as described in Managing the Destination Shadow and click **Suspend**.
2. If you are adding a new database to the shadow, create it on the destination and the source, then add the source database mapping to the shadow configuration, as described in Map the Databases.

   If you are synchronizing an existing source database, create a backup on the source and restore it on the destination. If you are adding an existing database on the source that was not previously included in the shadow, create a backup on the source and restore it on the destination, then add the source database mapping to the shadow configuration, as described in Map the Databases.

   Note: If you wish to use an older backup, you must use one of the following procedures.
3. Resume the shadow.

Synchronizing Using an Existing Copy of a Source Database

Rather than creating a new backup of the source databases as described in the previous procedure, you may want or need to use an existing copy of the database needing synchronization—that is, a version of the database older than the journal files currently being dejournaled by the shadow. For example, you may be restoring a damaged source database using a backup from an earlier time, adding a database on the source to the shadow under circumstances which prevent you from creating a new backup, or catching up a destination database that fell behind after being dismounted.

When synchronizing databases like these, you have several options. The simplest involves restarting the entire shadow from the source event representing the appropriate journal file—either the journal file corresponding to the backup you are restoring or the journal file that was being dejournaled when the destination database was dismounted. If you are catching up only one or a few databases out of many databases in the shadow, however, this option has the following disadvantages:

- The operation requires the time needed to dejournal all of the databases in the shadow, not just the databases being synchronized.
- The destination databases will be in an inconsistent state, and thus cannot be used, until all databases are caught up.
- If one of the necessary journal files is unavailable or damaged, the shadow will be unable to catch up, which requires that all databases be restored from a new backup of the shadow source.

Even with these disadvantages, if many databases require synchronization or the amount of journal to be applied is relatively modest, restarting the entire shadow from the appropriate journal file is typically preferred for its simplicity.

To synchronize by restarting the entire shadow from the appropriate journal file:

1. Suspend (do not stop) the shadow: navigate to the Shadows page (**System Operation** > **Shadow Servers** > **System as Shadow Server**) as described in Managing the Destination Shadow and click **Suspend**.
2. Restore the backup or mount the dismounted database.
3. If you restored a backup of a source database that is not yet in the shadow, add the source database mapping to the shadow configuration, as described in Map the Databases.
4. Start the shadow: navigate to the Shadow Server Settings page (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**) and start the shadow (as described in Start Shadowing), choosing as the source event (see Select a Source Event) either the journal file corresponding to the backup you restored or the journal file that was being dejournaled when the destination database was dismounted.

The alternative procedures involve more steps, but because journal data is applied only to the databases being synchronized, the disadvantages listed for the previous procedure are minimized. To catch up specific databases without synchronizing the entire shadow:

1. Suspend (do not stop) the shadow: navigate to the Shadows page (**System Operation** > **Shadow Servers** > **System as Shadow Server**) as described in Managing the Destination Shadow and click **Suspend**.
2. Restore the database(s) from backup, or mount the dismounted database(s).
3. Choose one of the following options:
   - Use the **^JRNRESTO** utility to restore the needed journal files to each database, starting with either the journal file corresponding to the backup you restored or the journal file that was being dejournaled when the destination database was dismounted. See Restore Globals from Journal Files Using ^JRNRESTO in the "Journaling" chapter of this guide for information about using this utility.
   - Configure and start an alternate shadow including only the affected databases, using the procedures described in Configuring Shadowing. Once the databases are caught up, stop the alternate shadow and then delete the alternate shadow configuration,
4. If you restored a backup of a source database that is not yet in the shadow, add the source database mapping to the shadow configuration, as described in Map the Databases.
5. Resume the shadow.

5.4 Using Shadowing for Disaster Recovery

As described in the "Mirroring" chapter of the *Caché High Availability Guide*, Caché mirroring with automatic failover provides an effective and economical high availability solution for planned and unplanned outages. Mirroring includes a full disaster recovery capability.

When mirroring is not in use, however, Caché shadowing is a good low-cost solution for off site disaster recovery, and may be used in conjunction with one of the numerous Caché-compatible high availability failover strategies provided by the makers of operating systems and computer hardware (see the "System Failover Strategies" chapter of the *Caché High Availability Guide*). Shadowing's benefits in disaster recovery include the following:

- Data loss is typically small.
- You can locate the shadow far away from the primary location.
- Time to recovery is typically only minutes.

This section covers the following topics:

- Planned production transfer to the shadow destination
- Disaster recovery using the shadow destination

5.4.1 Planned Production Transfer to the Shadow Destination

Shadowing is very suitable for planned temporary relocation of the production databases, for example to perform maintenance on the production host system, because it includes built-in mechanisms to allow shadow destinations to catch up when you perform a planned production cutover to a shadow destination.

To perform a planned transfer of production to a shadow destination, use the following procedure:

1. Halt application activity on the production server (shadow source).
2. Gracefully shut down the shadow source Caché instance, for example using the **ccontrol stop** command (see Controlling Caché Instances in the "Using Multiple Instances of Caché" chapter of the *Caché System Administration Guide*).

   When a shadow source is shutting down, the shutdown process waits for shadow destinations to receive all current journal files from the source before terminating the jobs servicing those shadow destinations. If the source encounters an error attempting to retrieve information about a destination, or does not get confirmation that all journal files have been received by a destination within the waiting period, a message is written to the console log (see Monitoring Log Files in the "Monitoring Caché Using the Management Portal" of the *Caché Monitoring Guide*). You can, therefore, confirm that the destination you intend to switch to is caught up with the source by checking the source's console log and confirming that there are no such messages pertaining to it.
3. Confirm that the shadow destination has finished dejournaling all journal data from the shadow source, then follow the procedure for stopping shadowing on the shadow destination.
4. Shut down Caché on the destination and do one of the following:
   - Change the IP address and fully qualified domain name (FQDN) of the shadow destination so that it exactly matches the shadow source your are transferring production from.
   - Change the web application DNS to point to the IP address of the shadow destination.
5. Restart Caché on the destination.
6. Resume application activity on the new production server (shadow destination).

If you are certain that all journal data was received from the original shadow source and fully dejournaled on the destination (that is, that there was no data loss) in the previous procedure, you can return to the original configuration when your planned outage is complete and the original production instance has been restarted by reversing the original direction of shadowing—that is, configuring the current production instance (former destination) as the shadow source and the former source as a destination—following the instructions in the Configuring Shadowing section as needed. You can then repeat the previous procedure in the opposite direction.

5.4.2 Disaster Recovery Using the Shadow Destination

Shadowing is a good mechanism for recovery from disk failure, database degradation due to hardware or software failure, or destruction of the primary physical plant. (Shadowing, however, cannot recover from malicious deletion of globals.) A Caché shadow server can apply journals from several dissimilar platforms on a small-scale server over any TCP network. Since shadowing conveys only logical updates to the destination, it eliminates the risk of propagating any structural problem. When deciding if Caché shadowing best suits your disaster recovery strategy, however, you should consider the following limitations when shadowing is interrupted by a failure of the production server:

- The shadow destination applies source journals asynchronously so as not to affect performance on the production server. This results in possible latency in data applied to the shadow destination, although it is generally seconds behind

at most. Consequently, if you use the shadow destination databases, they might be slightly out of date. This latency could increase if the shadow destination connection with the shadow source is lost for any sustained period. Caché provides mechanisms to monitor the state and progress of the shadow destination to help you determine the risk of using the destination databases during disaster recovery.

- Open transactions may remain. You can choose whether or not to roll back any incomplete transactions when you stop a shadow, which may depend on the state of the source journal files at the time of the disaster.

If your production/shadow source system functions as an application server, install identical applications on your disaster recovery shadow destination to speed recovery.

To use a shadow destination for disaster recovery, you can use the procedure in the previous section, assuming that the shadow source was not gracefully shut down but rather failed or became unavailable, and beginning with the step of confirming that the destination has finished dejournaling all journal data it received from the shadow source before the failure before following the procedure for stopping shadowing on the shadow destination. As previously noted, you can choose to roll back open transactions while stopping shadowing, and once you have stopped shadowing you can evaluate the risk of using the destination databases for disaster recovery.

Because of the near certainty of data loss under unplanned outage circumstances, you cannot simply reverse the direction of shadowing once the original shadow source is restored to operation to catch it up and to return to the original configuration, as described in the previous section. In this situation you must also fully resynchronize all databases, as describe in Synchronizing or Resynchronizing a Destination Database.

6

Cluster Journaling

This chapter contains information about journaling on ECP-based shared-disk clustered systems in Caché. It discusses the following topics:

- Journaling on Clusters
- Cluster Failover
- Cluster Shadowing
- Tools and Utilities

For related information, see the following chapters:

- "Backup and Restore" in this guide
- "Journaling" in this guide
- "Shadowing" in this guide
- Configuring Cluster Settings in the "Configuring Caché" chapter of the *Caché System Administration Guide*

6.1 Journaling on Clusters

Journaling is necessary for cluster failover to bring the databases up to date and to use transaction processing. Each node in a cluster maintains its own journal files, which must be accessible to all other nodes in the cluster to ensure recoverability. A *cluster session ID* (CSI) is the time when the session begins, that is, the cluster start time, which is stored in the header of every journal file on a clustered system.

In addition to the information journaled on a nonclustered system, the following specifics apply to clustered systems:

- Updates to clustered databases are always journaled, (usually on the master node only) except for scratch globals. On a cluster database, even globals whose database journaling attribute is `No` are journaled regardless of whether they are updated outside or within a transaction.
- Database updates via the **$Increment** function are journaled on the master node as well as on the local node if it is not the master.
- Other updates are journaled locally if so configured.

The journal files on clustered systems are organized using the following:

- Cluster Journal Log

- Cluster Journal Sequence Numbers

Important: The default location of the journal files is in the *install-dir*\Mgr\journal directory of the Caché instance. However, InterSystems recommends placing journal files on separate storage devices from those on which database files and the CACHE.WIJ file are located. See Journaling Best Practices in the "Journaling" chapter of this guide for more information about separating journal storage.

CAUTION: Do *not* stop journaling in a cluster environment, although it is possible to do so with the **^JOURNAL** routine. If you do, the recovery procedure is vulnerable until the next backup.

6.1.1 Cluster Journal Log

Journal files used by members of a cluster are logged in a file, CACHEJRN.LOG, located in the cluster pre-image journal, or PIJ directory. It contains a list of journal files maintained by nodes while they are part of the cluster. Journal files maintained by a node when it is *not* part of the cluster may not appear in the cluster journal log; for more information, see Journal History Log in the "Journaling" chapter of this guide.

Following is an example of part of a cluster journal log:

```
0,_$1$DRA1:[TEST.50.MGR.JOURNAL]20030913.004
1,_$1$DKA0:[TEST.50.MGR.JOURNAL]20030916.002
0,_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.001
0,_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.002
1,_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.002
1,_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.003
```

The first value in each comma-delimited row is the cluster system number (CSN) of the node to which the journal file, the second field, belongs. The log is useful for locating journal files of all the members of a cluster, especially the members that have left the cluster. The CSN of a node may change when it restarts.

When a node joins the cluster its current journal file is added to the journal log. Processes that start journaling or switch journal files also add entries. The log is used in a cluster journal restore, by shadowing, and by the journal dump utility.

6.1.2 Cluster Journal Sequence Numbers

InterSystems recommends locking globals in cluster-mounted databases if you require a record of the sequence of updates. This is the tool Caché uses to record the time-sequencing of updates in the journal files of the cluster nodes. If cluster failover occurs, the journals of all the nodes can be applied in the proper order. Updates may not be dejournaled in the same order as they originally occurred, but they are valid with respect to the synchronization guaranteed by the Lock command and $Increment function.

To restore clustered databases properly from the journal files, the updates must be applied in the order they occurred. The cluster journal sequence number, which is part of every journal entry of a database update, and the cluster session ID, which is part of the journal header, provide a way to sequence transactions from the journal files of all the cluster members. For each cluster session, the sequence number starts at `1` and can go as high as `18446744073709551619` (that is, `2**64-1`).

The master node of a cluster maintains a master copy of the sequence number, which is incremented during database mounting, and with each use of **Lock** and **$Increment**. The master value of the sequence propagates to one or all cluster nodes, depending on the type of operation.

The sequence number is used by cluster journal restore and shadowing, which is a special form of journal restore. Both utilities operate on the assumption that the sequence number increases monotonically during a cluster session.

At the end of a backup, the cluster journal sequence number on all cluster nodes is incremented to be higher than the previous master cluster journal sequence number. Then a journal marker bearing the new cluster journal sequence number is placed in the current local journal file. In a sense, the journal marker serves as a barrier of cluster journal sequence numbers, sep-

arating the journaled database updates that are covered in the backup from those that are not. Following the restore of the backup, cluster journal restore can start from the cluster journal sequence number of the journal marker and move forward.

You can also set your own journal markers using the **^JRNMARK** utility. See Setting Journal Markers on a Clustered System for details.

6.2 Cluster Failover

The Caché cluster failover process protects the integrity of data on other cluster nodes when one cluster member fails. It allows the remaining cluster members to continue to function. The following conditions must be met for cluster failover to work successfully:

- All directories containing CACHE.DAT files must be accessible to all surviving nodes.
- Journaling must be enabled at all times while Caché is running.
- Networking must be properly configured.

If a cluster member fails, the cluster master executes cluster failover. If the master is the failing node, the cluster member that least recently joined the cluster becomes the new master and executes failover. Cluster failover consists of two phases.

In the first phase, the cluster master does the following:

- Checks the cluster PIJ and the write image journal files (CACHE.WIJ) on each node to determine what recovery is needed from these files.
- Executes recovery from the WIJ files to all databases that had been mounted in cluster mode.

If an error occurs during this phase, the cluster crashes and further Cluster Recovery must take place.

In the second phase, the cluster master does the following:

- Mounts databases in private mode, as required, to restore the journals of all cluster members.
- Attempts to mount the databases in cluster mode if it cannot mount them in private mode.
- Restores any Caché journal entries after the current index kept in the CACHE.WIJ file for each cluster member's journal. For details, see Cluster Restore.
- Rolls back incomplete transactions in the failed node's journal file.
- Reforms the lock table if it is the new cluster master; otherwise, it discards the locks of the failing node.

During failover, the journals from all cluster members are applied to the database and any incomplete transactions are rolled back. If cluster failover completes successfully, there is no database degradation or data loss from the surviving nodes. There is only minimal data loss (typically less than the last second) not visible to other cluster members from the failing node.

If failover is unsuccessful, the cluster crashes and you must shut down all cluster nodes before restarting the cluster. See the Failover Error Conditions section for more details.

6.2.1 Cluster Recovery

Recovery occurs when Caché stops on any cluster member. The procedure changes depending on how Caché stops. During successful failover, the recovery procedure is fairly straightforward and automatic. If, however, a clustered system crashes, the recovery is more complex.

Following a cluster crash, a clustered Caché node cannot be restarted until all of the nodes in the cluster have been stopped. If a cluster member attempts Caché startup or tries to join the cluster by cluster-mounting a disk before all other cluster members have been stopped, the following message is displayed:

```
ENQ daemon failed to start because cluster is crashed.
```

Once all members are stopped, start each node. The first node that starts runs the Caché recovery procedure if it detects there was an abnormal system shutdown. This node becomes the new cluster master. Cluster members that are not the cluster master are frequently referred to as slave nodes.

The Recovery daemon (**RCVRYDMN**) performs recovery on the surviving or new master node based on whether the crashed node was the master or a slave. To enable recovery, the node's databases and WIJ files must be accessible cluster-wide. The master is responsible for managing the recovery cluster-wide based on the WIJ file on each node.

When a slave crashes, the Recovery daemon on the master node does the following:

1. It uses the journal information provided by the WIJ files to apply journal files on all cluster nodes (including the one that crashed) from a common starting point, as with all cluster journal restores.
2. It rolls back all incomplete transactions on the crashed system. Again, the rollbacks are journaled, this time on the host system of the Recovery daemon. For this reason, if you restore journal files, it is safer to do a cluster journal restore than a stand-alone journal restore, as the rollback of an incomplete transaction in one node's journal may be journaled on another node.

When the (former) master crashes, the Recovery daemon does the following:

1. As in step 1 above, it applies the journal files on all the cluster nodes.
2. In between the two steps above, it adjusts the cluster journal sequence number on its host system, which is the new master, so that it is higher than that of the last journaled entry on the crashed system, which was the old master. This guarantees the monotonic increasing property of cluster journal sequence in cluster-wide journal files.
3. As above, all incomplete transactions are rolled back on the crashed system.

   If the last remaining node of a cluster crashes, restarting the first node of the cluster involves cluster journal recovery, which includes rolling back any transactions that were open (uncommitted) at the time of crash.

6.2.2 Cluster Restore

Typically, journal files are applied after backups are restored to bring databases up to date or up to the point of a crash. If nodes have not left or joined a cluster since the last backup, you can restore the journal files starting from the marker corresponding to the backup. If one or more nodes have joined the cluster since the last backup, the restore is more complex.

A node joins a cluster either when it restarts with a proper configuration or cluster mounts a database after startup (as long as you properly set up other parameters, such as the PIJ directory, at startup). To make journal restore easier in the latter case, switch the journal file on the node as soon as it joins the cluster.

For each node that has joined the cluster since the last backup of the cluster:

1. Restore the latest backup of the node.
2. If the backup occurred before the node joined the cluster, restore the private journal files from where the backup ends, up to the point when it joined the cluster. (You can make this easier by switching the journal file when the node joins the cluster.)
3. Restore the latest cluster backup.
4. Restore the cluster journal files starting from where the backup ends.

See Cluster Journal Restore for detailed information about running the utility.

This procedure works well for restoring databases that were privately mounted on nodes before they joined the cluster and then are cluster-mounted after that node joined the cluster. It is based on the following assumptions:

- A cluster backup covers all cluster-mounted databases and a system-only backup covers private databases that, by definition, are not accessible to other systems of the cluster.
- The nodes did not leave and rejoin the cluster since the last cluster backup.

In more complicated scenarios, these assumptions may not be true. The first assumption becomes false if, say, rather than centralizing backups of all cluster-mounted databases on one node, you configure each node to back up selected cluster-mounted databases along with its private databases.

In this case, you may have to take a decentralized approach by restoring one database at a time. For each database, the restore procedure is essentially the same:

1. Restore the latest backup that covers the database.
2. Restore the private journal files up to the point when the node joined the cluster, if it postdates the backup.
3. Restore the cluster journal files from that point forward.

CAUTION: Even if the database has always been privately mounted on the same node, it is safer to restore the cluster journal files than to apply only the journal files of that node. If the node crashed or was shut down when it was part of the cluster, open transactions on the node would have been rolled back by and journaled on a surviving node of the cluster. Restoring the cluster journal files ensures that you do not miss such rollbacks in journal files of other nodes.

InterSystems does not recommend or support the scenario where a node joins and leaves the cluster multiple times.

6.2.3 Failover Error Conditions

When a cluster member fails, the other cluster members notice a short pause while failover occurs. In rare situations, some processes on surviving cluster nodes may receive <CLUSTERFAILED> errors. You can trap these errors with the **$ZTRAP** error-trapping mechanism.

Cluster failover does not work if one of the following is true:

- One or more cluster members go down during the failover process.
- There is disk drive failure and the first failover phase encounters an error.
- One of the surviving cluster members does not have a Recovery daemon.

If failover is unsuccessful, the cluster crashes and the following message appears at the operator's console:

```
****** Caché : CLUSTER CRASH - ALL Caché SYSTEMS ARE SUSPENDED ******
```

The other cluster members freeze when Caché processes reach a **Set** or **Kill** command.

Examine the failover log, which is contained in the console log (normally cconsole.log in the manager's directory) of the cluster master, to see the error messages generated during the failover process.

If a cluster member that failed attempts startup, or a node tries to join the cluster by cluster-mounting a database while the cluster is in failover, the following message is displayed:

```
The cluster appears to be attempting to recover from the failure of one or more
members at this time. Waiting 45 seconds for failover to complete...
```

A period (.) appears every five seconds until the active recovery phase completes. The mount or startup then proceeds.

If the cluster crashes during this time, the following message is displayed:

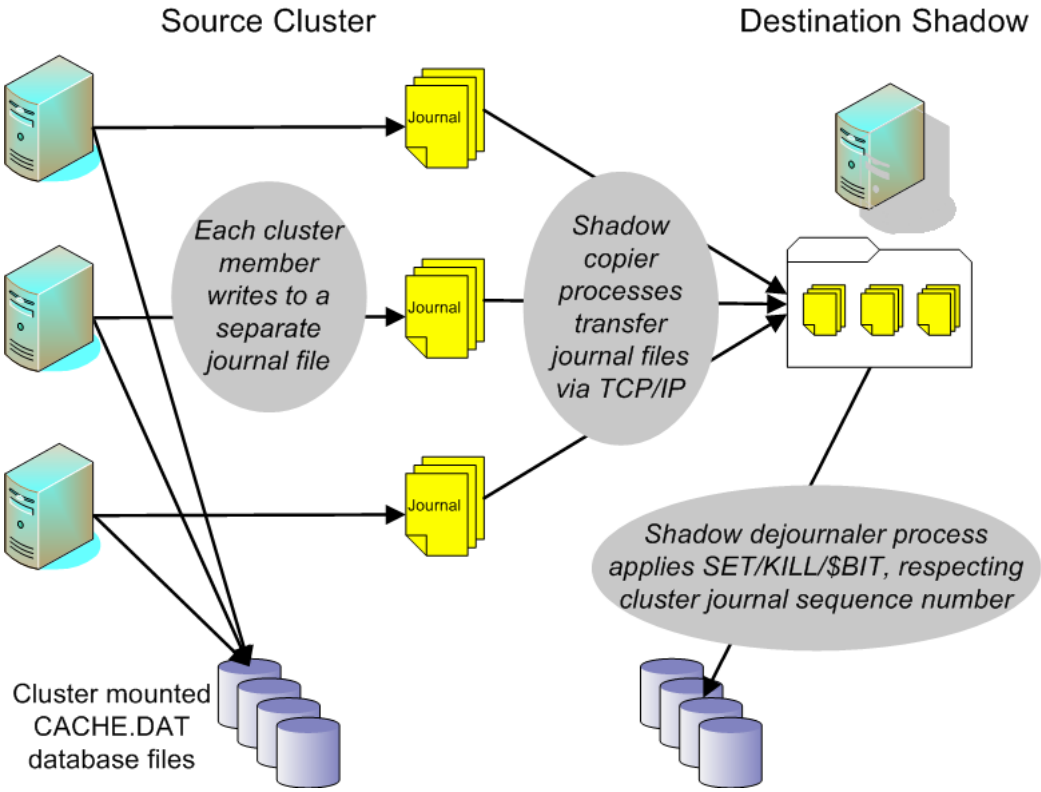
```
ENQ daemon failed to start because cluster is crashed.
```

See the Cluster Recovery section for an explanation of what happens when a cluster crashes.

6.3 Cluster Shadowing

The use of journaling in a clustered system also makes it possible to shadow a Caché cluster. In a cluster, each node manages its own journal files, which contain data involving private or cluster-mounted databases. The shadow mirrors the changes to the databases (assuming all changes are journaled) on a Caché system that is connected to the cluster via TCP. The following diagram gives an overview of the cluster shadowing process:

Figure 6–1: Cluster Shadowing Overview



The destination shadow connects to the specified Caché superserver on the cluster, requesting a list of journal files at or after the specified start location (the combination of cluster start time and cluster journal sequence number), one starting file for each cluster member.

For each node (with a unique CSN) returned from the source cluster, the shadow starts a copier process that copies journal files, starting with the file returned, from the server to the shadow. Each copier acts as a semi-independent shadow itself, similar to a nonclustered block-mode shadow.

Once all copiers are up and running, the cluster shadow starts a dejournaling process that applies journal entries from the copied journal files to the databases on the shadow side respecting the cluster journal sequence numbers of each journal record. The cluster shadow maintains a list of current live members (including port numbers and IP addresses) of the cluster which it receives from the source cluster.

The following sections describe what information is necessary and the procedures involved in setting up a cluster shadow as well as the limitations to the completeness and timeliness of the shadow databases:

- Configuring a Cluster Shadow
- Cluster Shadowing Limitations

Note: The shadow does not have to be a clustered system. The word "cluster" in cluster shadowing refers to the source database server, not the shadow.

6.3.1 Configuring a Cluster Shadow

You must provide several types of information to properly configure a cluster shadow. An overview of the required data items is divided into the following categories:

Establishing a Connection

Although a cluster is identified by the PIJ directory that all its member nodes share, the uniqueness of the identifier does not go beyond the physical cluster that hosts the Caché cluster. The shadow needs a way to make a TCP connection to the source cluster; therefore, on the shadow you must specify the IP address or host name of one member of the Caché cluster and the port number of the superserver running on that member. Also provide the shadow with a unique identity to distinguish it from other shadows, if any, in the same Caché instance.

Identifying the Starting Location

Configure the shadow to identify the journal starting location— a cluster start time (CSI) and, optionally, a cluster journal sequence number— for dejournaling. If you do not specify a cluster journal sequence number, dejournaling starts at the beginning of the cluster session.

Copying Journal Files

Similar to noncluster shadowing, specify a directory to put the journal files copied over from the cluster. However, a single directory is not adequate; journal files from different members of the cluster must be kept separate. The directory you specify serves as the parent of the directories for the shadow copies of the journal files. In fact, the shadow creates directories on the fly to keep up with the dynamic nature of the cluster components.

At run time, for each journal directory on the server cluster, the shadow sets up a distinct subdirectory under the user-specified parent directory and copies journal files from a journal directory on the server to its corresponding directory on the shadow—this is called *redirection of journal files*. The subdirectories are named by sequential numbers, starting with `1`. You cannot override a redirection by specifying a different directory on the shadow for a journal directory on the server.

Redirecting Dejournaled Transactions

As with nonclustered shadowing, specify database mapping, or redirections of dejournaled **Set** and **Kill** transactions.

There are two ways to provide the information to set up a cluster destination shadow:

- Using the Management Portal
- Using Caché Routines

6.3.1.1 Using the Management Portal

You can configure a shadow server using the Management Portal. Perform the following steps:

1. From the Shadow Server Settings page of the Managerment Portal (**System Administration** > **Configuration** > **Connectivity** > **Shadow Server Settings**), follow the procedure described in the Configuring the Destination Shadow section in the "Shadowing" chapter. Use the following specifics particular to cluster shadowing:

   a. **Database Server** — Enter the IP address or host name (DNS) of one member of the source cluster to which the shadow will connect.

   b. **Database Server Port #** — Enter the port number of the source specified in the previous step.

2. After entering the location information for the source instance, click **Select Source Event** to choose where to begin shadowing. A page displays the available cluster events from the cluster journal file directory.
3. Click **Advanced** to fill in the **Journal file directory** field. Enter the full name, including the path, of the journal file directory on the destination shadow system, which serves as the parent directory of shadow journal file subdirectories, created automatically by the shadow for each journal directory on the source cluster. Click**Browse** for help in finding the proper directory.
4. After you successfully save the configuration settings, add database mapping from the cluster to the shadow.
5. Next to **Database mapping for this shadow** click **Add** to associate the database on the source system with the directory on the destination system using the **Add Shadow Mapping** dialog box.
6. In the **Source database directory** box, enter the physical pathname of the source database file—the CACHE.DAT file. Enter the pathname of its corresponding destination shadow database file in the **Shadow database directory** box, and then click **Save**.
7. Verify any pre-filled mappings and click **Delete** next to any invalid or unwanted mappings. Shadowing requires at least one database mapping to start.
8. Start shadowing.

6.3.1.2 Using Caché Routines

You can also use the shadowing routines provided by Caché to configure the cluster shadow. Each of the examples in this section uses the technique of setting a return code as the result of executing the routine. After which you check the return code for an error or `1` for success.

To initially configure the cluster shadow:

ObjectScript

```
Set rc=$$ConfigCluShdw^SHDWX(shadow_id,server,jrndir,begloc)
```

Where ...	is ...
shadow_id	A string that uniquely identifies the shadow
server	Consists of the superserver port number and IP address or host name of one cluster node, delimited by a comma
jrndir	Parent directory of shadow journal file subdirectories, where journal files fetched from the source cluster are stored on the destination shadow, one subdirectory for each journal directory on the source cluster
begloc	Beginning location consisting of a cluster session ID (cluster startup time, in **YYYYMMDD HH:MM:SS** format) and a cluster sequence number, delimited by a comma

You can run the routine again to change the values of *server*, *jrndir*, or *begloc*. If you specify a new value for *jrndir*, only *subsequent* journal files fetched from the source are stored in the new location; journal files already in the old location remain there.

You can also direct the shadow to store journal files fetched from the journal directory, *remdir*, on the source in a local repository, *locdir*, instead of the default subdirectory of *jrndir*. Again, this change affects only journal files to be fetched, not the journal files that have been or are being fetched.

ObjectScript

```
Set rc=$$ConfigJrndir^SHDWX(shadow_id,locdir,remdir)
```

The last mandatory piece of information about a shadow, **Set** and **Kill** transaction redirection, can be given as follows:

ObjectScript

```
Set rc=$$ConfigDbmap^SHDWX(shadow_id,locdir,remdir)
```

This specifies that **Set** and **Kill** transactions from the source directory, *remdir*, be redirected to the shadow directory, *locdir*. Unlike journal files, there is no default redirection for a source database—if it is not explicitly redirected, **Set** and **Kill** transactions from that database are ignored by the dejournaling process of the shadow.

Finally, to start and stop shadowing:

ObjectScript

```
Set rc=$$START1^SHDWCLI("test")
Set rc=$$STOP1^SHDWCLI("test")
```

See Shadow Information Global and Utilities for more information.

6.3.2 Cluster Shadowing Limitations

There are a few limitations in the cluster shadowing process.

Database updates that are journaled outside a cluster are not shadowed. Here are two examples:

- After a cluster shuts down, if a former member of the cluster starts up as a stand-alone system and issues updates to some (formerly clustered) databases, the updates do not appear on the shadow.
- After a formerly stand-alone system joins a cluster, the new updates made to its private databases appear on the shadow (if they are defined in the database mapping), but none of the updates made before the system joined the cluster appear. For this reason, joining a cluster on the fly (by cluster-mounting a database) should be planned carefully in coordination with any shadow of the cluster.

In cluster shadowing, there is latency that affects the dejournaler. Journal files on the destination shadow side are not necessarily as up to date as what has been journaled on the source cluster. The shadow applies production journals asynchronously so as not to affect performance on the production server. This results in possible latency in data applied to the shadow.

Only one Caché cluster can be the target of a cluster shadow at any time, although there can be multiple shadows on one machine. There is no guarantee regarding the interactions between the multiple shadows, thus it is the user's responsibility to ensure that they are mutually exclusive. If you have a single instance with multiple shadows pointing at different clusters, it is your responsibility to ensure that the shadow datasets are mutually exclusive. For example, assume you have two clusters (A and B) shadowed on a single machine: cluster A has datasets X, Y, and Z; cluster B has datasets L, M, and N. Since Caché does not guarantee that datasets X (cluster A) and L (cluster B) are mutually exclusive during the dejournaling process (that is, that one shadow is dejournaling dataset X while the other is dejournaling dataset L), it is your responsibility to ensure that the datasets do not overlap.

Note: Exclude Caché databases from RTVScan when using Symantec Antivirus software to avoid the condition of the cluster shadow hanging on Windows XP. For detailed information, see the Symantec documentation.

6.4 Tools and Utilities

The following tools and utilities are helpful in cluster journaling processes:

- Cluster Journal Restore — **^JRNRESTO**
- Journal Dump Utility — **^JRNDUMP**
- Startup Recovery Routine — **^STURECOV**
- Setting Journal Markers on a Clustered System — **^JRNMARK**
- Cluster Journal Information Global — *^%SYS("JRNINFO")*
- Shadow Information Global and Utilities — *^SYS("shdwcli")*

6.5 Cluster Journal Restore

The cluster journal restore procedure allows you to start or end a restore using the journal markers placed in the journal files by a Caché backup. You can run a cluster journal restore either as part of a backup restore or as a stand-alone procedure.

Caché includes an entry point to the journal restore interface for performing specific cluster journal restore operations. From the %SYS namespace, run the following:

ObjectScript

```
Do CLUMENU^JRNRESTO
```

This invokes a menu that includes the following options:

1. Perform a cluster journal restore.
2. Generate a common journal file from specific journal files.
3. Perform a cluster journal restore after a backup restore.
4. Perform a cluster journal restore based on Caché backups.

6.5.1 Perform a Cluster Journal Restore

The first option of the cluster journal restore menu allows you to run the general journal restore on a clustered or nonclustered system. It is the equivalent of running **^JRNRESTO** and answering **Yes** to the `Cluster journal restore?` prompt.

```
%SYS>Do ^JRNRESTO

This utility uses the contents of journal files
to bring globals up to date from a backup.
Replication is not enabled.

Restore the Journal? Yes => Yes

Cluster journal restore? Yes
```

You are asked to describe the databases to be restored from the journal and the starting and ending points of the restore. The starting and ending points can be based on a backup, on a set of journal markers, at a cluster start, or any arbitrary point in the journal.

The interface prompts for directory information, including any redirection specifics, and whether all databases and globals are to be processed. For example:

```
Directory: _$1$DKB300:[TEST.CLU.5X]
Redirect to Directory: _$1$DKB300:[TEST.CLU.5X] => _$1$DKB300:[TEST.CLU.5X]
                                                --> _$1$DKB300:[TEST.CLU.5X]
Restore all globals in _$1$DKB300:[TEST.CLU.5X]? Yes => Yes

Directory:
```

For each directory you enter you are asked if you want to redirect. Enter the name of the directory to which to restore the dejournaled globals. If this is the same directory, enter the period (.) character or press **Enter**.

Also specify for each directory whether you want to restore all journaled globals. Enter **Yes** or press **Enter** to apply all global changes to the database and continue with the next directory. Otherwise, enter **No** to restore only selected globals.

At the `Global^` prompt, enter the name of the specific globals you want to restore from the journal. You may select patterns of globals by using the asterisk (*) to match any number of characters and the question mark (?) to match any single character. Enter `?L` to list the currently selected list of globals.

When you have entered all your selected globals, press **Enter** at the `Global^` prompt and enter the next directory. When you have entered all directories, press **Enter** at the `Directory` prompt. Your restore specifications are displayed as shown in this example:

```
Restoring globals from the following clustered datasets:
 1. _$1$DKB300:[TEST.CLU.5X]   All Globals

Specifications for Journal Restore Correct? Yes => Yes
Updates will not be replicated
```

Verify the information you entered before continuing with the cluster journal restore. Answer **Yes** or press **Enter** if the settings are correct; answer **No** to repeat the process of entering directories and globals.

Once you verify the directory and global specifications, the `Main Settings` menu of the cluster journal restore setup process is displayed with the current default settings, as shown in the following example:

```
Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DKB400:[TEST.5X]CACHEJRN.LOG
   with NO redirections of journal files
2. To START restore at the beginning of cluster session <20030319 15:35:37>
3. To STOP restore at sequence #319 of cluster session <20030319 15:35:37>
      134388,_$1$DRA2:[TEST.5Y.JOURNAL]20030320.005
4. To SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions

Select an item to modify ('Q' to quit or ENTER to accept and continue):
```

From this menu you may choose to modify any of the default values of the five settings by entering its menu item number:

1. Change the source of the restore.
2. Change the starting point of the restore.
3. Change the ending point of the restore.
4. Toggle the switching journal file setting.
5. Toggle the disable journaling setting.

After each modification, the `Main Settings` menu is displayed again, and you are asked to verify the information you entered before the restore begins. The following is an example of how the menu may look after several changes:

```
Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DKB400:[TEST.5Y.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DKB400:[TEST.5X.JOURNAL] -> _$1$DRA2:[TEST.5X.JOURNAL]
      _$1$DKB400:[TEST.5Y.JOURNAL] -> _$1$DRA2:[TEST.5Y.JOURNAL]
      _$1$DKB400:[TEST.5Z.JOURNAL] -> _$1$DRA2:[TEST.5Z.JOURNAL]
2. To START restore at the journal marker located at
      138316,_$1$DKB400:[TEST.5X.JOURNAL]20030401.001
          -> _$1$DRA2:[TEST.5X.JOURNAL]20030401.001
3. To STOP restore at the journal marker located at
      133232,_$1$DKB400:[TEST.5X.JOURNAL]20030401.003
          -> _$1$DRA2:[TEST.5X.JOURNAL]20030401.003
4. NOT to SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions

Select an item to modify ('Q' to quit or ENTER to accept and continue):
Start journal restore?
```

Press **Enter** to accept the settings and continue. If you are using the journal log of the current cluster, you are informed that the restore will stop at the currently marked journal location and asked if you want to start the restore.

```
Select an item to modify ('Q' to quit or ENTER to accept and continue):

To stop restore at currently marked journal location
  offset 134168 of _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008

Start journal restore?
```

Enter **`Yes`** to begin the cluster journal restore. Once the restore finishes your system is ready for activity.

Enter **`No`** to go back to the main menu where you can continue to make changes to the cluster journal restore setup or enter **`Q`** to abort the cluster journal restore. After aborting the cluster journal restore, you can run a private journal restore or abort the restore process entirely.

```
Select an item to modify ('Q' to quit or ENTER to accept and continue): Q
Run private journal restore instead? No

[Journal restore aborted]

Replication Enabled
```

6.5.1.1 Change the Source of the Restore

The first item on the `Main Settings` menu contains the information required to find the journal files for all the cluster members. The information has two elements:

- Cluster journal log — a list of journal files and their original full paths.
- Redirection of journal files — necessary only if the system where you are running the restore is not part of the cluster associated with the journal log.

By default, the restore uses the cluster journal log associated with the current clustered system. If you are running the restore on a nonclustered system, you are prompted for a cluster journal log before the main menu is displayed.

Choose this option to restore the journal files on a different Caché cluster from the one that owns the journal files. You can either:

- Identify the cluster journal log used by the original cluster.
- Create a cluster journal log that specifies where to locate the journal files.

Note: The option to redirect the journal files is available only if the specified cluster journal log is not that of the current cluster.

Identify the Cluster Journal Log

The Journal File Information menu displays the cluster journal log file to be used in the cluster journal restore. If the journal files on the original cluster are not accessible to the current cluster, copy them to a location accessible to the current cluster and specify how to locate them by entering redirect information.

Enter **I** to identify the journal log used by the original cluster.

```
Select an item to modify ('Q' to quit or ENTER to accept and continue): 1

Cluster Journal Restore - Setup - Journal File Info
[I]dentify an existing cluster journal log to use for the restore
   Current: _$1$DRA1:[TEST.50]CACHEJRN.LOG
  - OR -
[C]reate a cluster journal log by specifying where journal files are


Selection ( if no change): I

                              *** WARNING ***
        If you specify a cluster journal log different from current one, you
        may need to reenter info on journal redirection, restore range, etc.

Enter the name of the cluster journal log ( if no change)
=> cachejrn.txt

Cluster Journal Restore - Setup - Journal File Info
[I]dentify an existing cluster journal log to use for the restore
   Current: _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
[R]edirect journal files in _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
  - OR -
[C]reate a cluster journal log by specifying where journal files are
```

You must redirect journal files if the journal files being restored are not in their original locations, as specified in the cluster journal log. To redirect the journal files listed in the cluster journal log, provide the original and current locations when prompted. You may give a full or partial directory name as an original location. All original locations with leading characters that match the partial name are replaced with the new location. An example of redirecting files follows:

```
Selection ( if no change): R
Journal directories in _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
      _$1$DRA1:[TEST.50.MGR.JOURNAL]
      _$1$DRA1:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL]
Enter the original and current locations of journal files (? for help)
Journal files originally from: _$1$DRA1:
are currently located in: _$1$DRA2:
    _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
    _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
    _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
Journal files originally from:

Cluster Journal Restore - Setup - Journal File Info
[I]dentify an existing cluster journal log to use for the restore
   Current: _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
[R]edirect journal files in _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
  - OR -
[C]reate a cluster journal log by specifying where journal files are

Selection ( if no change):
```

This example shows the choice of an alternative cluster journal log, CACHEJRN.TXT, which contains a list of journal files originally located on _1DRA1:. These files are redirected to be retrieved from their new location, _1DRA2:, during the restore.

When you have finished entering the redirection information, press **Enter** to return to the Main Settings menu.

Journal redirection assumes a one-to-one or many-to-one relationship between source and target directory locations. That is, journal files from one or multiple original directories may be located in one new location, but not in multiple new locations.

To restore from journal files that are in multiple new locations, create a cluster journal log that specifies where to locate the journal files.

Create a Cluster Journal Log

If the journal files on the original cluster are not accessible to the current cluster, create a cluster journal log that specifies the locations of the journal files. The files in the specified locations must all be part of the cluster. Copy them to a location accessible to the current cluster and specify how to locate them by entering redirect information.

```
Selection ( if no change): C

                              *** WARNING ***
        If you specify a cluster journal log different from current one, you
        may need to reenter info on journal redirection, restore range, etc.

Enter the name of the cluster journal log to create (ENTER if none) =>
cachejrn.txt
How many cluster members were involved? (Q to quit) => 3
For each cluster member, enter the location(s) and name prefix (if any) of the
journal files to restore --
Cluster member #0                       Journal File Name Prefix:
  Directory: _$1$DRA1:[TEST.50.MGR.JOURNAL]
  Directory:
Cluster member #1                       Journal File Name Prefix:
  Directory: _$1$DRA1:[TEST.5A.MGR.JOURNAL]
  Directory:
Cluster member #2                       Journal File Name Prefix:
  Directory: _$1$DRA1:[TEST.5B.MGR.JOURNAL]
  Directory:
```

This example shows the creation of a cluster journal log, CACHEJRN.TXT, for a cluster with three members whose journal files were originally located on _1DRA1:.

The next menu contains the additional option to redirect the journal files in the cluster journal log you created:

```
Cluster Journal Restore - Setup - Journal File Info
[I]dentify an existing cluster journal log to use for the restore
   Current: _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
[R]edirect journal files in _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
  - OR -
[C]reate a cluster journal log by specifying where journal files are

Selection ( if no change):
```

Enter **R** to redirect the files as described in Identify the Cluster Journal Log. When finished entering redirect information, press **Enter** to return to the Main Settings menu.

6.5.1.2 Change the Starting Point of the Restore

The second and third items on the Main Settings menu specify the range of restore—where in the journal files to begin restoring and where to stop. The starting point information contains the starting journal file and sequence number for each cluster member. The default for where to begin is determined in the following order:

- If a cluster journal restore was performed after any backup restore, restore the journal from the end of last journal restore.
- If a backup restore was performed on the current system, restore the journal from the end of the last restored backup.
- If the current system is associated with the cluster journal log being used, restore the journal from the beginning of the current cluster session
- Otherwise, restore the journal from the beginning of the cluster journal log.

```
Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
```

```
      _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
2. To START restore at the end of last restored backup
   _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
          -> _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
3. To STOP restore at the end of the cluster journal log
4. To SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions

Select an item to modify ('Q' to quit or ENTER to accept and continue): 2

Cluster Journal Restore - Setup - Where to Start Restore
1. At the beginning of a cluster session
2. At a specific journal marker
3. Following the restore of backup _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK  (*)
   i.e., at the journal marker located at
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008

Selection ( if no change): 1

To start journal restore at the beginning of cluster session ...
1. 20030904 09:47:01
2. 20031002 13:19:12
3. 20031002 13:26:40
4. 20031002 13:29:10
5. 20031002 13:51:31
6. 20031002 13:58:57
7. 20031002 14:29:42
8. 20031002 14:33:55
9. 20031002 14:35:48

=> 5

Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
      _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
2. To START restore at the beginning of cluster session <20031002 13:51:31>
3. To STOP restore at the end of the cluster journal log
4. To SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions

Select an item to modify ('Q' to quit or ENTER to accept and continue): 2

Cluster Journal Restore - Setup - Where to Start Restore
1. At the beginning of a cluster session (*): <20031002 13:51:31>
2. At a specific journal marker
3. Following the restore of backup _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK

Selection ( if no change): 2

To start restore at a journal marker location (in original form)
journal file: _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
offset: 134120

You have chosen to start journal restore at
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
the journal location by the end of backup _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK
```

The submenu varies slightly based on the current settings. For example, if no backup restore was performed, the submenu for specifying the beginning of the restore does not list option 3 to restore from the end of last backup. In a submenu, the option that is currently chosen is marked with an asterisk (*).

6.5.1.3 Change the Ending Point of the Restore

By default, the restore ends at either the current journal location, if the current system is associated with the selected cluster journal log, or the end of the journal log. The submenu for option 3 is similar to that for option 2:

```
Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
      _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
```

```
2. To START restore at the end of last restored backup
   _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
         -> _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
3. To STOP restore at the end of the cluster journal log
4. To SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions

Select an item to modify ('Q' to quit or ENTER to accept and continue): 3

Cluster Journal Restore - Setup - Where to Stop Restore
1. At the end of a cluster session
2. At the end of _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
3. At a specific journal marker
```

This is the menu you would see if the journal log is the one for the current cluster:

```
Select an item to modify ('Q' to quit or ENTER to accept and continue): 3

Cluster Journal Restore - Setup - Where to Stop Restore
1. At the end of a cluster session
2. At current journal location (*)
3. At a specific journal marker
```

The submenu varies slightly based on the current settings. For example, depending whether or not the journal log is the one for current cluster, option 2 in the menu for specifying the end of the restore would be either the current journal location or the end of the journal log. In a submenu, the option that is currently chosen is marked with an asterisk (*).

6.5.1.4 Toggle the Switching Journal File Setting

The fourth menu item specifies whether to switch the journal file before the restore. If you select this item number, the value is toggled between the values `To SWITCH` and `NOT to SWITCH` the journal file; the menu is displayed again with the new setting:

```
Select an item to modify ('Q' to quit or ENTER to accept and continue): 4

Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
      _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
2. To START restore at the end of last restored backup
   _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
         -> _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
3. To STOP restore at the end of the cluster journal log
4. NOT to SWITCH journal file before journal restore
5. To DISABLE journaling the dejournaled transactions
```

The default is to switch the journal file before the restore. This provides a clean start so that updates that occur after the restore are in new journal files.

6.5.1.5 Toggle the Disable Journaling Setting

The fifth menu item specifies whether to disable journaling of the dejournaled transactions during the restore. If you select this item, the value is toggled between the values `DISABLE` and `NOT to DISABLE` journaling the dejournaled transactions; the menu is redisplayed with the new setting.

```
Select an item to modify ('Q' to quit or ENTER to accept and continue): 5

Cluster Journal Restore - Setup - Main Settings
1. To LOCATE journal files using cluster journal log
      _$1$DRA1:[TEST.50.MGR]CACHEJRN.TXT
   with redirections of journal files
      _$1$DRA1:[TEST.50.MGR.JOURNAL] -> _$1$DRA2:[TEST.50.MGR.JOURNAL]
      _$1$DRA1:[TEST.5A.MGR.JOURNAL] -> _$1$DRA2:[TEST.5A.MGR.JOURNAL]
      _$1$DRA1:[TEST.5B.MGR.JOURNAL] -> _$1$DRA2:[TEST.5B.MGR.JOURNAL]
2. To START restore at the end of last restored backup
   _$1$DRA1:[TEST.50.MGR]CLUFULL.BCK
      134120,_$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
         -> _$1$DRA1:[TEST.50.MGR.JOURNAL]20031002.008
3. To STOP restore at the end of the cluster journal log
4. NOT to SWITCH journal file before journal restore
5. NOT to DISABLE journaling the dejournaled transactions
```

For better performance, the default setting is to disable journaling the dejournaled transactions. However, if you are running a cluster shadow, you may want to choose not to disable journaling.

Note: If you choose *not* to disable journaling, the dejournaled transactions are journaled only if they otherwise meet the normal criteria for being journaled.

6.5.2 Generate a Common Journal File

The user interface for this option is similar to the first with additional questions about the contents and format of the output file. However, instead of restoring the journal files, this option produces a common-format journal file that can be read by the **^%JREAD** utility on a Caché system that does not support cluster journal restores or on another platform such as DSM.

^JCONVERT provides the same functionality if you answer `Yes` to the `Cluster Journal Convert?` question.

The second option produces a single common-format output file from the cluster journal files. It calls the **^JCONVERT** utility, which takes a journal file from a single system and writes it out in a common format to be read by the **%JREAD** routine. This is useful for restoring journal files across versions of Caché where the journal files are not compatible (for example, as part of an "almost rolling" upgrade) or as part of failing back to an earlier release. You can also use this option to write the journal file in a format that can be loaded into another platform such as DSM.

```
Cluster Journal Restore Menu
------------------------------------------------------------
1) Cluster journal restore
2) Generate common journal file from specific journal files
3) Cluster journal restore after backup restore
4) Cluster journal restore corresponding to Caché backups
------------------------------------------------------------
H) Display Help
E) Exit this utility
------------------------------------------------------------

Enter choice (1-4) or [E]xit/[H]elp? 2
```

6.5.3 Perform a Cluster Journal Restore after a Backup Restore

Option three restores the journal files after a Caché backup has been restored. This is similar to the restore performed by the incremental backup restore routine, **^DBREST**, after a cluster backup restore when there is no way to run it independently of restoring a backup. (To restart the journal restore, for example.) One difference between this option and restoring using

^DBREST is that this option does not start with the list of databases contained in the backup; you must enter the database list.

The routine offers to include all currently cluster-mounted databases in the restore, but if it is being run after restoring a backup, the databases restored by the backup are then privately mounted unless you change the mount state. (The restore mounts them privately and leaves them privately mounted when it is finished.) It starts with the markers recorded in the journal files by the backup and ends with the end of the journal data.

6.5.4 Perform a Cluster Journal Restore Based on Caché Backups

The fourth menu option restores the journal files using journal markers that were added by a Caché backup to specify the starting point and, optionally, the end point. It is similar to option three except that it uses backups which have been performed to designate where to start rather than backups which have been restored. Functionally they are the same; both options use a marker which has been placed into the journal file by a Caché backup as the starting point. The difference is in the list of choices of where to start.

6.6 Journal Dump Utility

On a Caché clustered system, the **^JRNDUMP** routine displays the cluster session ID (cluster startup time) of a journal file instead of the word `JRNSTART`. The **^JRNDUMP** routine displays a list of records in a journal file, showing the cluster session ID along with the journal file sizes.

The utility lists journal files maintained by the local system as well as journal files maintained by other systems of the Caché cluster, in the order of cluster startup time (cluster session ID) and the first and last cluster journal sequence numbers of the journal files. Journal files created by **^JRNSTART** are marked with an asterisk (*). Journal files that are no longer available (purged, for example) are marked with `D` (for deleted). Journal file names are displayed with indentions that correspond to their CSN, that is: no indention for journal files from system `0`, one space for system `1`, two spaces for system `2`, etc.

Sample output from the cluster version of **^JRNDUMP** follows. By default, The level-1 display on a clustered system is quite different from the nonclustered one:

```
 FirstSeq   LastSeq   Journal Files

 Session 20030820 11:02:43
    0          0 D /bench/test/cache/50a/mgr/journal/20030820.003
    0          0   /bench/test/cache/50b/mgr/journal/20030820.004
 Session 20030822 10:55:46
    3          3   /bench/test/cache/50b/mgr/journal/20030822.001

(N)ext,(P)rev,(G)oto,(E)xamine,(Q)uit =>
```

Besides a list of journal files from every cluster node, (even the dead ones), there are cluster session IDs and the first and the last cluster journal sequence numbers of each journal file. A cluster session ID (the date-time string following `Session`) is the time the first node of the cluster starts. A cluster session ends when the last node of the cluster shuts down. Files from different nodes are shown with different indention: no indentation for the node with CSN 0, one space for the node with CSN 1, and so on. The CSN of a node uniquely identifies the node within the cluster at a given time. The files labeled `D` have most likely been deleted from their host systems.

The previous version of **^JRNDUMP** for clusters is available as **OLD^JRNDUMP**, if you prefer that output.

6.7 Startup Recovery Routine

The following is the help display of the startup recovery routine, **^STURECOV**:

```
%SYS>Do ^STURECOV

Enter error type (? for list) [^] => ?

Supported error types are:
     JRN - Journal restore and transaction rollback
     CLUJRN - Cluster journal restore and transaction rollback

Enter error type (? for list) [^] => CLUJRN


Cluster journal recovery options
----------------------------------------------------------------
1) Display the list of errors from startup
2) Run the journal restore again
4) Dismount a database
5) Mount a database
6) Database Repair Utility
7) Check Database Integrity
----------------------------------------------------------------
H) Display Help
E) Exit this utility
----------------------------------------------------------------

Enter choice (1-8) or [E]xit/[H]elp? H
----------------------------------------------------------------
Before running ^STURECOV you should have corrected the
errors that prevented the journal restore or transaction rollback
from completing. Here you have several options regarding what
to do next.

Option 1: The journal restore and transaction rollback procedure
     tries to save the list of errors in ^%SYS(). This is not always
     possible depending on what is wrong with the system. If this
     information is available, this option displays the errors.
Option 2: This option performs the same journal restore and
     transaction rollback which was performed when the system was
     started. The amount of data is small so it should not be
     necessary to try and restart from where the error occurred.
Option 3 is not enabled for cluster recovery
Option 4: This lets you dismount a database. Generally this
     would be used if you want to let users back on a system but
     you want to prevent them from accessing a database which still
     has problems (^DISMOUNT utility).
Option 5: This lets you mount a database (^MOUNT utility).
Option 6: This lets you edit the database structure (^REPAIR utility).
Option 7: This lets you validate the database structure (^INTEGRIT utility).
Option 8 is not enabled for cluster recovery. Shut the system
  down using the bypass option with ccontrol stop and then start it
  with ccontrol start. During startup answer YES when asked if you
  want to continue after it displays the message related to errors
  during recovery.

Press <enter> continue

Cluster journal recovery options
----------------------------------------------------------------
1) Display the list of errors from startup
2) Run the journal restore again
4) Dismount a database
5) Mount a database
6) Database Repair Utility
7) Check Database Integrity
----------------------------------------------------------------
H) Display Help
E) Exit this utility
----------------------------------------------------------------

Enter choice (1-8) or [E]xit/[H]elp?
```

6.8 Setting Journal Markers on a Clustered System

To set a journal marker effective cluster-wide, use the following routine

```
$$CLUSET^JRNMARK(id,text,swset)
```

Where....	is...
id	Marker ID (for example, -1 for backup)
text	Marker text (for example, “timestamp” for backup)
swset	1 — if the switch that inhibits database reads and writes (switch 10) has been set cluster-wide (and locally) by the caller. The caller is responsible for clearing it afterwards.
	0 — if the switch has not been set. The routine takes care of setting and clearing the switch properly

Note that switch 10 must be set locally and cluster-wide to ensure the integrity of the journal marker. If successful, the routine returns the location of the marker — the offset of the marker in the journal file and the journal file name — delimited by a comma. Otherwise, it returns an error code (<=0) and error message, also delimited by a comma.

6.9 Cluster Journal Information Global

The global node *^%SYS("JRNINFO")* is where cluster journal information is maintained. It is indexed by current cluster session ID and is recreated every time the cluster restarts. This allows you to modify or delete the cluster journal log (presumably after deleting the journal files) between two cluster sessions, as the update algorithm assumes that you do not alter the cluster journal log during a cluster session.

The *^%SYS("JRNINFO")* global has three subcomponents:

- The *jrninfo* table is indexed by journal file names, with the value of the top node being the number of entries in the cluster journal log and the value of each subnode being a comma-delimited list of the attributes of that journal file: CSN, line number of the journal file in the cluster journal log, CSI, first and last sequence numbers.
- The *jrninfor* (*r* for reverse) table is a list of journal files, with CSN as the primary key and the line number of the journal file in the cluster journal log as the secondary key.
- The *seqinfo* table contains the following subscripts: CSI, first and last sequence numbers, CSN, and line number of the journal file in the cluster journal log.

Here is a sample of *^%SYS("JRNINFO")* contents:

```
^%SYS("JRNINFO",1032803946,"jrninfo")=16
^%SYS("JRNINFO",1032803946,"jrninfo",
  "_$1$DKA0:[TEST.50.MGR.JOURNAL]20030916.002")=1,2,1031949277,160,160
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030913.004")=0,1,1031949277,3,3
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.001")=0,3,1031949277,292,292
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.002")=0,4,1032188507,3,417
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.003")=0,7,1032188507,3,422
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.004")=0,8,1032197355,3,4
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.005")=0,9,1032197355,3,7
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.006")=0,10,1032197355,3,10
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.007")=0,11,1032197355,3,17
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.008")=0,12,1032197355,3,17
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030918.001")=0,13,1032197355,3,27
  "_$1$DRA1:[TEST.50.MGR.JOURNAL]20030923.001")=0,15,1032803946,3,133
  "_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.002")=1,5,1032188507,3,3
```

```
  "_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.003")=1,6,1032188507,131,131
  "_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030923.001")=1,14,1032197355,39,39
  "_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030923.002")=1,16,1032803946,3,3

^%SYS("JRNINFO",1032803946,"jrninfor",0,
            1)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030913.004
            3)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.001
            4)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.002
            7)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.003
            8)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.004
            9)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.005
           10)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.006
           11)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.007
           12)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030916.008
           13)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030918.001
           15)=_$1$DRA1:[TEST.50.MGR.JOURNAL]20030923.001

^%SYS("JRNINFO",1032803946,"jrninfor",1,
                  2)=_$1$DKA0:[TEST.50.MGR.JOURNAL]20030916.002
                  5)=_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.002
                  6)=_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030916.003
                 14)=_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030923.001
                 16)=_$1$DRA1:[TEST.5A.MGR.JOURNAL]20030923.002

^%SYS("JRNINFO",1032803946,"seqinfo",1031949277,3,3,0,1)=
^%SYS("JRNINFO",1032803946,"seqinfo",1031949277,160,160,1,2)=
^%SYS("JRNINFO",1032803946,"seqinfo",1031949277,292,292,0,3)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032188507,3,3,1,5)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032188507,3,417,0,4)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032188507,3,422,0,7)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032188507,131,131,1,6)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,3,4,0,8)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,3,7,0,9)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,3,10,0,10)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,3,17,0,11)=
                                                      12)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,3,27,0,13)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032197355,39,39,1,14)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032803946,3,3,1,16)=
^%SYS("JRNINFO",1032803946,"seqinfo",1032803946,3,133,0,15)=
```

6.10 Shadow Information Global and Utilities

The global node *^SYS("shdwcli")* is where shadow client information is maintained. Most of the values are available through the utilities **ShowState^SHDWX**, **ShowError^SHDWX**, and **ShowWhere^SHDWX**.

Running **ShowState^SHDWX** displays most of the data contained in the global:

```
%SYS>d ShowState^SHDWX("clutest",1)

Shadow ID        PrimaryServerIP  Port  R S Err
------------------------------------------------------------------------------
clutest                   rodan 42009  0 1   1
\_ clutest~0         192.9.202.5 42009  0 1
\_ clutest~1         192.9.202.5 42009  0 1
\_ clutest~2              rodan 42009  0 1

Redirection of Global Sets and Kills:
  ^^_$1$DKB300:[TEST.CLU.5X] -> ^^_$1$DKA0:[TEST.CLU.5X]
Redirection of Master Journal Files:
  Base directory for auto-redirection: _$1$DKA0:[TEST.5X.SHADOW]
  _$1$DRA2:[TEST.5X.JOURNAL] -> _$1$DKA0:[TEST.5X.SHADOW.1]
  _$1$DRA2:[TEST.5Y.JOURNAL] -> _$1$DKA0:[TEST.5X.SHADOW.2]
  _$1$DRA2:[TEST.5Z.JOURNAL] -> _$1$DKA0:[TEST.5X.SHADOW.3]
Primary Server Cluster ID: _$1$DKB400:[TEST.5X]CACHE.PIJ
Primary Server Candidates (for failover):
         192.9.202.5   42009
               rodan   42009
         192.9.202.5   42019
         192.9.202.5   42029
When to purge a shadow journal file: after it's dejournaled
```

The output displayed from the *^SYS("shdwcli")* global has the following components:

- *Shadow ID* — the ID of a copier shadow is partially inherited from the parent shadow. The *clu* subnode of a copier contains the ID of the parent, and the *sys* subnode of the parent contains a list of the IDs of the copiers.
- *PrimaryServerIP* and *Port* — for a copier, these specify the system from which it gets journal files; for the dejournaler, the system from which it gets journal information (JRNINFO server). The values are stored in the *ip* and *port0* subnodes.
- *R* — has the value `1` if the shadow is running; from the *stat* subnode.
- *S* — has the value `1` if the shadow is requested to stop (due to latency, it is possible that both *R* and *S* have the value `1` if the shadow has yet to check for the stop request); from the *stop* subnode.
- *Err* — number of errors encountered. See details through **ShowError^SHDWX**, which displays the information from the *err* subnode.
- *Redirection of Global Sets and Kills* — referred to as database mapping in the Management Portal; from the *dbmap* subnode of the cluster shadow.
- *Redirection of Master Journal Files* — discussed in the Using Caché Routines section; stored in the *jrndir* subnode of the cluster shadow. The value of the *jrndir* subnode is the number of journal directories that have been automatically redirected (in the preceding example output, the next new journal directory is redirected to a subdirectory [.4]). (*jrndir*,0) is the base shadow directory and everything else indicates a redirection of journal directories with the server journal directory being the key and the shadow journal directory being the value.
- *Primary Server Cluster ID* — used to prevent the shadow from following a node to a different cluster; from the *DBServerClusterID* subnode.
- *Primary Server Candidates (for failover)* — the list of current live members of the cluster. If one member dies, a shadow (either the dejournaler or a copier) that gets information from the member tries other members on the list until it succeeds. A new member is added to the list as soon as the shadow knows its presence; from the *servers* subnode.
- *When to purge a shadow journal file* — works in the same way as purging of local journal files. The age threshold is set by the *lifespan* subnode of the cluster shadow. Unlike purging of local journal files, however, if the value of *lifespan* is 0, the shadow journal files are purged as soon as they have been dejournaled completely. The purged journal files are listed in the *jrndel* subnode of the copiers.

The *chkpnt* subnode stores a list of checkpoints. A checkpoint is a snapshot of the work queue of the dejournaler—the current progress of dejournaling. The value of the *chkpnt* subnode indicates which checkpoint to use when the dejournaler resumes. This is the checkpoint displayed by **ShowWhere^SHDWX**. Updating the value of the *chkpnt* subnode after having completely updated the corresponding checkpoint, avoids having a partial checkpoint in the case of system failover in the middle of an update (in that case, dejournaler would use previous checkpoint).

The copiers keep the names of the copied (or being copied) journal files in the *jrnfil* subnode. This makes it possible to change the redirection of journal files by allowing the dejournaler to find the shadow journal files in the old directory while the copiers copy new journal files to the new location. Once a shadow journal file is purged, it is moved from the *jrnfil* list to the *jrndel* list.

Here is a sample of the *^SYS("shdwcli")* contents for the nodes for the cluster shadow, *clutest*, and two of its copier shadows:

```
^SYS("shdwcli","clutest")=0
^SYS("shdwcli","clutest","DBServerClusterID")=_$1$DKB400:[TEST.5X]CACHE.PIJ
                                                "at")=0
                                           "chkpnt")=212
^SYS("shdwcli","clutest","chkpnt",1)=1,1012488866,-128
^SYS("shdwcli","clutest","chkpnt",1,1012488866,0,1)=0,,,,
^SYS("shdwcli","clutest","chkpnt",2)=2,1012488866,-128
^SYS("shdwcli","clutest","chkpnt",2,1012488866,-128,2)=
                    -128,_$1$DKA0:[TEST.5X.SHADOW.1]20020131.001,0,,0
^SYS("shdwcli","clutest","chkpnt",3)=6,1012488866,5
^SYS("shdwcli","clutest","chkpnt",3,1012488866,11,6)=
                   5,_$1$DKA0:[TEST.5X.SHADOW.1]20020131.001,-132252,,0
^SYS("shdwcli","clutest","chkpnt",4)=35,1012488866,85
^SYS("shdwcli","clutest","chkpnt",4,1012488866,95,35)=
                  85,_$1$DKA0:[TEST.5X.SHADOW.1]20020131.001,-136984,,0
^SYS("shdwcli","clutest","chkpnt",5)=594,1012488866,807
^SYS("shdwcli","clutest","chkpnt",5,1012488866,808,594)=
```

```
              808,_$1$DKA0:[TEST.5X.SHADOW.1]20020131.001,262480,1,0
...
^SYS("shdwcli","clutest","chkpnt",212)=24559,1021493730,5
^SYS("shdwcli","clutest","chkpnt",212,1021493730,37,24559)=
            5,_$1$DKA0:[TEST.5X.SHADOW.1]20020515.001,-132260,,0
^SYS("shdwcli","clutest","cmd")=
^SYS("shdwcli","clutest","dbmap","^^_$1$DKB300:[TEST.CLU.5X]")=
                                          ^^_$1$DKA0:[TEST.CLU.5X]
^SYS("shdwcli","clutest","end")=0
                          "err")=1
^SYS("shdwcli","clutest","err",1)=
20020519 14:16:34 568328925 Query+8^SHDWX;-12;
reading ans from |TCP|42009timed out,Remote server is not responding
^SYS("shdwcli","clutest","err",1,"begin")=20020519 14:09:09
                                 "count")=5
^SYS("shdwcli","clutest","errmax")=10
                           "intv")=10
                             "ip")=rodan
                         "jrndir")=3
^SYS("shdwcli","clutest","jrndir",0)=_$1$DKA0:[TEST.5X.SHADOW]
       "_$1$DRA2:[TEST.5X.JOURNAL]")=_$1$DKA0:[TEST.5X.SHADOW.1]
       "_$1$DRA2:[TEST.5Y.JOURNAL]")=_$1$DKA0:[TEST.5X.SHADOW.2]
       "_$1$DRA2:[TEST.5Z.JOURNAL]")=_$1$DKA0:[TEST.5X.SHADOW.3]
^SYS("shdwcli","clutest","jrntran")=0
                         "lifespan")=0
                           "locdir")=
                           "locshd")=
                              "pid")=568328919
                            "port")=
                           "port0")=42009
                           "remjrn")=
^SYS("shdwcli","clutest","servers","42009,192.9.202.5")=
                                          "42009,rodan")=
                                    "42019,192.9.202.5")=
                                    "42029,192.9.202.5")=
^SYS("shdwcli","clutest","stat")=0
                          "stop")=1
^SYS("shdwcli","clutest","sys",0)=
                                1)=
                                2)=
^SYS("shdwcli","clutest","tcp")=|TCP|42009
                       "tpskip")=1
                          "type")=21

^SYS("shdwcli","clutest~0")=0
^SYS("shdwcli","clutest~0","at")=0
                           "clu")=clutest
                           "cmd")=
                           "end")=132260
                           "err")=0
                          "intv")=10
                            "ip")=192.9.202.5
^SYS("shdwcli","clutest~0","jrndel",
                     "_$1$DKA0:[TEST.5X.SHADOW.1]20020131.001")=
                     "_$1$DKA0:[TEST.5X.SHADOW.1]20020131.002")=
                     "_$1$DKA0:[TEST.5X.SHADOW.2]20020510.010")=
^SYS("shdwcli","clutest~0","jrnfil")=36
^SYS("shdwcli","clutest~0","jrnfil",35)=
                          _$1$DRA2:[TEST.5X.JOURNAL]20020513.006
^SYS("shdwcli","clutest~0","jrnfil",35,"shdw")=
                          _$1$DKA0:[TEST.5X.SHADOW.1]20020513.006
^SYS("shdwcli","clutest~0","jrnfil",36)=
                          _$1$DRA2:[TEST.5X.JOURNAL]20020515.001
^SYS("shdwcli","clutest~0","jrnfil",36,"shdw")=
                          _$1$DKA0:[TEST.5X.SHADOW.1]20020515.001
^SYS("shdwcli","clutest~0","jrntran")=0
                            "locdir")=
                            "locshd")=
                 _$1$DKA0:[TEST.5X.SHADOW.1]20020515.001
                            "pause")=0
                              "pid")=568328925
                             "port")=42009
                            "port0")=42009
                           "remend")=132260
                           "remjrn")=
                             "stat")=0
                             "stop")=1
                              "tcp")=|TCP|42009
                           "tpskip")=1
                             "type")=12

^SYS("shdwcli","clutest~1")=0
^SYS("shdwcli","clutest~1","at")=0
                           "clu")=clutest
                           "cmd")=
```

```
                          "end")=132248
                          "err")=0
                         "intv")=10
                           "ip")=192.9.202.5
^SYS("shdwcli","clutest~1","jrndel",
            "_$1$DKA0:[TEST.5X.SHADOW.1]20020510.003")=
            "_$1$DKA0:[TEST.5X.SHADOW.2]20020131.001")=
            "_$1$DKA0:[TEST.5X.SHADOW.2]20020510.008")=
^SYS("shdwcli","clutest~1","jrnfil")=18
^SYS("shdwcli","clutest~1","jrnfil",17)=
             _$1$DRA2:[TEST.5X.JOURNAL]20020510.011
^SYS("shdwcli","clutest~1","jrnfil",17,"shdw")=
             _$1$DKA0:[TEST.5X.SHADOW.1]20020510.011
^SYS("shdwcli","clutest~1","jrnfil",18)=
             _$1$DRA2:[TEST.5Y.JOURNAL]20020510.011
^SYS("shdwcli","clutest~1","jrnfil",18,"shdw")=
             _$1$DKA0:[TEST.5X.SHADOW.2]20020510.011
^SYS("shdwcli","clutest~1","jrntran")=0
                           "locdir")=
                           "locshd")=
            _$1$DKA0:[TEST.5X.SHADOW.2]20020510.011
                              "pid")=568328925
                            "port")=42009
                           "port0")=42009
                          "remend")=132248
                          "remjrn")=
                            "stat")=0
                            "stop")=1
                             "tcp")=|TCP|42009
                          "tpskip")=1
                            "type")=12
```

7

Data Consistency on Multiple Systems

When mirroring, shadowing, or other mechanisms are used to maintain a copy of data on another system, you may want to check the consistency of that data between the two systems. DataCheck provides this checking and includes provisions to recheck transient discrepancies.

This chapter discusses the following topics:

- DataCheck Overview
- DataCheck for Mirror Configurations
- DataCheck Setup Procedure
- ^DATACHECK Routine
- Special Considerations for Data Checking

7.1 DataCheck Overview

DataCheck provides a mechanism to compare the state of data on two systems — the DataCheck source and the DataCheck destination — to determine whether or not they match. All configuration, operational controls and results of the check are provided on the destination system; the source system is essentially passive.

On the instance of Caché that is to act as the DataCheck destination, you must create a DataCheck destination configuration. You can create multiple destination configurations on the same instance, which you can configure to check data against multiple source systems (or configure them to check different data against a single source). If DataCheck is being used to check the consistency of a shadow system, it is recommended that the Caché instance serving as the destination of shadowing also be configured as the DataCheck destination. If you are using DataCheck to check the consistency of a mirror, see DataCheck for Mirror Configurations for more details.

The following subsections describe DataCheck topics in more detail:

- DataCheck Queries
- DataCheck Jobs
- DataCheck Results
- DataCheck Workflow

7.1.1 DataCheck Queries

The destination system submits work units called DataCheck "queries" to the source system. Each query specifies a database, an initial global reference, a number of nodes, and a target global reference. Both systems calculate an answer by traversing the specified number of global nodes starting with the initial global reference, and hashing the global keys and values. If the answers match, the destination system records the results and resubmits the query with a larger number of nodes and the initial global reference advanced; if they don't match, the query is resubmitted with a smaller number of nodes until the discrepancy is isolated down to the configured minimum query size.

You can display information about the queries submitted by the destination system using the **View Queries** option of the View Details submenu of the **^DATACHECK** routine, including the globals that remain to be processed (or global ranges if subscript include/exclude ranges are used), and the active queries currently being worked on by DataCheck.

7.1.2 DataCheck Jobs

The answer to each query is calculated by DataCheck worker jobs running on both the source system and the destination system. The number of worker jobs is determined by the dynamically tunable performance settings of the destination system; for more information, see "Performance Considerations" in this chapter.

In addition to the worker jobs, there are other jobs on each system. The following additional jobs run on the destination system:

- Manager job — Loads and dispatches queries, compares query answers, and manages the progression through the workflow phases; this job is connected to the source system Manager job.
- Receiver job — Receives answers from the source system.

The following additional jobs run on the source system:

- Manager job — Receives requests from the destination system Manager job and sends them to worker jobs.
- Sender job — Receives query answers from the worker jobs and sends them to the destination system Receiver job; this job is connected to the destination system Receiver job.

7.1.3 DataCheck Results

The results of the check lists global subscript ranges with one of the following states:

- Unknown — DataCheck has not yet checked this range.
- Matched — DataCheck has found that this range matches.
- Unmatched — DataCheck has found a discrepancy in this range.
- Collation Discrepancy — Global was found to have differing collation between the source system and the destination system.
- Excluded — This range is excluded from checking.

You can view the results from the current check and the final results from the last check on the destination system; for more information, see the SYS.DataCheck.RangeList class. For all subscript ranges within DataCheck, the beginning of a range is inclusive and the end exclusive. See Specifying Globals and Subscript Ranges to Check in this chapter for information about subscript ranges.

The following provides a sample check result:

```
c:\InterSystems\cache\mgr\mirror2 ^XYZ       Unmatched
        ^XYZ --Matched--> ^XYZ(3001,4)
        ^XYZ(3001,4) --Unmatched--> ^XYZ(5000)
        ^XYZ(5000) --Matched--> [end]
```

This result indicates that the nodes in the range starting at ^*XYZ* up to but not including ^*XYZ(3001,4)* are matched, while there is at least one discrepancy in the range of nodes from ^*XYZ(3001,4)* up to but not including ^*XYZ(5000)*. The nodes in the range from ^*XYZ(5000)* to the end are matched.

The minimum number and frequency of discrepancies in the unmatched range depends on the minimum query size (see Performance Considerations). For example, if the minimum query size is set to the default of 32 in this case, there is at least one discrepancy every 32 nodes from ^*XYZ(3001,4)* until ^*XYZ(5000)*; if there were a sequence within this range of more than 32 nodes without a discrepancy, it would appear in the results as a separate matched range.

7.1.4 DataCheck Workflow

During the check, data may be changing and transient discrepancies may be recorded. Rechecking may be required to eliminate these transient discrepancies. The destination system has a workflow that defines a strategy for how to check the globals.

A typical workflow begins with the "Check" phase as phase #1. (Phase #1 should always be defined as the logical starting point of the check cycle, since it is used by the workflow timeout and the Start dialog of the **^DATACHECK** routine to indicate a "reset" from beginning, as described in the next section.) At the beginning of this phase, the current set of results are saved as the last completed results and a new set of active results is established. DataCheck makes an initial pass through all globals specified for inclusion in the check.

Following the Check phase, the "Recheck Discrepancies" phase is typically specified with the desired number of iterations. Each iteration rechecks all unmatched ranges in an effort to eliminate transient discrepancies.

As each phase of the workflow is completed, DataCheck moves to the next phase. The workflow is implicitly restarted from phase #1 after the last phase is complete. The "Stop" phase shuts down all DataCheck jobs and the "Idle" phase causes DataCheck to wait for you to manually specify the next phase.

7.1.4.1 Starting/Stopping/Reconnecting DataCheck

You can stop and start DataCheck at any time; when you start DataCheck, it resumes the workflow from where it left off. In addition, you can specify a different workflow phase to follow the current phase and/or abort the current phase at any time.

If, during a check, DataCheck is stopped, becomes disconnected, or pauses due to mirroring, the routine reports why the system was stopped, what phase it stopped in, and what it will do when it starts (for example, resume processing, move to the next phase, change phase due to user request or restart at phase #1 due to workflow timeout). If, upon starting, DataCheck

is going to resume processing the current phase or make a transition to any phase other than phase #1, you are offered the option of restarting at phase #1, as in the following example:

```
Option? 4

Configuration Name: test

State:  Stopped due to Stop Requested
Current Phase: 1 - Check
Workflow Phases:
  1 - Check
  2 - RecheckDiscrepancies, Iterations=10
  3 - Stop
  (restart)
Workflow Timeout: 432000
New Phase Requested: 2
Abort Current Phase Requested

DataCheck is set to abort the current phase and transition to phase #2.

You may enter RESTART to restart at phase #1

Start Datacheck configuration 'test'? (yes/no/restart)
```

In cases in which DataCheck becomes disconnected and reconnects only after an extended period, it may be more desirable to restart from phase #1 of the workflow instead. For example, if the systems were disconnected for several weeks in the middle of a check and then the check is resumed, the results are of questionable value, having been collected in part from two weeks prior and in part from the present time. The workflow has a Timeout property that specifies the time, in seconds, within which DataCheck may resume a partially completed workflow phase. If the timeout is exceeded, DataCheck restarts from phase #1 the next time it reaches the running state. The default value is five days (432000 seconds), based on the assumption that a large amount of data is checked by this DataCheck configuration and the check may take hours or days to complete normally; a smaller value may be preferable for configurations that complete a check in a shorter amount of time. A value of zero means no timeout.

Note: As noted, you should define phase #1 to be the logical starting point of the check cycle, since it is used by the workflow timeout and the Start dialog of the **^DATACHECK** routine to indicate a "reset" from beginning, as shown in the previous example.

7.2 DataCheck for Mirror Configurations

Upon creating a DataCheck destination configuration, if the system is a member of a mirror (see the "Mirroring" chapter of the *Caché High Availability Guide*), you are given the option to configure DataCheck to check the mirrored data. If you choose this option, you need only select the mirror member to act as the DataCheck source, and the rest of the configuration is automatic.

When a check begins, all mirrored databases are included in the check; you do not have to map databases individually. You can specify which globals are checked or exclude entire databases, as described in Specifying Globals and Subscript Ranges to Check. A mirror-based DataCheck configuration cannot be used to check non-mirrored databases, but a separate non-mirrored DataCheck configuration can be created for such purposes.

This section discusses the following topics:

- Planning DataCheck within the Mirror
- Selecting Globals to Check

7.2.1 Planning DataCheck within the Mirror

Each DataCheck destination configuration connects to one source mirror member. Although the source member should not be changed, additional DataCheck configurations can be created to check against more than one source mirror member (or to check different sets of data from the same source).

This section includes the following member-specific subsections:

- Checking Data Between Failover Members
- Checking Data on Async Members

7.2.1.1 Checking Data Between Failover Members

When checking between failover mirror members, the check is typically run with the backup failover member configured as the DataCheck destination for the following reasons:

- The DataCheck destination uses more resources than the source in order to maintain the results of the check and other state information (which is itself journaled).
- If the backup failover member is the DataCheck destination, the results are available for review on the backup if the primary failover member goes down.

  Note: In most configurations, it is assumed that the failover has already occurred and any review of the results probably happens after the failover decision point.

Whenever DataCheck loses its connection to the source, it retries the connection, waiting indefinitely for the source machine to become available again. If a mirror-based DataCheck is started on the destination when it was not the primary failover member, and that member becomes the primary, DataCheck stops rather than automatically try to reconnect. This prevents DataCheck from unintentionally running on the primary. For more information about reconnecting, see Starting/Stopping/Reconnecting DataCheck in this chapter.

7.2.1.2 Checking Data on Async Members

When mirror-based DataCheck is checking between a failover member and an async member, the async member is typically the destination. This is for the same reasons mentioned above (see Checking Data Between Failover Members) in regards to checking between failover members, but primarily because the results of the check should be stored on the async member during disaster recovery.

When there are two failover members, it is often desirable to create one DataCheck destination configuration on an async member for each of the two failover members as sources. The **^DATACHECK** routine offers to create both for you, and offers settings for how they behave with respect to which of the two is the primary failover member.

Each DataCheck configuration has a setting to govern how it behaves based on the source failover member's status as the primary member. The settings are:

- **No restriction**

  Checking both without restriction (the default) is desirable because it uses the async member as an agent to check both failover members without needing to run DataCheck between the failover members.

- **Check primary only** (pause until DataCheck source is primary)

  Checking against the primary only is desirable because the primary is the true source of the data for this async member.

- **Do not check primary** (pause when DataCheck source is primary)

  Checking against the backup is desirable because it does not consume resources on the production primary system.

Note: For information about reconnecting after a pause, see Starting/Stopping/Reconnecting DataCheck in this chapter.

For DataCheck configurations that are run manually (on demand) by a system administrator, these settings may not be of particular importance; they are more important for DataCheck configurations that are run continuously (or nearly so).

Any member may check another member without any particular relation. For example, if an async member is being used to check both failover members, it could also be used as the source of a check for other async members, thus avoiding the need to have any other async members check against the failover members.

7.2.2 Selecting Globals to Check

All mirrored databases that exist when DataCheck is run are checked automatically; for information about controlling which globals and databases are checked, see Specifying Globals and Subscript Ranges to Check in this chapter.

7.3 DataCheck Setup Procedure

You can set up DataCheck destination systems with the **^DATACHECK** routine and enable DataCheck source systems through the Management Portal. To set up a new DataCheck system, do the following:

1. Create new destination system.
2. Set up/edit destination system configurations, as follows:
   a. For non-mirror-based configurations, specify the hostname/IP address, superserver port, and optional SSL configuration for the TCP connection to the source system.

      For mirror-based configurations, specify the mirror member you want to check.

   b. For non-mirror-based configurations, specify the set of databases to be checked and their corresponding paths on the source system.

      For mirror-based configurations, all mirrored databases are included.

   c. Optionally, specify global selection masks and subscript ranges for fine-grained control over which databases, globals, and global ranges to include or exclude. For more information, see Specifying Globals and Subscript Ranges to Check in this chapter.

   d. Optionally, adjust the dynamically tunable settings to control the performance and system resource consumption for the check. For more information, see Performance Considerations in this chapter.

   e. Optionally, modify the workflow specifying the strategy for the check. For more informations, see "DataCheck Workflow" in this chapter.
3. Enable the **%Service_DataCheck** service on the source system. For more information, see "Enabling the DataCheck Service" in this chapter.
4. Start the destination system, which controls the checking.
5. Monitor the status of the check, as follows:
   - On the source system, view the status and log file.
   - On the destination system, view the status and log file, as well lists of queries and results.

7.3.1 Enabling the DataCheck Service

Use the Management Portal from the Caché instance running on the source system to enable the data checking service and, optionally, restrict connections:

1. Navigate to the **Services** page (**System Administration** > **Security** > **Services**) of the Management Portal.
2. Click **%Service_DataCheck** in the list of service names to edit the data checking service properties.
3. Select the **Service enabled** check box. Before clicking **Save**, you may want to first restrict which IP addresses can connect to this database source. If so, perform the next step, and then click **Save**.

   Note: When configured to check a mirror, DataCheck uses SSL if the mirror is set to use SSL (for more information, see DataCheck for Mirror Configurations in this chapter). The DataCheck service, however, does not automatically restrict access only to mirror members. If you wish to restrict DataCheck connections from other systems, you must configure the **Allowed Incoming Connections** for the **%Service_DataCheck** service.

4. Optionally, to restrict access to the service, in the **Allowed Incoming Connections** box (which displays previously entered server addresses), click **Add** to add an **IP Address**. Repeat this step until you have entered all permissible addresses.

   You may delete any of these addresses individually by clicking **Delete** in the appropriate row, or click **Delete All** to remove all addresses, therefore allowing connections from any address.

7.3.2 Specifying Globals and Subscript Ranges to Check

DataCheck lets you specify global names and subscript ranges to include in or exclude from checking using the options detailed in the following.

Note: Only literal values are accepted as global names and subscripts when specifying global and subscript ranges.

- **Check All Globals in All [Mapped / Mirrored] Databases** — Checks all globals in all mapped databases in non-mirror-based configurations; in mirror-based configurations, checks all globals in all mirrored databases.
- **Include/Exclude Some Globals/Databases** — Checks globals in the selected database based on the specified mask(s). Subscripts are not allowed.

  You can add/edit a mask or comma-separated list of masks, as follows:

  – `*` — Checks all globals (default).
  – `*` as the last character — Checks all globals starting with the preceding character(s).
  – `'` before a mask — Excludes globals from being checked.

  For example:

  – `ABC*` — all global names starting with ABC
  – `A:D` — all global names from A to D
  – `A:D,Y*` — all global names from A to D, and starting with Y
  – `*,'C*,'D*` — all globals except those starting with C or D
  – `'*` — exclude all globals

  In addition to defining a global selection mask for specific databases, you can explicitly set a "default global selection mask" that is used for databases for which no global selection mask has been defined. Initially, the default mask to set to `*`.

Note: For mirror-based DataCheck, newly added mirrored databases are included in the next check. Therefore, if you do not want newly added mirrored databases to be checked automatically, set the default mask to `'*`.

For example, to specify a default mask for all databases for which no mask is defined (`*,'^DontCheckMe`) as well as specify a global selection mask (`A:D`) specifically for the USER and USER2 databases, do the following from the **Edit Configuration** submenu of the **^DATACHECK** routine (see ^DATACHECK Routine, later in this chapter):

```
1) Import Settings from a Shadow
2) Connection Settings
3) Database Mappings
4) Globals to Check
5) Performance Settings
6) Manage Workflow

Option? 4

1) Check All Globals in All Mapped Databases
2) Include/Exclude Some Globals/Databases
3) Include/Exclude Some Globals/Databases and Subscript Ranges

Option? 1 => 2
Save changes? Yes =>

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks

Option? 2
Enter a mask string, * to include all, '* to exclude all, ? for help
Mask: * => *,'^DontCheckMe
Save changes? Yes =>

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks

Option? 3

1) C:\InterSystems\cache\mgr\dockbook\ [no mask defined, use default]
2) C:\InterSystems\cache\mgr\user\ [no mask defined, use default]
3) C:\InterSystems\cache\mgr\user2\ [no mask defined, use default]

Database (multiple selections allowed): 2,3
Enter a mask string, * to include all, '* to exclude all, ? for help
                     ! to delete this mask and revert to default
Mask: A:D

1) C:\InterSystems\cache\mgr\dockbook\ [no mask defined, use default]
2) C:\InterSystems\cache\mgr\user\ [A:D]
3) C:\InterSystems\cache\mgr\user2\ [A:D]

Database (multiple selections allowed):
Save changes? Yes =>

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks

Option?
```

- **Include/Exclude Some Globals/Databases and Subscript Ranges** — In addition to letting you perform the same tasks as the **Include/Exclude Some Globals/Databases** option, this option lets you identify subscript ranges in specific globals; global subscript ranges marked for inclusion are included whether or not the global is included in the global selection mask. For all subscript ranges within DataCheck, the beginning of a range is inclusive and the end exclusive.

  Note: DataCheck may mark data in an excluded range as matched if this is determined in the course of its operation. Discrepancies in excluded ranges, however, are never marked.

For example, continuing with the preceding example, in which you specified a global selection mask (A:D) for the USER2 database; you can include a subscript range in the ^*NAME* global by responding to the prompts, as follows:

```
1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks

Option? 1

1) Check All Globals in All Mapped Databases
2) Include/Exclude Some Globals/Databases
3) Include/Exclude Some Globals/Databases and Subscript Ranges

Option? 2 => 3
Save changes? Yes =>

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks
5) Add/Edit Subscript Ranges for a Global
6) Delete All Subscript Ranges for a Global
7) Delete All Subscript Ranges
8) View Defined Subscript Ranges

Option? 5

1) C:\InterSystems\cache\mgr\dockbook\
2) C:\InterSystems\cache\mgr\user\
3) C:\InterSystems\cache\mgr\user2\

Database: 3 C:\InterSystems\cache\mgr\user2\
Global Name: ^NAME
There are no subscript ranges defined for this global.
You may start by including all or excluding all subscripts.
Answer YES to include, NO to exclude: no

C:\InterSystems\cache\mgr\user2\        ^NAME
        ^NAME --Excluded--> [end]

From (inclusive):  ?

  Enter a global reference with or without subscripts or null for end.
  The leading ^ may be omitted.  For subscripted references the entire
  global name may be omitted and simply begin with open parentheses

From (inclusive):  (10)
To (exclusive):  (20)
Answer YES to include, NO to exclude: yes

C:\InterSystems\cache\mgr\user2\        ^NAME
        ^NAME --Excluded--> ^NAME(10)
        ^NAME(10) --Included--> ^NAME(20)
        ^NAME(20) --Excluded--> [end]

From (inclusive):
Continue editing subscript ranges for this global? Yes => no

C:\InterSystems\cache\mgr\user2\        ^NAME
        ^NAME --Excluded--> ^NAME(10)
        ^NAME(10) --Included--> ^NAME(20)
        ^NAME(20) --Excluded--> [end]

Save changes? Yes =>

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks
5) Add/Edit Subscript Ranges for a Global
6) Delete All Subscript Ranges for a Global
7) Delete All Subscript Ranges
8) View Defined Subscript Ranges

Option?
```

You can view the mask information as follows:

```
Option? 4
The default include/exclude mask is:
    *,'^DontCheckMe
```

```
The following databases are using non-default global selection criteria

  C:\InterSystems\cache\mgr\user\
    A:D
  C:\InterSystems\cache\mgr\user2\
    * Has additional global subscript ranges to include/exclude that apply
      regardless of whether those globals are included in this mask.
    A:D

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks
5) Add/Edit Subscript Ranges for a Global
6) Delete All Subscript Ranges for a Global
7) Delete All Subscript Ranges
8) View Defined Subscript Ranges

Option?
```

Since the mask information includes a note about subscript ranges, you can display that information, as follows:

```
Option? 8
Device:
Right margin: 80 =>

DataCheck Destination System: GLOBTEST
Global Selection Subscript Ranges

C:\InterSystems\cache\mgr\user2\        ^NAME
        ^NAME --Excluded--> ^NAME(10)
        ^NAME(10) --Included--> ^NAME(20)
        ^NAME(20) --Excluded--> [end]

1) Options for selecting globals to check
2) Set default include/exclude mask for databases with no mask defined
3) Add or remove include/exclude mask for databases
4) View include/exclude masks
5) Add/Edit Subscript Ranges for a Global
6) Delete All Subscript Ranges for a Global
7) Delete All Subscript Ranges
8) View Defined Subscript Ranges

Option?
```

7.4 ^DATACHECK Routine

You can use the **^DATACHECK** routine (in the %SYS namespace) to configure and manage the data checking. To obtain Help at any prompt, enter ?.

To start the **^DATACHECK** routine, do the following:

1. Enter the following commands in the Terminal:

```
ZNSPACE "%SYS"
%SYS>do ^DATACHECK
```

2. The main menu is displayed. Enter the number of your choice or press **Enter** to exit the routine:

```
1) Create New Configuration
2) Edit Configuration
3) View Details
4) Start
5) Stop
6) Delete Configuration
7) Incoming Connections to this System as a DataCheck Source

Option?
```

Note: For options 2 through 6, if you created multiple destination systems, a list is displayed so that you can select the destination system on which to perform the action.

The main menu lets you select DataCheck tasks to perform as described in the following table:

Option	Description
`1) Create New Configuration`	Prompts for the name of a new DataCheck destination system configuration via the Create New Configuration prompt.
`2) Edit Configuration`	Displays the Edit Configuration submenu.
`3) View Details`	Displays the View Details submenu.
`4) Start`	Starts/restarts the destination system. If you are restarting, it resumes from where you stopped it.
`5) Stop`	Stops the destination system. If you restart the destination system after stopping it, it resumes from where you stopped it.
`6) Delete Configuration`	Deletes the specified destination system configuration.
`7) Incoming Connections to this System as a DataCheck Source`	Displays the Incoming Connections to this System as a DataCheck Source submenu. **Note:** This option must be selected on a source system.

7.4.1 Create New Configuration

This submenu lets you configure the destination system. When you select this option, the following prompt is displayed:

```
Configuration Name:
```

If you are creating a DataCheck configuration on a system that *is not* a mirror member, the **Edit Settings** submenu is displayed, and you complete the configuration manually as described in Editing DataCheck Configurations on Non-mirror-based Systems.

If you are creating a DataCheck configuration on a system that *is* a mirror member, you are prompted for additional information that is dependent upon whether or not you want to base the data checking on mirroring. Choosing to configure DataCheck that is not based on mirroring displays the **Edit Settings** submenu, which you use to complete the configuration manually as described in Editing DataCheck Configurations on Non-mirror-based Systems. However, choosing to configure DataCheck based on mirroring restricts data checking to mirrored databases, and subsequent prompts are dependent on whether the destination system is a failover or async mirror member; for more information, see DataCheck for Mirror Configurations in this chapter.

7.4.2 Edit Configuration

The submenu lets you modify the destination system configurations. The options in the submenus are different depending on whether you are editing mirror-based or non-mirror-based configurations. For more information, see the following subsections:

- Editing DataCheck Configurations on Non-mirror-based Systems

- Editing Mirror-based DataCheck Configurations

7.4.2.1 Editing DataCheck Configurations on Non-mirror-based Systems

On a non-mirror-based system, when you select this option, the following prompts are displayed:

```
Configuration Name: dc_test

1) Import Settings from a Shadow   (static)
2) Connection Settings             (static)
3) Database Mappings               (static)
4) Globals to Check                (dynamic)
5) Performance Settings            (dynamic)
6) Manage Workflow                 (dynamic)

Option?
```

Note: In edit mode, if you created multiple destination systems, a list is displayed so that you can select a destination system to edit. In addition, before you edit the settings for options 1 through 3, you must stop the system.

Enter the number of your choice or press ^ to return to the previous menu. The options in this submenu let you configure the destination system as described in the following table:

Option	Description
`1) Import Settings from a Shadow`	If you are using DataCheck to verify that the source and destination of a shadowing system are synchronized, the option lets you import settings from an existing shadowing system.
`2) Connection Settings`	Information to connect to the source system.
`3) Database Mappings`	Lets you add, delete, or list database mappings on the source and destination systems.
`4) Globals to Check`	Globals to check or exclude from checking. For more information, see Specifying Globals and Subscript Ranges to Check in this chapter.
`5) Performance Settings`	Adjusts system resources (throttle) used and/or granularity with which DataCheck isolates discrepancies (minimum query size). For more information, see Performance Considerations in this chapter.
`6) Manage Workflow`	Manages the order of workflow phases. For more informations, see DataCheck Workflow in this chapter.

7.4.2.2 Editing Mirror-based DataCheck Configurations

On a mirror-based system, the following submenu is displayed:

```
Configuration Name: MIRRORSYS2_MIRRORX201112A_1

1) Globals to Check
2) Performance Settings
3) Manage Workflow
4) Change Mirror Settings (Advanced)

Option?
```

Enter the number of your choice or press ^ to return to the previous menu. The options in this submenu let you configure the destination system as described in the following table:

Option	Description
`1) Globals to Check`	Globals to check or exclude from checking. For more information, see Specifying Globals and Subscript Ranges to Check in this chapter.
`2) Performance Settings`	Adjusts system resources (throttle) used and/or granularity with which DataCheck isolates discrepancies (minimum query size). For more information, see Performance Considerations in this chapter.
`3) Manage Workflow`	Manages the order of workflow phases. For more informations, see DataCheck Workflow in this chapter.
`4) Change Mirror Settings (Advanced)`	See Planning DataCheck within the Mirror in the "Mirroring Considerations" section of this chapter

7.4.3 View Details

This submenu lets you monitor the status of the destination system, as well as view detailed information about the queries that are running and the results of data checking:

```
System Name: dc_test

1) View Status
2) View Results
3) View Queries
3) View Log

Option?
```

Enter the number of your choice or press ^ to return to the previous menu. The options in this submenu let you view information about the destination system as described in the following table:

Option	Description
`1) View Status`	Displays information about the selected destination system, including performance metrics for the DataCheck worker jobs, the source and state, current phase, workflow timeout, new phases requested, percentage of queries completed in the current phase, and the number of discrepancies recorded in this phase.
`2) View Results`	Displays the results for the selected destination system. For more information, see DataCheck Results in this chapter.
`3) View Queries`	Displays information about the queries submitted by the selected destination system (see DataCheck Queries). This includes the globals that remain to be processed (or global ranges if subscript include/exclude ranges are used), and indicates the active queries currently being worked on by DataCheck. A summary count is displayed at the end of the list.
`4) View Log`	Displays the selected destination system log file.

Note: When **^DATACHECK** is run against the two copies of a mirrored database on two mirror member instances, and that database is experiencing the rapid setting and killing of a whole global, it can display confusing results from the `View Status` option when compared to the `View Results` option. For example, it will report that there are unmatched answers in status, but will not actually report the globals that caused these answers in results (because further passes resolved the discrepancies). In addition, displayed answer counts can be larger than the actual number of globals within the instance (as displayed in the management portal, and as actually reported in the results).

When `View Status` shows **Answers Rcvd** having a non-zero unmatched value but discrepancies having a zero value, this is indicative of transient globals, not a data issue.

7.4.4 Incoming Connections to this System as a DataCheck Source

This submenu lets you view information about the source system:

```
1) List Source Systems
2) View Log

Option?
```

Enter the number of your choice or press **^** to return to the previous menu. The options in this submenu let you view information about the source system as described in the following table:

Option	Description
`1) List Source Systems`	Displays information about the DataCheck source system.
`2) View Log`	Displays the source system log file.

7.5 Special Considerations for Data Checking

Review the following special considerations when using DataCheck:

- Performance Considerations
- Security Considerations

7.5.1 Performance Considerations

While data checking is useful to ensure consistency of databases on multiple systems, it consumes resources on both the source and destination systems. This could negatively impact performance of other processes on either system, depending on load and the configured DataCheck settings. DataCheck includes controls to help you manage performance.

The *throttle* is an integer between 1 and 10 that controls how much of the available system resources (CPU, disk I/O, database cache) DataCheck may use. The throttle value can be changed at any time, to take effect immediately; for example, the value can be increased during periods when the system load is otherwise expected to be light, and decreased during periods when system load is heavy. This is useful for checks that are expected to run for an extended period of time. (The DataCheck routine can also be stopped during periods of high load; upon being restarted, it automatically resumes at the point in the check at which it was stopped.)

The characteristics of every system are different, but the following general descriptions of throttle values apply:

- A throttle setting of `1` uses no more resources than one process for performing DataCheck queries. In other words, it uses at most one CPU and executes only one disk I/O at a time. Whether the resources used are primarily CPU or primarily disk I/O depends on whether the data is in buffers already; this can vary as the check progresses.
- As the throttle is raised up to `8`, more system resources are consumed at each step. For systems with large amounts of resources (many CPUs, and so on), each interval is scaled to increase resource consumption by, very roughly, the same multiplicative factor, such that at a throttle setting of `8`, DataCheck uses a large portion of system resources, taking into account the number of CPUs and other factors. At a throttle setting of `8`, however, the system is still expected to be responsive to a light load of application activity, and settings of `6`, `7`, or `8` may be appropriate on a typical system at off-peak hours.
- A throttle setting of `9` is like `8`, but allows DataCheck jobs to use the entire buffer pool (unsets the batch flag).
- A throttle setting of `10` attempts to utilize nearly all system resources for completing the check.

The `View Status` option on the **^DATACHECK** `View Details` submenu shows performance metrics for the DataCheck worker jobs, helping you understand performance characteristics and how they relate to the throttle setting.

The implementation of the throttle may differ over time as software and hardware characteristics evolve.

The *minimum query size* represents the minimum number of global nodes allowed to traverse a query; in other words, it determines the minimum size of the range of global nodes to which DataCheck isolates discrepancies. Lower values help locate discrepancies more easily, while higher values significantly improve the speed of the check through unmatched sections. For example, if the minimum query size were set to `1` (not recommended), each discrepant node could be reported as a separate unmatched range, or at least as a range of all unmatched globals, precisely identifying the discrepancies but greatly impacting performance; if the minimum query size were set to `1000` (also not recommended), one or more discrepancies would be reported as a range of at least 1000 unmatched nodes, making it difficult to find them, but the check would be much faster. The default is `32`, which is small enough to allow for relatively easy visual inspection of the global nodes in a range using the Management Portal (see the "Managing Globals" chapter of *Using Caché Globals*) while not greatly impacting performance.

7.5.2 Security Considerations

The destination system stores subscript ranges for globals that it has checked and is checking (results and queries). (See Specifying Globals and Subscript Ranges to Check in this chapter.) This subscript data is stored in the ^SYS.DataCheck* globals in the %SYS namespace (in the CACHESYS database by default). Global values are not stored; only subscripts are stored. These global subscripts from other databases that are stored in the %SYS namespace may contain sensitive information that may not otherwise be visible to some users, depending on the security configuration. Therefore, some special care is needed in secured deployments.

Use of the **^DATACHECK** routine, including the ability to configure, start, and stop, requires both **`%Admin_Operate:Use`** privilege and **`Read/Write`** privilege (**`Write`** for configuring a check, **`Read`** for all other tasks) on the database containing the ^SYS.DataCheck* globals which, by default, is CACHESYS. The configuration and results data stored in the ^SYS.DataCheck* globals can be viewed and manipulated outside of the routine by anyone with sufficient database privileges.

For any secure deployment in which **`%DB_CACHESYS:Read`** privilege is given to users that should not have access to DataCheck data, you can add a global mapping to the %SYS namespace to map ^SYS.DataCheck* globals to a separate database other than CACHESYS. This database can be assigned a new resource name; read permission for the resource can then be restricted to those roles authorized to use DataCheck.

The ability for another destination system to connect to this system as a source is governed by this system's **`%Service_DataCheck`** service. This service is disabled by default on new installations and can be configured with a list of allowed IP addresses. For more information, see Enabling the DataCheck Service in this chapter.

For encryption of the communication between the two systems, the destination system can be configured to use SSL to connect to the source. See Configuring the Caché Superserver to Use SSL/TLS in the "Using SSL/TLS with Caché" chapter of the *Caché Security Administration Guide* for details.