Compiler Utilization: Tracking and Reporting

Compiler Utilization: Tracking and Reporting

Note

Before using this information and the product it supports, read the information in “Notices” on page 41.

Second edition

This edition applies to IBM XL C/C++ Enterprise Edition for AIX, V9.0 with the November 2010 PTF, IBM XL C/C++ for AIX, V10.1 with the October 2010 PTF , IBM XL C Enterprise Edition for AIX, V9.0 with the November 2010 PTF, IBM XL C for AIX, V10.1 with the October 2010 PTF, IBM XL Fortran Enterprise Edition for AIX, V11.1 with the November 2010 PTF or IBM XL Fortran for AIX, V12.1 with the October 2010 PTF.

Contents

About this document

This document contains overview and basic usage information for the compiler utilization and reporting feature for the IBM® XL C/C++ for AIX® compiler, V9.0 or V10.1, the IBM XL C for AIX, compiler V9.0 or V10.1, or the IBM XL Fortran for AIX, compiler V11.1 or V12.1.

Who should read this document

This document is intended for people who want to track XL C, XL C/C++ or XL Fortran compiler utilization in their organization and produce reports about their compliance with their license requirements for concurrent users. To track and create reports about your compiler utilization, you must have installed the appropriate compiler PTF and have downloaded the associated shared library and reporting tool. For details, see Chapter 2, "Prerequisites for tracking and reporting your compiler utilization," on page 3.

How to use this document

Throughout this document, the xlf, xlc and xlc++ compiler invocations are used to describe the actions of the compiler. You can, however, substitute other forms of the compiler invocation command if your particular environment requires it, and compiler option usage will remain the same unless otherwise specified.

The XL compilers provide several compiler invocation commands depending on source code languages and language levels. However, for convenience, this document uses only the basic **xlc**, **xlc++**, or **xlf**, invocation commands.

While this document covers information on the compiler utilization tracking and reporting feature of the compilers, you will need to access the main library of documentation for all other compiler information.

Conventions

Typographical conventions

The following table explains the typographical conventions used in this document.

Table 1. Typographical conventions

Typeface	Indicates	Example
bold	Lowercase commands, executable names, compiler options, and directives.	The compiler provides basic invocation commands, **xlc**, **xlC** (**xlc++**) or **xlf**, along with several other compiler invocation commands to support various programming language levels and compilation environments.
italics	Parameters or variables whose actual names or values are to be supplied by the user. Italics are also used to introduce new terms.	Make sure that you update the *size* parameter if you return more than the *size* requested.

Table 1. Typographical conventions (continued)

Typeface	Indicates	Example
<u>underlining</u>	The default setting of a parameter of a compiler option or directive.	nomaf | <u>maf</u>
`monospace`	Programming keywords and library functions, compiler builtins, examples of program code, command strings, or user-defined names.	To compile and optimize `myprogram.C`, enter: `xlC myprogram.c -O3`.

Qualifying elements (icons)

Most features described in this information apply to C, C++ and Fortran languages. In descriptions of language elements where a feature is exclusive to one language, or where functionality differs between languages, this information uses icons to delineate segments of text as follows:

Table 2. Qualifying elements

Qualifier/Icon	Meaning
C compiler ▶ C	The text describes a feature that is only for the XL C compiler
C++ compiler ▶ C++	The text describes a feature that is only for the XL C/C++ compiler
Fortran compiler FORTRAN	The text describes a feature that is only for the XL Fortran compiler

Syntax diagrams

Throughout this information, diagrams illustrate compiler syntax. This section will help you to interpret and use those diagrams.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►─── symbol indicates the beginning of a command, directive, or statement.

  The ───► symbol indicates that the command, directive, or statement syntax is continued on the next line.

  The ►─── symbol indicates that a command, directive, or statement is continued from the previous line.

  The ───►◄ symbol indicates the end of a command, directive, or statement.

  Fragments, which are diagrams of syntactical units other than complete commands, directives, or statements, start with the |─── symbol and end with the ───| symbol.

- Required items are shown on the horizontal line (the main path):

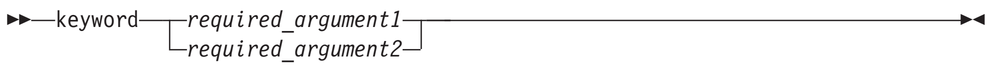
```
►►──keyword──required_argument──────────────────────────────────────────►◄
```

- Optional items are shown below the main path:

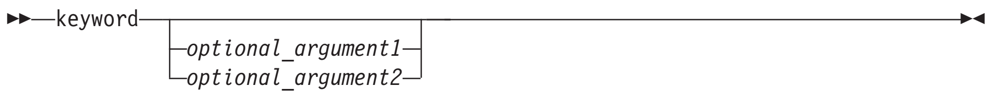
```
►►──keyword──┬───────────────────┬──────────────────────────────────────►◄
             └─optional_argument─┘
```

- If you can choose from two or more items, they are shown vertically, in a stack.

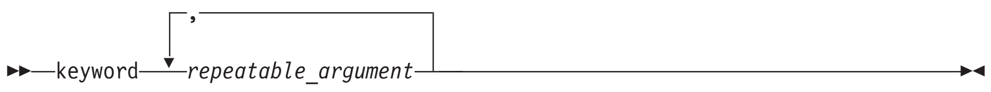
  If you *must* choose one of the items, one item of the stack is shown on the main path.



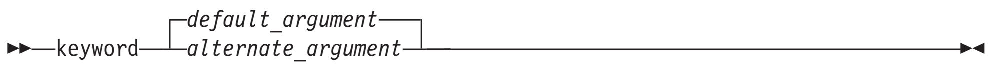
  If choosing one of the items is optional, the entire stack is shown below the main path.



- An arrow returning to the left above the main line (a repeat arrow) indicates that you can make more than one choice from the stacked items or repeat an item. The separator character, if it is other than a blank, is also indicated:



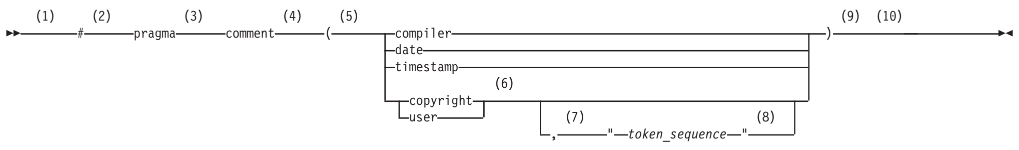
- The item that is the default is shown above the main path.



- Keywords are shown in nonitalic letters and should be entered exactly as shown.
- Variables are shown in italicized lowercase letters. They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Sample syntax diagram

The following syntax diagram example shows the syntax for the **#pragma comment** directive.



Notes:

1 This is the start of the syntax diagram.

2 The symbol # must appear first.

3 The keyword `pragma` must appear following the # symbol.

4 The name of the pragma `comment` must appear following the keyword `pragma`.

5 An opening parenthesis must be present.

6 The comment type must be entered only as one of the types indicated: `compiler`, `date`, `timestamp`, `copyright`, or `user`.

7 A comma must appear between the comment type `copyright` or `user`, and an optional character string.

8 A character string must follow the comma. The character string must be enclosed in double quotation marks.

9 A closing parenthesis is required.

10 This is the end of the syntax diagram.

The following examples of the #pragma comment directive are syntactically correct according to the diagram shown above:

```
#pragma comment(date)
#pragma comment(user)
#pragma comment(copyright,"This text will appear in the module")
```

Documentation

Documentation for this feature of the XL compilers is provided in the following:

- There is an installable man page for the utilization reporting tool that is available with the download.
- This PDF file.

There is a library of documentation available for your compiler. This includes both an information center of searchable HTML files and PDF documents. The library pages also include links to IBM Redbooks® publications, white papers, tutorials, and other articles. You can find links to these from your compiler library pages at:

- XL C for AIX library page
- XL C/C++ for AIX library page
- XL Fortran for AIX library page
- For C/C++ users, there is additional information about boosting performance, productivity, and portability on the C/C++ café

Technical support

Additional technical support is available from the compiler support page for your compiler:

- XL C for AIX support
- XL C/C++ for AIX support
- XL Fortran for AIX support

This page provides a portal with search capabilities to a large selection of Technotes and other support information.

If you cannot find what you need, you can send e-mail to compinfo@ca.ibm.com.

For the latest information about your compiler, visit the product information site at:

- XL C for AIX
- XL C/C++ for AIX
- XL Fortran for AIX

Chapter 1. Tracking and reporting compiler usage

You can use the utilization tracking and reporting feature to record and analyze which users in your organization are using the compiler and the number of users using it concurrently. This information can help you determine whether your organization's use of the compiler exceeds your compiler license entitlements.

To use this feature, follow these steps:

1. Understand how the feature works. See Chapter 3, “Understanding utilization tracking and reporting,” on page 5 for more information.
2. Investigate how the compiler is used in your organization, and decide how you track the compiler usage accordingly. See Chapter 4, “Preparing to use this feature,” on page 15 for more information.
3. Configure and enable utilization tracking. See Chapter 6, “Configuring utilization tracking,” on page 23 for more information.
4. Use the utilization reporting tool to generate usage reports or prune usage files. See Chapter 8, “Generating usage reports,” on page 33 or Chapter 9, “Pruning usage files,” on page 37 for more information.

Chapter 2. Prerequisites for tracking and reporting your compiler utilization

Ensure that you have the correct compiler versions and PTFs installed. You also need to download and install the `libut11.a` shared library and the `urt` binary file.

To install the PTFs and the associated download packages, you must have root access.

Compiler PTFs

The compilers and the associated PTF level that can use this feature are:
- IBM XL C for AIX, V10.1 with October 2010 PTF
- IBM XL C/C++ for AIX, V10.1 with October 2010 PTF
- IBM XL Fortran for AIX, V12.1 with October 2010 PTF
- IBM XL C Enterprise Edition for AIX, V9.0 with November 2010 PTF
- IBM XL C/C++ Enterprise Edition for AIX, V9.0 with November 2010 PTF
- IBM XL Fortran Enterprise Edition for AIX, V11.1 with November 2010 PTF

Utilization tracking library download

To track compiler utilization, you must download a shared library. The `libut11.a` shared library is contained in the `libut11` fileset. This fileset is contained in the `libut11.tar.Z` download. The download site is http://www-01.ibm.com/support/docview.wss?uid=swg24027640.

This download includes the following installp filesets:

libut11

libut11.licAgreement

It also contains License Agreement PDF files and a README file. The shared library installs into the `/usr/lib/libut11/` directory.

Utilization reporting tool download

To generate reports on compiler utilization, you must also have installed the utilization reporting tool. The `urt` binary file and its associated runtime library, message catalogs, man pages and readme files are contained in the `urt11.tar.Z` download. This tool is available from the download site

This download includes the following installp filesets:

urt11

urt.man.*Lang* where *Lang* is the language locale

urt.msg.*Lang*

The utilization reporting tool is installed into the `/opt/ibmurt/1.1/` directory.

Chapter 3. Understanding utilization tracking and reporting

The utilization tracking and reporting feature provides a mechanism for you to detect whether your organization's use of the compiler exceeds your compiler license entitlements. This section introduces the feature, describes how it works, and illustrates its typical usage scenarios.

Overview

When utilization tracking is enabled, all compiler invocations are recorded in a file. This file is called a usage file and it has the `.cuf` extension. You can then use the utilization reporting tool to generate a report from one or more of these usage files, and optionally prune the usage files.

You can use the utilization tracking and reporting feature in various ways based on how the compiler is used in your organization. "Four usage scenarios" on page 6 illustrates the typical usage scenarios of this feature.

The following sections introduce the configuration of the utilization tracking functionality and the usage of the utilization reporting tool.

Utilization tracking

A utilization tracking configuration file is included with the compiler PTF that includes this feature. You can use this file to enable utilization tracking and control different aspects of the tracking. The name of this file depends on which compiler version you have installed. For example the following configuration files might be installed:

- ▶ C `urtxlc0900aix.cfg` is shipped with XL C Enterprise Edition for AIX, V9.0, and `urtxlc1001aix.cfg` is shipped with XL C for AIX, V10.1
- ▶ C++ `urtxlc_cpp0900aix.cfg` is shipped with XL C/C++ Enterprise Edition for AIX, V9.0, and `urtxlc_cpp1001aix.cfg` is shipped with XL C/C++ for AIX, V10.1
- FORTRAN `urtxlf1101aix.cfg` is shipped with XL Fortran Enterprise Edition for AIX, V11.1, and `urtxlf1201aix.cfg` is shipped with XL Fortran for AIX, V12.1

A symlink `urt_client.cfg` is also included in the default compiler installation. It points to the location of the utilization tracking configuration file. If you want to put the utilization tracking configuration file in a different location, you can modify the symlink accordingly.

For more information, see Chapter 6, "Configuring utilization tracking," on page 23.

Note: Utilization tracking is disabled by default.

Utilization reporting tool

The utilization reporting tool generates compiler usage reports based on the information in the usage files. You can optionally prune the usage files with the tool. For more information, see Chapter 8, "Generating usage reports," on page 33 and Chapter 9, “Pruning usage files,” on page 37.

Four usage scenarios

This section describes four possible scenarios for managing the compiler usage, for recording the compiler usage information and for generating reports from this information.

The following scenarios describe some typical ways that your organization might be using the compiler and illustrates how you can use this feature to track compiler usage in each case.

Note: Actual usage is not limited to these scenarios.

“Scenario: One machine, one shared .cuf file”

“Scenario: One machine, multiple .cuf files” on page 8

“Scenario: Multiple machines, one shared .cuf file” on page 10

“Scenario: Multiple machines, multiple .cuf files” on page 12

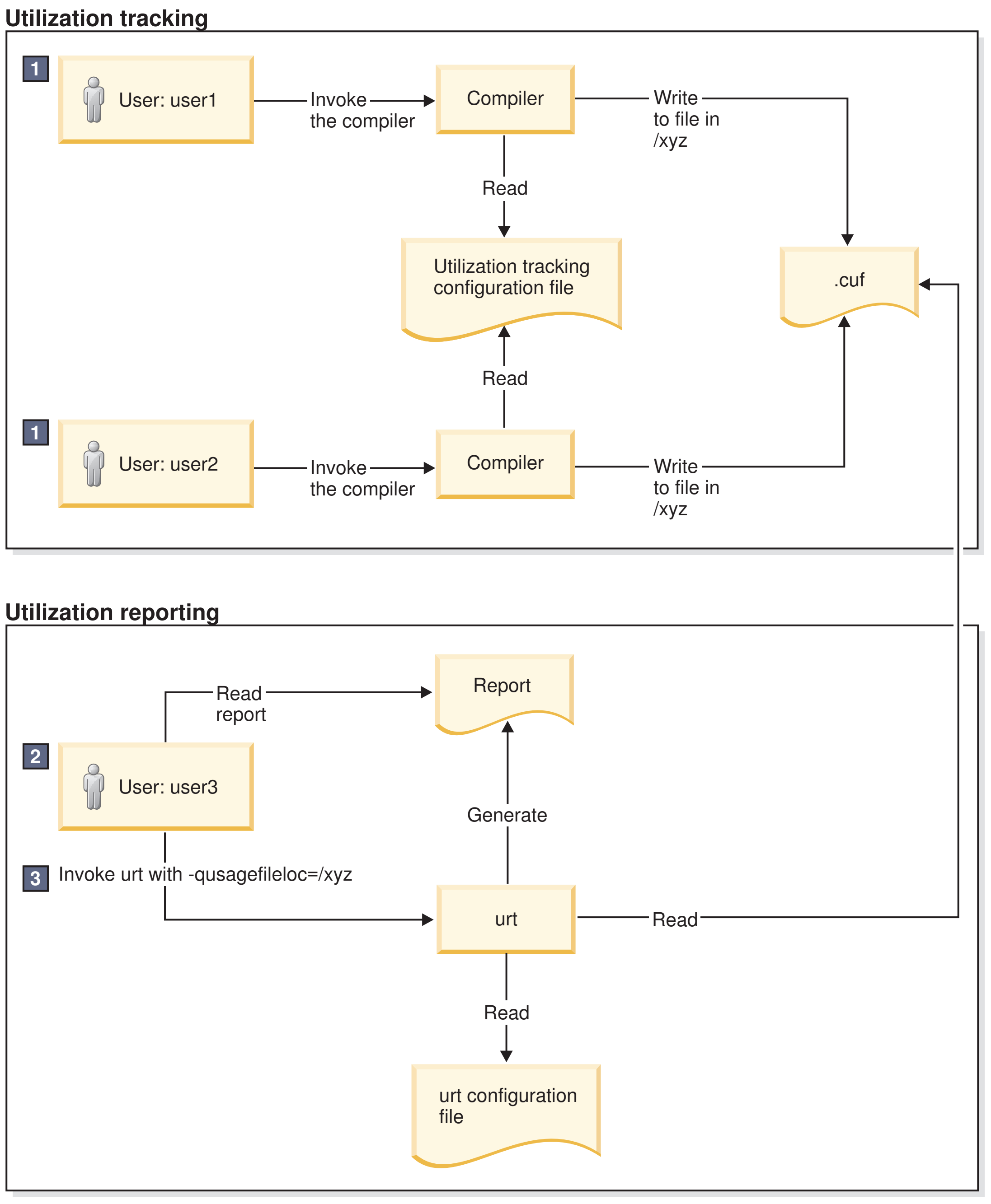
Scenario: One machine, one shared .cuf file

This scenario describes an environment where all the compilations are done on one machine and all users share one `.cuf` file.

The advantage of using the approach in this scenario is that it simplifies report generation and usage file pruning, because the utilization report tool only need to access one `.cuf` file. The disadvantage is that all compiler users need to compete for access to this file. Because the file might become large, it might have an impact on performance. Some setup work is also required to create the shared `.cuf` file and to give all compiler users write access. The “The number of usage files” on page 17 section provides detailed information about using a single usage file for all compiler users.

In this scenario, two compiler users run the compiler on the same machine and their utilization information is recorded in a shared `.cuf` file. The utilization tracking configuration file for the compiler is modified to point to the location of the `.cuf` file. When the compiler is invoked, it writes the utilization information to that file. You can then use the utilization reporting tool to retrieve the utilization information from the file and generate usage reports.

The following diagram illustrates this scenario.



1. Both user1 and user2 need write access to the `.cuf` file in `/xyz`.
2. user3 needs read access to the `.cuf` file in`/xyz` to generate the usage report, and write access to prune the `.cuf` file.
3. A cron job can be created to run urt automatically on a regular basis.

Figure 1. Compiler users use a single machine, with a shared .cuf file

The diagram reflects the following points:

1. user1 and user2 use the same utilization tracking configuration file, which manages the tracking functionality centrally. A common location `/xyz` is created to keep a shared `.cuf` file.
2. When user1 and user2 invoke the compiler, the utilization information is recorded in the `.cuf` file under the common directory `/xyz`.

3. user3 invokes urt with `-qusagefileloc=/xyz` to generate usage reports.

Note: Regular running of the utilization reporting tool can prevent these files from growing too big, because you can prune the usage files with this tool.

Scenario: One machine, multiple .cuf files

This scenario describes an environment where all the compilations are done on one machine and all users have their own `.cuf` files.

The approach in this scenario has the following advantages:

- Compiler users do not have to compete for access to a single `.cuf` file, and this might result in better performance.
- You do not need to set up write access to a single common location for all compiler users. They already have write access to their own home directories.

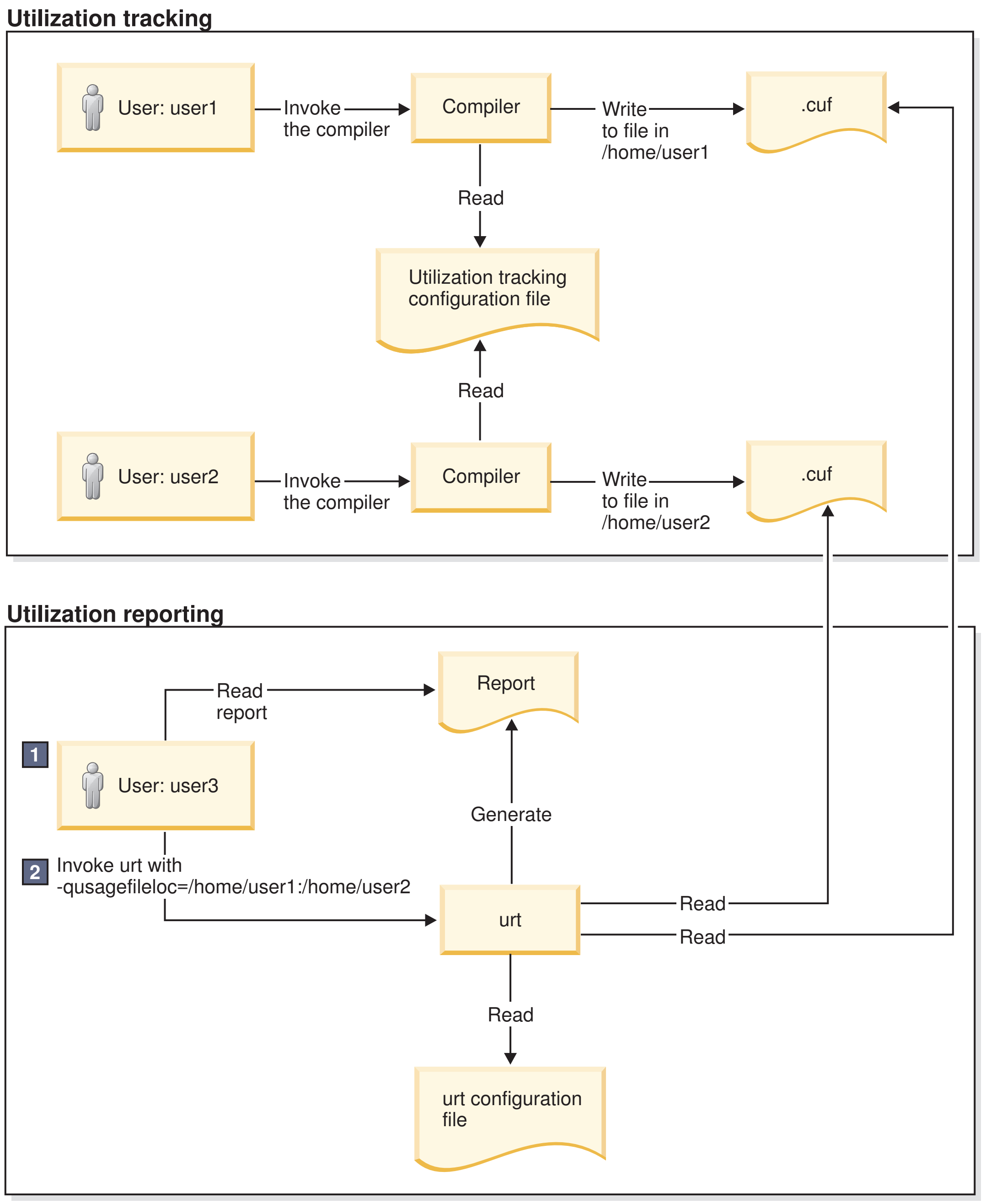
However, using multiple `.cuf` files that are automatically created in each user's home directory might have the following issues:

- Compiler users might not know that the file has been created or what it is when they see the file. In this case, they might delete the file.
- Some users' home directories might be on file systems that are mounted from a remote system. This causes utilization tracking to use a remote file, which might affect performance.
- Compiler users might not want `.cuf` files to take up space in their /home directories.

Instead of using each user's home directory, the `.cuf` files for each user can be created in a common location. The "Usage file location" on page 17 section provides detailed information about how to create these files in a common location.

In this scenario, two compiler users run the compiler on the same machine and they have their own `.cuf` files. When the compiler is invoked, it automatically creates a `.cuf` file for each user and writes the utilization information to that file. You can then use the utilization reporting tool to retrieve the utilization information from the file and generate usage reports.

The following diagram illustrates this scenario.



1. user3 needs read access to `.cuf` files in `/home/user1` and `/home/user2` to generate the usage report, and write access to prune the usage files.
2. A cron job can be created to run urt automatically on a regular basis.

Figure 2. Compiler users use one machine, with separate .cuf files

This diagram reflects the following points:

1. user1 and user2 use the same utilization tracking configuration file, which manages the tracking functionality centrally.
2. When user1 and user2 invoke the compiler, the utilization information is recorded in the two `.cuf` files under their respective home directories, `/home/user1` and `/home/user2`.
3. user3 invokes urt with `-qusagefileloc=/home/user1:/home/user2` to generate usage reports.

Note: If you need to find out which home directories contain usage files, you can invoke urt as follows:

```
urt -qusagefileloc=/home -qmaxsubdirs=1
```

In this case, urt looks for `.cuf` files in all users' home directories.

Scenario: Multiple machines, one shared .cuf file

This scenario describes an environment where the compilations are done on multiple machines but all users share a single `.cuf` file.

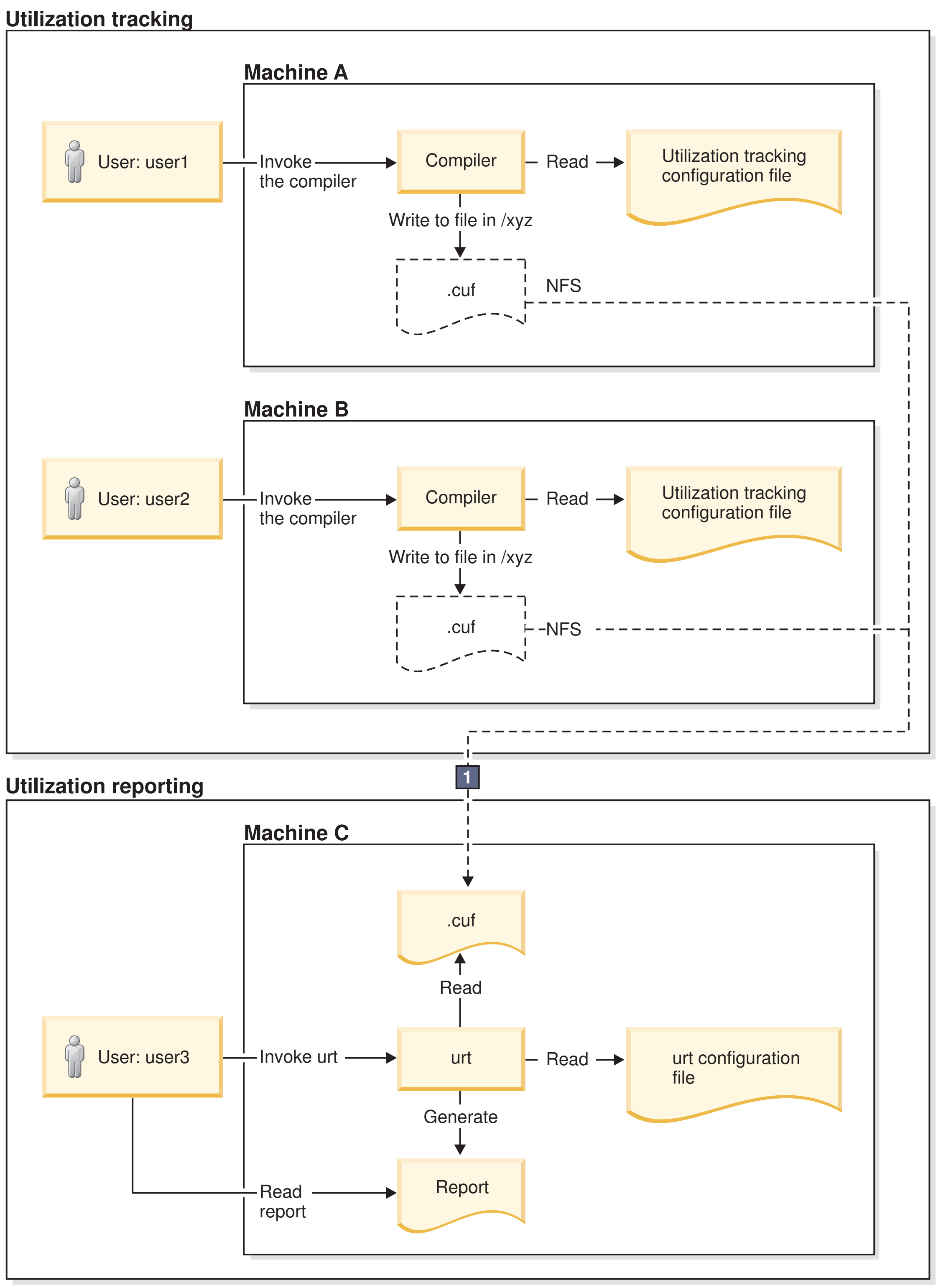
The advantage of the approach in this scenario is that using one `.cuf` file can simplify the report generation and the usage file pruning process. The section "The number of usage files" on page 17 provides detailed information about using a single usage file for all compiler users. The `.cuf` file is already on the machine where the utilization reporting tool is installed. You do not need to copy the file to that machine or install the tool on multiple machines to prune the `.cuf` files.

This approach has the following disadvantages:

- The compiler users must compete for access to one usage file. Because the file might become large, it might have an impact on performance.
- Some setup work is required to create the shared `.cuf` file and to give all compiler users write access on a network file system.
- The efficiency of the whole process depends on the speed and reliability of the network file system, because the compilers and the `.cuf` file are on different machines. For example, some file systems are better than others in supporting file locking, which is required for concurrent access by multiple users.

In this scenario, two compiler users run the compilers on separate machines and they use one shared `.cuf` file on a network file system, such as NFS, DFS, or AFS. When the compiler is invoked, it writes the utilization information to that file. You can then use the utilization reporting tool to retrieve the utilization information from the file and generate usage reports.

The following diagram illustrates this scenario.



1. On Machine A and Machine B, mount point `/xyz` is created to Machine C. All compiler utilization is recorded in the `.cuf` file, from which the usage report is generated.

Figure 3. Compiler users use multiple machines, with a shared .cuf file

This diagram reflects the following points:

1. Utilization tracking is configured respectively on Machine A and Machine B.

Notes:

- Although each machine has its own configuration file, the contents of these files must be the same.
- Centrally managing the utilization tracking functionality can reduce your configuration effort and eliminate possible errors. The "Central configuration" on page 15 section provides detailed information about how you can use a common configuration file shared by compiler users using different machines.

2. A network file system is set up for the central management of the `.cuf` files. When user1 and user2 invoke the compilers from Machine A and Machine B, the utilization information of both compilers is written to the `.cuf` file on Machine C.
3. user3 invokes urt to generate usage reports from the `.cuf` file on Machine C.

Note: You can use the utilization reporting tool to prune the usage files regularly to prevent them from growing too big.

Scenario: Multiple machines, multiple .cuf files

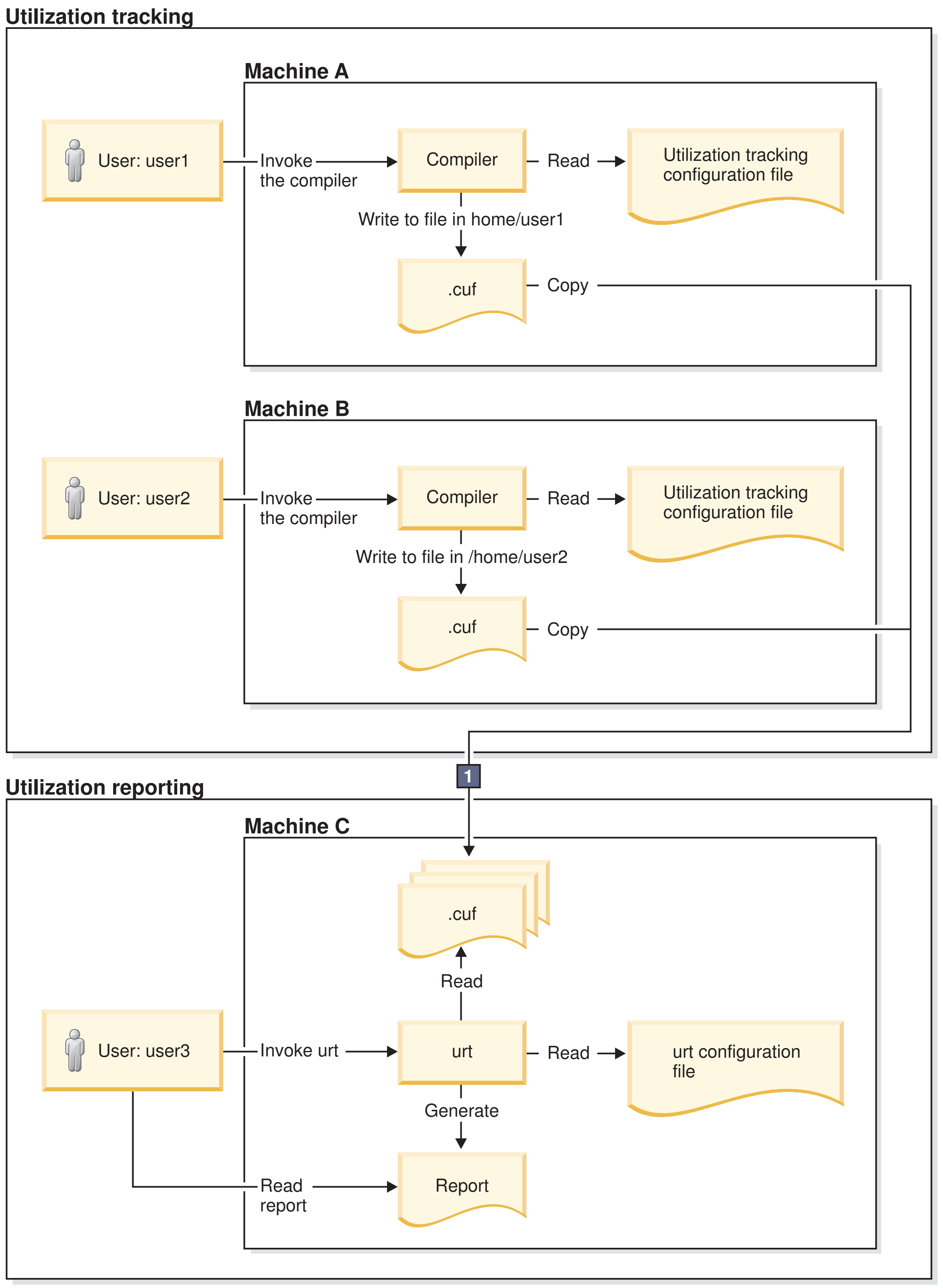
This scenario describes an environment where the compilations are done on multiple machines and all users have their own usage files.

In this scenario, two compiler users run the compilers on separate machines and they have their own `.cuf` files. When the compiler is invoked, it writes the utilization information to that file. You can then use the utilization reporting tool to retrieve the utilization information from the file and generate usage reports. This tool can be run on either of the machines on which the compiler is installed or on a different machine.

Note: The utilization reporting tool requires access to all the `.cuf` files.
You can use either of the following methods to make the files accessible in this example:

- Use a network file system, such as NFS, DFS, or AFS.
- Copy the files from their original locations to the machine where you plan to run the utilization reporting tool. You can use ftp, rcp, rsync or any other remote copy command to copy the files.

The following diagram illustrates this scenario.



1. user3 copies the `.cuf` files to Machine C. A cron job can be created to copy the files automatically on a regular basis.

Figure 4. Compiler users use multiple machines, with multiple .cuf files

This diagram reflects the following points:

1. Utilization tracking is configured respectively on Machine A and Machine B.

Notes:
- Although each machine has its own configuration file, the contents of these files must be the same.
- Centrally managing the utilization tracking functionality can reduce your configuration effort and eliminate possible errors. The "Central configuration" on page 15 section provides detailed information about how you can use a common configuration file shared by compiler users using different machines.

2. When user1 and user2 invoke the compilers, the utilization information is recorded in the two `.cuf` files under their respective home directories, `/home/user1` and `/home/user2`.

   Note: These `.cuf` files can also be created in another common location, for example, `/var/tmp`. The "Usage file location" on page 17 section provides detailed information about how to create these files in a common location.
3. user3 copies the two `.cuf` files from Machine A and Machine B to Machine C.
4. user3 invokes urt to generate usage reports from the `.cuf` files on Machine C.

Related information
- Chapter 4, "Preparing to use this feature," on page 15
- Chapter 6, "Configuring utilization tracking," on page 23
- Chapter 8, "Generating usage reports," on page 33
- Chapter 9, "Pruning usage files," on page 37

Chapter 4. Preparing to use this feature

Before enabling utilization tracking within your organization, you must consider certain factors related to how the compiler is used in your organization.

The following sections describe those considerations in detail:

Time synchronization

If you plan to track the utilization of the compiler on more than one machine, you must consider synchronizing the time across the machines.

The usage report generated by the utilization reporting tool lists the time when the compiler invocations start and end. The report also determines which invocations are concurrent. This information is much less reliable and useful if time is not synchronized across these machines.

If you are unable to synchronize time across different machines, you can use the **-qadjusttime** option to instruct the utilization reporting tool to adjust the times that have been recorded.

License types and user information

Before you start to use this feature, you need the number and type of license entitlements for your organization.

The license and user information that you need is as follows:

- The number of Concurrent User licenses that you have for this compiler. This information is required for the **-qmaxconcurrentusers** entry in the utilization tracking configuration file.
- The users who have their own Authorized User license for this compiler. This information is used for the **-qexemptconcurrentusers** entry in the utilization tracking configuration file.
- The users who use the compiler with multiple accounts. This information is used for the **-qsameuser** option for the utilization reporting tool.

Note: It is not mandatory to specify the users who have their own Authorized User license and the users who use the compiler with multiple accounts, but it improves the accuracy of the usage reports generated by the utilization reporting tool. For detailed information, see “Concurrent user considerations” on page 16.

Central configuration

Configuring utilization tracking the same for all compiler users is very important, because it can ensure the accuracy of your utilization tracking, and minimize your configuration and maintenance effort. You can achieve this by ensuring that all users use the same utilization tracking configuration file.

If you have only one installation of the compiler, you can directly edit the utilization tracking configuration file. Every compiler user can automatically use that configuration file.

If you have multiple installations of the compiler, you need to maintain a single utilization tracking config file and reference it from all installations. Any changes you make to the utilization tracking configuration file, including enabling or disabling utilization tracking, can automatically apply to all compiler installations when users invoke the compiler. In each installation, there is a symlink named `urt_client.cfg`, located in `usr/vacpp/urt`, `usr/vac/urt`, or `/usr/lpp/xlf/urt`. Modify the symlink to point to this shared instance of the configuration file.

If the compiler is installed on multiple machines, the utilization tracking configuration file needs to be placed on a network file system, such as NFS, DFS, or AFS, to be used by the compiler on each machine.

Note: If it is not possible for you to use a single utilization tracking configuration file for all compiler users, you must ensure all utilization tracking configuration files for each compiler installation are consistent. Using different configurations for the same compiler is not supported.

Concurrent user considerations

Invocations of the compiler are considered concurrent when their start time and end times overlap. This section provides the information about how the utilization reporting tool counts concurrent users and the ways to increase the accuracy of the usage reports.

When the utilization reporting tool counts concurrent users, it looks at the user account information that has been captured in the usage files. The account information consists of a user name, a user ID, and a host name. By default, each unique combination of this account information is considered and counted as a different user. However, invocations of the compiler by the following users must not be included in the count of concurrent users:

- Users who have their own Authorized User license are considered exempt users, because their use of the compiler does not consume any Concurrent User licences.
- Users who have multiple accounts. Because the accounts belong to the same user, invocations of the compiler while logged on using those accounts are counted as usage by a single user.

The utilization reporting tool can account for the above situations if you provide it with information regarding exempt users and users with multiple accounts. Here is how you can provide the information:

- Specify the **-qexemptconcurrentusers** entry in the utilization tracking configuration file. This entry specifies users with Authorized User licenses.
- Specify the **-qsameuser** urt command-line option. This option specifies users with multiple accounts.

Notes:

- When the number of concurrent users is adjusted with **-qexemptconcurrentusers** or **-qsameuser**, the utilization reporting tool generates a message to indicate that the concurrent usage information is adjusted.
- The number of concurrent users might be zero if all concurrent invocations are invoked by exempt users. The tool also generates a message with this information.

Usage file considerations

Usage (`.cuf`) files are used to store compiler usage information. This section provides information that helps you decide how you want to generate and use these files.

Usage file location

Usage files can be created in each user's home directory, or they can be created in a central location for all users.

With utilization tracking enabled, when a compiler user compiles a program, a `.cuf` file is automatically created in the user's home directory in case the file does not exist. This is convenient for testing the utilization tracking feature because users already have write access to their own home directories, which means no additional setup is required. However, this might have the following issues:

- Compiler users might not know that the file has been created or what it is when they see the file. In this case, they might delete the file.
- Some users' home directories might be on file systems that are mounted from a remote system. This causes utilization tracking to use a remote file, which might affect performance.
- Compiler users might not want usage files to take up space in the `/home` directory.

A good alternative is to set up a central location where the usage files can be created, and provide appropriate access to that location for both the compiler users and the utilization reporting tool users. This can be set up by using the `other/world` permissions or by using group permissions.

For example, if the central location is a directory named `/var/tmp/track_compiler_use`, you can modify the **-qusagefileloc** entry in the utilization tracking configuration file as follows:

```
-qusagefileloc=/var/tmp/track_compiler_use/$LOGNAME.cuf
```

This creates a `.cuf` file for each user in the specified location, such as `user1.cuf` or `user2.cuf`. It is easier to run the utilization reporting tool to generate the usage report from the `.cuf` files in this central location. You only need to pass the path of the location, `/var/tmp/track_compiler_use` to the utilization reporting tool , and then the tool can read all the `.cuf` files in that location.

If the compiler users are running the compiler on more than one machine, you need to add *$HOSTNAME* to the **-qusagefileloc** entry to ensure that there are no collisions in the file names. For example, you can specify the **-qusagefileloc** entry as follows:

```
-qusagefileloc=/var/tmp/track_compiler_use/$HOSTNAME_$LOGNAME.cuf
```

This creates a `.cuf` file for each user, and the name of that `.cuf` file also contains the name of the host on which the compiler is used, such as `host1_user1.cuf`.

The number of usage files

You can use one usage file or separate usage files for different compiler users.

Using separate usage files for different compiler users

The advantages of using separate usage files are as follows:

- It might provide better performance because compiler users access their own usage files instead of competing for access to a shared one and separate usage files are usually smaller.
- Usage file for a user can be automatically created when the user uses the compiler to compile a program. There is no need to explicitly create a usage file for each user beforehand. For more information, see "Usage file location" on page 17.
- When generating utilization reports, you usually include all compiler users. However, if there are circumstances in which you want to exclude some users, you can simply omit their usage files when you invoke the utilization reporting tool. For example, you might want to omit users who have their own Authorized User license.

The disadvantage is that you might have to maintain separate usage files for different users.

Using a single usage file for all compiler users

The advantage of using a shared usage file for all users is that you only need to maintain a single file instead of multiple files. However, with a single usage file, you lose the flexibility and possible performance benefits of using multiple usage files, as described in the preceding subsection.

The compiler provides an empty usage file `urtstub.cuf` in the `usr/vac/urt` or `usr/lpp/xlf/urt/` directory. You can create a usage file for all compiler users by copying the empty usage file to a directory where they all have write access. In this case, you need to change the **-qusagefileloc** entry in the utilization tracking configuration file to point to the location of the usage file.

Usage files on multiple machines

If you use the compiler on multiple machines, you need to decide how to make the usage files available for the utilization reporting tool.

You can use various methods to make the usage files available for the utilization reporting tool to generate usage reports and prune the usage files. Choose one of the following approaches to manage usage files on multiple machines:

- Copy the usage files from the machines where the compiler is used to the machine where the utilization reporting tool is installed. You can use any remote copy command, for example, ftp, rcp, scp, and rsync. In this case, the usage files are being accessed locally by both the compiler, for utilization tracking, and by the utilization reporting tool, for generating the usage report. Accessing the files locally yields the best performance.
- Use a distributed file system to export the file system from the machines where the compiler is used, and mount those file systems on the machine where the utilization reporting tool is installed. When you run the utilization reporting tool, it can access the usage files remotely via the mounted file systems.
- You can also export the file system from the machine where the utilization reporting tool is installed, and mount that file system on each machine where the compiler is used, using it as the location of the usage files where the compiler is recording its utilization. In this approach, the compiler records utilization in a remote usage file, and the utilization reporting tool reads the usage file locally.

Note: If you find this degrades the performance of the compiler, consider using one of the first two approaches instead.

Usage file size

You need to consider the fact that the size of the usage files might grow quickly, especially when you use a shared file for all compiler users. If the usage file gets too large, it might affect utilization tracking performance.

To keep the usage files from growing quickly, you can optionally prune the usage files when you generate usage reports. You can also prune the files regularly using cron.

For more information about how to prune files, see Chapter 9, "Pruning usage files," on page 37.

Regular utilization checking

You can run the utilization reporting tool on a regular basis to verify whether the usage of the compiler is compliant with the Concurrent User licenses you have purchased. You can create a cron job to do this automatically.

If the usage files need to be copied to the machine where the utilization reporting tool is running, you can also automate the copying task with a cron job.

Another reason for running the tool regularly is to prune the usage files to control the size of these files.

Note: To reduce contention for read and write access to the usage files, run the utilization reporting tool or copy the usage files when the compiler is not being used.

Chapter 5. Testing utilization tracking

Before you begin to track the compiler usage for all users in your organization, you can test the feature with a limited number of users or with a separate compiler installation. During this testing, you can try different configurations so as to decide the best setup for your organization.

Testing with a limited number of users

To enable compiler utilization tracking for a limited number of users, you can use a separate utilization tracking configuration file and ask only these users to use the file. Other users of the same installation use the default utilization tracking configuration file in which utilization tracking is disabled, and their use of the compiler is therefore not recorded.

The default compiler configuration file, such as `vac.cfg.61`, contains two entries, *xlurt_cfg_path* and *xlurt_cfg_name*, which specify the location of the utilization tracking configuration file. You need to perform the following tasks to let the specified users use the separate utilization tracking configuration file:

1. Create a separate compiler configuration file or stanza, in which the *xlurt_cfg_path* and *xlurt_cfg_name* entries specify the location of the utilization tracking configuration file you want to use.
2. Ask these users to use the following compiler option or environment variable to instruct the compiler to use the separate compiler configuration file or stanza, which in turn allows them to use the separate utilization tracking configuration file.
   - The -F option
   - The **XLC_USR_CONFIG** (for XL C or XL C/C++ compilers) or **XLF_USR_CONFIG** (for XL Fortran compilers) environment variable.

Note: This approach is only for testing the utilization tracking feature. Do not use it for tracking all compiler utilization in your organization unless you can ensure that all compiler invocations are done with the -F option or the **XLC_USR_CONFIG** (for XL C or XL C/C++ compilers) or **XLF_USR_CONFIG** (for XL Fortran compilers) enironment variable set.

Testing with a separate compiler installation

You can install a separate instance of the compiler for testing utilization tracking. In this case, you can directly modify the utilization tracking configuration file in that installation to enable and configure utilization tracking. The compiler users involved in the testing do not need to perform any task for the tracking.

When you are satisfied that you have found the best utilization tracking configuration for your organization, you can enable it for all compiler users in your organization.

Related information

- Chapter 6, “Configuring utilization tracking,” on page 23

Chapter 6. Configuring utilization tracking

You can use the utilization tracking configuration file to enable and configure the utilization tracking functionality.

The file name and default location of the configuration file for your compiler is:

- C `urtxlc0900aix.cfg` or `urtxlc1001aix.cfg` in `/usr/vac/urt`,
- C++ `urtxlc_cpp0900aix.cfg` or `urtxlc_cpp1001aix.cfg` in `/usr/vacpp/urt`, or
- FORTRAN `urtxlf1101aix.cfg` or `urtxlf1201aix.cfg` in `/usr/lpp/xlf/urt`

The compiler uses a symlink to specify the location of the utilization tracking configuration file. The symlink is also located in `/usr/vacpp/urt`, `/usr/vac/urt`, or `/usr/lpp/xlf/urt` and its name is `urt_client.cfg`. In the following situations, you might need to change the symlink:

- If you want to use a utilization tracking configuration file in a different location, change the symlink to point to this location.
- If you have multiple installations of the same compiler, and you plan to use a single utilization tracking configuration file, change the symlink in each installation to point to that file. For more information, see "Central configuration" on page 15.

Note: Installing a PTF update does not overwrite the utilization tracking configuration file.

You can use the entries in the utilization tracking configuration file to configure many aspects of the way compiler usage is tracked. For details about the specific entries in that file and how they can be modified, see "Editing utilization tracking configuration file entries."

Editing utilization tracking configuration file entries

You can configure different aspects of utilization tracking by editing the entries in the utilization tracking configuration file.

The entries are divided into two categories.

1. The entries in the *Product information* category identify the compiler. Do not modify these entries.
2. The entries in the *Tracking configuration* category can be used to configure utilization tracking for this product. Changes to these entries take effect in the usage file upon the next compiler invocation. In this case, the compiler emits a message to indicate that the new configuration values have been saved in the usage file. When you generate a report from the usage file, the new values are used.

The following rules apply when you modify the entries:

- The following entries are written to the usage files whenever you change them, and they are used the next time the utilization reporting tool generates a report from the usage files. These configuration entries must be the same for all compiler users.

  - **-qmaxconcurrentusers**
  - **-qexemptconcurrentusers**
  - **-qqualhostname**
- If **-qqualhostname** is changed, you must discard any existing usage files and start tracking utilization again with new usage files. Otherwise some invocations are recorded with qualified host names and some are recorded with unqualified host names.

Notes:
- The entries are not compiler options. They can only be used in the utilization tracking configuration file.
- If the **-qexemptconcurrentusers** entry is specified multiple times in the utilization tracking configuration file, all the specified instances are used. If other entries are specified multiple times, the value of the last one overrides previous ones.
- The compiler generates a message if you specify the above entries with different values for different users when using more than one utilization tracking configuration file. You must modify the entries to keep them consistent, or make sure all compiler users use a single utilization configuration file.

Product information

-qprodId=*product_identifier_string*
Indicates the unique product identifier string.

-qprodVer=*product_version*
Indicates the product version.

-qprodRel=*product_release*
Indicates the product release.

-qprodName=*product_name*
Indicates the product name.

-qconcurrentusagescope=prod | ver | rel
Specifies the level at which concurrent users are counted, and their numbers are limited. The suboptions are as follows:
- prod indicates the product level.
- ver indicates the version level.
- rel indicates the release level.

Default: -qconcurrentusagescope=prod

Tracking configuration

-qmaxconcurrentusers=*number*

Specifies the maximum number of concurrent users. It is the number of Concurrent User license that you have purchased for the product. When the utilization reporting tool generates a report from the usage file, it determines whether your compiler usage in your organization has exceeded this maximum number of concurrent users.

Note: You must update this entry to reflect the actual number of Concurrent User licenses that you have purchased.

Default: 0

-qexemptconcurrentusers ="*user_account_info_1* **[|** *user_account_info_2* **| ... |** *user_account_info_n***]"**

Specify exempt users who have their own Authorized User license. Exempt users can have as many concurrent invocations of the compiler as they want, without affecting the number of Concurrent User licenses available in your organization. When the utilization reporting tool generates a usage report, it does not include such users in the count of concurrent users.

user_account_info can be any combination of the following items:

- name(*user_name*)
- uid(*user_ID*)
- host(*host_name*)

Users whose information matches the specified criteria are considered exempt users. For example, to indicate that *user1@host1* and *user2@host1* are exempt users, you can specify this entry in either of the following forms:

- `-qexemptconcurrentusers="name(user1)host(host1)"`
  `-qexemptconcurrentusers="name(user2)host(host1)"`
- `-qexemptconcurrentusers="name(user1)host(host1) | name(user2)host(host1)"`

For *user_name*, *user_ID*, and *host_name*, you can also use a list of user names, user IDs, or hostnames separated by a space within the parentheses. For example:

```
-qexemptconcurrentusers="name(user1 user2)host(host1)"
```

This is equivalent to the previous examples.

Note: Specifying this entry does not exempt users from compiler utilization tracking. It only exempts them from being counted as concurrent users. To optimize utilization tracking performance, the format of the specified value is not validated until the report is produced. For more information about counting concurrent users, see "Concurrent user considerations" on page 16.

-qqualhostname | -qnoqualhostname

Specifies whether host names that are captured in usage files and then listed in compiler usage reports are qualified with domain names.

If all compiler usage within your organization is on machines within a single domain, you can reduce the size of the usage files by using **-qnoqualhostname** to suppress domain name qualification.

Default: -qqualhostname, which means the host names are qualified with domain names.

-qenabletracking | -qnoenabletracking

Enables or disables utilization tracking.

Default: -qnoenabletracking, which means utilization tracking is disabled.

-qusagefileloc=*directory_or_ file_name*

Specifies the location of the usage file.

By default, a `.cuf` file is automatically created for each user in their home directory. You can set up a central location where the files for each user can be created. For more information, see "Usage file location" on page 17.

The following rules apply when you specify this entry:

- If a file name is specified, it must have the `.cuf` extension. If the file is a symlink, it must point to a file with the`.cuf` extension. If the specified file does not exist, a `.cuf` file is created, along with any parent directories that do not already exist.
- If a directory is specified, there must be exactly one file with the `.cuf` extension in the directory. A new file is not created in this case.
- The path of the specified directory can be a relative or an absolute path. Relative paths are relative to the compiler user's current working directory.

Note: If a compiler user cannot access the file, for example, because of insufficient permissions to use or create the file, the compiler generates a message and the compilation continues.

You can use the following variables for this option:

- *$HOME* for the user's home directory. This allows each user to have a `.cuf` file in their home directory or a subdirectory of their home directory.
- *$USER* or *$LOGNAME* for the user's login user name. You can use this variable to create a `.cuf` file for each user and include the user's login name in the name of the `.cuf` file or in the name of a parent directory.
- *$HOSTNAME* for the name of the host on which the compiler runs. This can be useful when you track utilization across different hosts.

-qfileaccessmaxwait=*number_of_milliseconds*

Specifies the maximum number of milliseconds to wait for accessing the usage file.

Note: This entry is used to account for unusual circumstances where the system is under extreme heavy load and there is a delay in accessing the usage file.

Default: 3000 milliseconds

Notes:

- These entries are not compiler options. They can only be used in the utilization tracking configuration file.
- If the **-qexemptconcurrentusers** entry is specified multiple times in the utilization tracking configuration file, all the specified instances are used. If other entries are specified multiple times, the value of the last one overrides previous ones.

Chapter 7. Understanding the utilization reporting tool

You can use the utilization reporting tool to generate compiler usage reports from the information in one or more usage files, and optionally prune the usage files when you generate the reports.

The tool is located in the `/opt/ibmurt/1.1/bin` directory. You can use the urt command to invoke it. The syntax of the urt command is as follows:

```
                ┌──────────────────────────┐
                ▼                          │
►►──urt─────────┴──┬──────────────────────┬┴──────────────────────────────────►◄
                   └─command_line_options─┘
```

The generated report is displayed on the standard output. You can direct the output to a file if you want to keep the report.

Command-line options control how usage reports are generated. For more information about the options, see "Utilization reporting tool command-line options."

A default configuration file `ibmurt.cfg` is provided in the `/opt/ibmurt/1.1/config` directory. Entries in this file take the same form as the command-line options and have the same effect. You can also create additional configuration files and use the **-qconfigfile** option to specify their names.

You can specify the options in one or more of the following places:
1. The default configuration file
2. The additional configuration file specified with **-qconfigfile**
3. The command line

The utilization reporting tool uses the options in the default configuration file before it uses the options on the command line. When it encounters a **-qconfigfile** option on the command line, it reads the options in the specified configuration file and puts them on the command line at the place where the **-qconfigfile** option is used.

If an option is specified multiple times, the last specification that the tool encounters takes effect. Exceptions are **-qconfigfile** and **-qsameuser**. For these two options, all specifications take effect.

Utilization reporting tool command-line options

The utilization reporting tool command-line options control the generation of the compiler utilization report.

Use these command-line options to modify the details of your compiler utilization report.

-qreporttype=detail | <u>maxconcurrent</u>

Specifies the type of the usage report to generate.

- detail specifies that all invocations of the compiler are listed in the usage report. With this suboption, you can get a detailed report, in which the invocations that have exceeded the allowed maximum number of concurrent users are indicated.
- maxconcurrent specifies that only the compiler invocations that have exceeded the allowed maximum number of concurrent users are listed. With this suboption, you can get a compact report, which does not list those invocations within the maximum number of allowed concurrent users.

Note: The allowed maximum number of concurrent users is specified with the **-qmaxconcurrentusers** entry in the utilization tracking configuration file.

Default: -qreporttype=maxconcurrent.

-qrptmaxrecords=*num* | **nomax**

Specifies the maximum number of records to list in the report for each product. *num* must be a positive integer.

Default: -qrptmaxrecords=nomax, which means all the records are listed.

-qusagefileloc=*directory_or_file_name*

Specifies the location of the usage files for report generation or pruning. It can be a list of directories or file names, or both, separated by colons.

The following rules apply when you specify this option:
- If one or more directories are specified, all files with the `.cuf` extension in those directories are used. Subdirectories can also be searched by using the **-qmaxsubdirs** option.
- The path of the specified directory can be relative or absolute. Relative paths are relative to the compiler user's current working directory.
- A symlink does not require the `.cuf` extension but the file to which it points must have that extension.

Note:
- The **-qusagefileloc** entry in the utilization tracking configuration file tells the compiler which usage files to use for recording compiler utilization. This **-qusagefileloc** option tells the utilization reporting tool where to find those usage files.

Default: *.:$HOME*, which means the utilization reporting tool looks for usage files in your current working directory and your home directory.

-qmaxsubdirs=*num* | **nomax**

Specifies whether to search subdirectories for usage files, and how many levels of subdirectories to search. *num* must be a non-negative integer.

If nomax is specified, all the subdirectories are searched. If 0 is specified, no subdirectories are searched.

Default: 0.

-qconfigfile=*file_path*

Specifies the user defined configuration file that you want to use.

For more information about how the utilization reporting tool uses the configuration file, see Chapter 7, "Understanding the utilization reporting tool," on page 27.

Note: If you specify this option multiple times, all specified instances are used.

-qsameuser=*user_account_info*

Specifies different user accounts that belong to the same compiler user. Use this option when a user accesses the compiler from more than one user ID or machine to avoid having that user reported as multiple users. Invocations of the compiler by these different accounts are counted as a single user instead of multiple different users.

user_account_info can be any combination of the following items:

- name(*user_name*)
- uid(*user_ID*)
- host(*host_name*)

There are two ways to pass these rules to the utilization reporting tool. You can supply specific lists of the *user_name*s, *user_ID*s or*host_name*s that are shared by the same user or you can use a more generic (=) syntax.

For example, to indicate that *user1* and *user2* are both user names belonging to the same person who uses the compiler on the *host1* machine, use the syntax in which you specify these user names and the host name explicitly:

```
-qsameuser="name(user1)host(host1) | name(user2)host(host1)"
```

or

```
-qsameuser="name(user1 user2)host(host1)"
```

Both of these examples use specific user names and host names to indicate accounts that belong to the same user, but they do so in slightly different ways. The first example uses a vertical bar to separate the different user accounts that belong to this user, while the second example uses a list of user names within the parentheses instead of repeating the same host information twice. They both convey the same account information, but the second example is more concise.

As an example of the more generic (=) syntax, you can indicate that all user accounts with the same user name and uid belong to the same user as follows:

```
-qsameuser="name(=)uid(=)"
```

With this option, you are not specifying specific user names or uids as you did in the previous example. User accounts that have the same user name and uid are considered as belonging to the same user, regardless of what the specific user names and uids are, and regardless of what the host name is. This establishes a general rule that applies to all accounts in your organization instead of specific ones.

The following rules apply when you specify this option:

- Each instance of the **-qsameuser** option must use either the list or generic (=) syntax. You cannot combine them in the same instance of the option but you can use multiple instances of the **-qsameuser** option to refine the report.
- The utilization reporting tool matches the user information based on the order that the **-qsameuser** option values are specified. Once it finds a match it stops matching the same user information against any subsequent options.

  The following examples illustrates the differences:

  – If you specify the **-qsameuser** option as follows:

    ```
    -qsameuser="name(user1)" -qsameuser="uid(=)"
    ```

Specifying the **-qsameuser** option in this order means that user accounts with the user name *user1* matches the first option and is not evaluated against the second option. User accounts *user1* and *user2* are not considered the same user even if they have the same *uid*.

– If you specify the **-qsameuser** option as follows:

```
-qsameuser="uid(=)" -qsameuser="name(user1)"
```

Specifying the **-qsameuser** option in this order means that user accounts with the same *uid* are always considered to be the same user, and in addition, any user accounts with a user name of *user1* should be considered belonging to the same user even if they do not match by *uid*.

Note: Specifying this option does not prevent user information from being listed in the usage report. For more information about concurrent users, see "Concurrent user considerations" on page 16.

-qadjusttime=*time_adjustments*

Adjusts the time that have been recorded in the usage files for the specified machines. *time_adjustments* is a list of entries with the format of *machine name* + | - *number of seconds*, separated by colons.

The following rules apply when you use this option:

- An entry of the form *machine name + number of seconds* causes the specified number of seconds to be added to the start and end times of any invocations recorded for the specified machine.
- An entry of the form *machine name - number of seconds* causes the specified number of seconds to be subtracted from the start and end times of any invocations recorded for the specified machine.

For example:

```
-qadjusttime="hostA+5:hostB-3"
```

Five seconds are added to the start and end times of the invocations on `hostA`, and three seconds are subtracted from the start and end times of the invocations on `hostB`.

Only use this option if the usage files contain utilization information from two or more machines, and time is not synchronized across those machines. The adjustments specified by this option compensate for the lack of synchronization

Notes:

- The specified adjustments are only used for the current run of the urt command. Specifying this option does not change the invocation information recorded in the usage files.
- Do not specify the same machine name more than once with this option.

-qusagefilemaxage=*number_of_days* | **nomax**

Prunes the usage files by removing all invocations older than the specified number of days.

Every usage file specified by the **-qusagefileloc** option is pruned. The usage report contains this information to indicate the number of records that have been pruned.

Default: -qusagefilemaxage=nomax, which means no pruning is performed.

-qusagefilemaxsize=*number_of_MB* **|** **nomax**

Prunes the usage files to keep them under the specified size. It prunes the files by removing the oldest invocations.

Every usage file specified by the **-qusagefileloc** option is pruned. The usage report contains this information to indicate the number of records that have been pruned.

Default: -qusagefilemaxsize=nomax, which means no pruning is performed.

-qtimesort=ascend | descend

Specifies the chronological order in which the usage report information is sorted.

- Specifying ascend means new information is listed after the older information.
- Specifying descend means the newest information is at the top of the report.

Default: -qtimesort=ascend.

Chapter 8. Generating usage reports

You can use the utilization reporting tool to generate compiler usage reports based on the usage information stored in the usage files.

To generate a report, use the command-line options or the urt configuration file to specify how you want a report to be generated. For more information about these options, see "Utilization reporting tool command-line options" on page 27.

Notes:

- You can set up a cron service to run the utilization reporting tool on a regular basis. If the usage files from which the tool generate reports need to be copied to the machine where the tool is running, you can also automate this copying task with cron.
- To reduce contention for read and write access to the usage files, do not run the tool or copy the usage files when the compiler is being used.

The generated report is displayed on the standard output. You can direct the output to a file if you want to keep the report.

After a usage report is generated, the utilization reporting tool uses the following exit codes to indicate the compliance status of your compiler license:

- Exit code =`"1"`.

  The utilization reporting tool has detected that the number of Concurrent User license entitlements specified in the **-qmaxconcurrentusers** entry in the utilization tracking configuration file has been exceeded. See the generated report for details and contact your IBM representative to purchase additional Concurrent User licenses.
- Exit code =`"0"`.

  The compiler utilization is within the number of Concurrent User license entitlements specified.

For more information about the urt command, see Chapter 7, "Understanding the utilization reporting tool," on page 27.

Understanding usage reports

You can use the report that the utilization report tool generates to analyze the compiler usage in your organization.

The report has a REPORT SUMMARY section that lists the following information:

1. The date and time when the report is generated.
2. The `.cuf` file or a list of all `.cuf` files used to generate the report.
3. The options that have been passed to the urt command, with default values for any unspecified options.
4. Possible messages categorized as ERROR, WARNING, or INFO. For detailed information about possible messages, see Chapter 10, "Diagnostic messages from utilization tracking and reporting," on page 39.

After the summary section, there is a REPORT DETAILS section for each compiler version. This section lists all of the compiler invocations recorded in the usage files.

The content of these sections varies depending on the report type that you have specified. For detailed information about the report types, see **-qreporttype**.

Here are the sample reports generated with the two different report types:

Sample 1: A sample report generated with **-qreporttype=detail**

```
REPORT SUMMARY
--------------

DATE: 12/18/09      TIME: 01:30:24

OPTIONS USED (* indicates that a default value was used):

reporttype=detail
maxsubdirs=0
configfile="/opt/ibmurt/1.1/config/ibmurt.cfg"
rptmaxrecords=nomax
*adjusttime=
usagefileloc="/home/testrun/ibmxlcompiler.cuf"
*sameuser=
timesort=ascend
usagefilemaxsize=nomax
usagefilemaxage=nomax


FILES USED:

/home/testrun/ibmxlcompiler.cuf

REPORT DETAILS
--------------

USAGE INFORMATION FOR PRODUCT: IBM XL C for AIX 10.1

Max. Concurrent Users Exceeded? : *** YES ***

Max. Concurrent Users Allowed: 1         Max. Concurrent Users Recorded: 5
Exempt Users:

Product invocations:

Start Time         End Time           User                Number of Concurrent Users
------------------ ------------------ -----------------   --------------------------
12/17/09 16:56:44  12/17/09 16:57:13  user1@host1.ibm.com 1
12/18/09 00:58:29  12/18/09 00:58:32  user2@host2.ibm.com 1
12/18/09 01:16:01  12/18/09 01:16:02  user3@host3.ibm.com 5 ( exceeds max. allowed)
12/18/09 01:16:02  12/18/09 01:16:26  user2@host2.ibm.com 5 ( exceeds max. allowed)
12/18/09 01:16:08  12/18/09 01:16:08  user3@host2.ibm.com 5 ( exceeds max. allowed)
12/18/09 01:16:12  12/18/09 01:16:12  user2@host1.ibm.com 5 ( exceeds max. allowed)
12/18/09 01:16:24  12/18/09 01:16:28  user1@host2.ibm.com 5 ( exceeds max. allowed)
12/18/09 01:26:11  12/18/09 01:27:46  user3@host3.ibm.com 2 ( exceeds max. allowed)
12/18/09 01:26:27  12/18/09 01:27:46  user1@host1.ibm.com 2 ( exceeds max. allowed)
12/18/09 01:29:59  12/18/09 01:30:00  user2@host1.ibm.com 1
12/18/09 01:30:00  12/18/09 01:30:00  user2@host2.ibm.com 3 ( exceeds max. allowed)
12/18/09 01:30:14  12/18/09 01:30:15  user3@host1.ibm.com 3 ( exceeds max. allowed)
12/18/09 01:30:14  12/18/09 01:30:14  user2@host2.ibm.com 3 ( exceeds max. allowed)
```

Sample 2: A sample report generated with **-qreporttype=maxconcurrent**

```
REPORT SUMMARY
--------------

DATE: 12/18/09      TIME: 01:32:53

OPTIONS USED (* indicates that a default value was used):
```

```
reporttype=maxconcurrent
maxsubdirs=0
configfile="/opt/ibmurt/1.1/config/ibmurt.cfg"
rptmaxrecords=nomax
*adjusttime=
usagefileloc="/home/testrun/ibmxlcompiler.cuf"
*sameuser=
timesort=ascend
usagefilemaxsize=nomax
usagefilemaxage=nomax


FILES USED:

/home/testrun/ibmxlcompiler.cuf

REPORT DETAILS
--------------

USAGE INFORMATION FOR PRODUCT: IBM XL C for AIX 10.1

Max. Concurrent Users Exceeded? : *** YES ***

Max. Concurrent Users Allowed: 1          Max. Concurrent Users Recorded: 5

Exempt Users:

Dates and times where usage exceeded the maximum allowed:

Date           Time        Number of Concurrent Users    Users
------------   ----        --------------------------    -----
12/18/09       01:16:01    5                             user3@host3.ibm.com
                                                         user2@host2.ibm.com
                                                         user3@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user1@host2.ibm.com
12/18/09       01:16:02    5                             user3@host3.ibm.com
                                                         user2@host2.ibm.com
                                                         user3@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user1@host2.ibm.com
12/18/09       01:16:08    5                             user3@host3.ibm.com
                                                         user2@host2.ibm.com
                                                         user3@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user1@host2.ibm.com
12/18/09       01:16:12    5                             user3@host3.ibm.com
                                                         user2@host2.ibm.com
                                                         user3@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user1@host2.ibm.com
12/18/09       01:16:24    5                             user3@host3.ibm.com
                                                         user2@host2.ibm.com
                                                         user3@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user1@host2.ibm.com
12/18/09       01:26:11    2                             user3@host3.ibm.com
                                                         user1@host1.ibm.com
12/18/09       01:26:27    2                             user3@host3.ibm.com
                                                         user1@host1.ibm.com
12/18/09       01:30:00    3                             user2@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user3@host1.ibm.com
12/18/09       01:30:14    3                             user2@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user3@host1.ibm.com
```

```
12/18/09       01:30:14    3                             user2@host2.ibm.com
                                                         user2@host1.ibm.com
                                                         user3@host1.ibm.com
```

Note: There are circumstances under which an end time might not be recorded. These might include:

- There was a major failure of the compiler, for example, power loss during a compilation.
- The invocation had not ended at the time when the report was generated, or at the time when the usage file was being copied.
- The permission to write to the usage file was revoked at some time before the end time of the invocation was recorded.

An invocation with no end time recorded is not included in the count of concurrent users.

Chapter 9. Pruning usage files

Usage files grow with each compiler invocation. You can prune the usage files with the utilization report tool.

When you generate a usage report, you can specify the following two options to optionally prune the usage files:

- **-qusagefilemaxage**: Removes the invocations older than the specified number of days. For example, to remove all entries in the usage files older than 30 days, use the following command:
  ```
  urt -qusagefilemaxage=30
  ```
- **-qusagefilemaxsize**: Removes the oldest invocations to keep the usage files under the specified size. For example, to remove the oldest invocations to keep the usage files under 30 MB, use the following command:
  ```
  urt -qusagefilemaxsize=30
  ```

When usage files are pruned, the usage report includes an information message that indicates the number of records that have been pruned. If you want to keep the generated report after the files are pruned, you can redirect the output to a file.

To control the size of the usage files, you can prune the usage files on a regular basis. You can create a cron job to do this automatically.

If you do not have the utilization reporting tool installed on each machine where the usage files are located, you have the following options:

- Export the file system from each machine where the usage files exist and mount it on the machine where the utilization reporting tool is installed. Then run the tool to prune the usage files on the mounted network file system.
- After copying the usage files to the machine where the utilization reporting tool is installed, delete the files and use new usage files to capture any subsequent compiler invocations. This approach might also speed up the report generation because the utilization reporting tool is not accessing the usage files remotely and it is not spending time pruning the usage files.

Pruning usage files might slow down the usage report generation process, especially when the number or the size of the usage files is large. If you do not want to prune the files every time you generate reports, you can set the values for the **-qusagefilemaxage** and **-qusagefilemaxsize** options as follows:

- If you generate the report daily, you can specify these two options with very high values so pruning does not occur. The default value **nomax** can be used in this case.
- You can set appropriate values for these two options and use a separate cron job to prune the usage files weekly.

Note: To reduce contention for read and write access to the usage files, do not run the utilization report tool or copy the usage files when the compiler is being used.

Chapter 10. Diagnostic messages from utilization tracking and reporting

The compiler generates diagnostic messages to indicate utilization tracking issues. These messages can help you to fix the associated problems.

For example:

```
Utilization tracking configuration file could not be read due to
insufficient permissions.
```

This message indicates that you need read access for utilization tracking configuration file.

When the utilization reporting tool is used to generate usage reports or prune usage files, it also generates diagnostic messages. For example:

```
Unrecognized option -qmaxsubdir.
```

This message indicates that you have specified a wrong option.

Note: Possible error, warning, or information messages are also included in the compiler usage report generated by the tool.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks and service marks