Sun StorageTek 5800 System SDK Developer's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–7558
November 2007

080317@19860

Contents

Preface

The *Sun StorageTek 5800 System SDK Developer's Guide* is written for programmers and application developers. This document, along with the *Sun StorageTek 5800 System Client API Reference Guide*, provides the information that you need in order to develop custom applications for the Sun StorageTek 5800 System.

How This Book Is Organized

- Chapter 1, "Introduction to the Sun StorageTek 5800 System Client SDK," provides an introduction to the Sun StorageTek 5800 System client Software Development Kit (SDK) and its components.
- Chapter 2, "Example Applications," provides information about the Java and C example applications that are provided with the Sun StorageTek 5800 SystemSDK.
- Chapter 3, "Sun StorageTek 5800 System Emulator," provides information on running the Sun StorageTek 5800 System emulator.

Related Books

- *Sun StorageTek 5800 System Regulatory and Safety Compliance Manual*, part number 819–3809
- *Sun StorageTek 5800 System Site Preparation Guide*, part number 820–1635
- *Sun StorageTek 5800 System Administration Guide*, part number 819–7555
- *Sun StorageTek 5800 System Client API Reference Guide*, part number 819–7557
- *Sun StorageTek 5800 System Release Notes*, part number 819–7559

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename.*

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Meaning	Example
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

`http://www.sun.com/hwdocs/feedback`

Please include the title and part number of your document with your feedback:

Sun StorageTek 5800 System SDK Developer's Guide, part number 819-7558

◆ ◆ ◆ CHAPTER 1

Introduction to the Sun StorageTek 5800 System Client SDK

This chapter provides an introduction to the Sun™ StorageTek™ 5800 System client Software Development Kit (SDK) and its components.

The following topics are discussed:

Overview of the SDK

The 5800 system client SDK enables you to work with data and metadata stored on a 5800 system. A 5800 system system emulator is included for the developer's convenience.

This document assumes that you have a basic understanding of how the 5800 system uses data and metadata. For more information, see "5800 System Semantics Overview" on page 13.

The SDK comes with an emulator that enables you to test client applications without having to connect to a 5800 system. For further documentation on the emulator, see Chapter 3, "Sun StorageTek 5800 System Emulator." Also refer to the *Sun StorageTek 5800 System Administration Guide* to better understand the type of server being emulated.

The SDK provides separate Application Programming Interfaces (APIs) for the Java™ and C languages. These APIs enable you to store and retrieve data and to store, retrieve, query, and delete metadata records.

For more information on the Java and C APIs, refer to the *Sun StorageTek 5800 System Client API Reference Guide*. Also, see Chapter 2, "Example Applications." For known bugs, see the *Sun StorageTek 5800 System Release Notes*.

SDK Terms of Use

The 5800 system client SDK is released to you under the following copyright notice:

"Copyright 2007 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms."

License terms are contained in the file `StorageTek_5800_SLA&Entitlement&PRN.txt` in the top-level directory of the installation. Licence terms for embedded software are in the file `LICENSE.txt` in the same directory.

SDK Components

The 5800 system client SDK distribution includes a Java distribution (Java examples and Java libraries), a C distribution (C examples and libraries), and the 5800 system emulator. All parts of the SDK support Solaris, Linux, and Windows environments.

The Java and C example programs serve both as examples of how to program a client application and as generally useful utilities in their own right.

The contents of the `.zip` archive include:

- `SDK_Overview_README.html`
- `doc/`
- `java/`
- `c/`
- `emulator/`
- `StorageTek_5800_SLA&Entitlement&PRN.txt`
- `LICENSE.txt`

Supported Operating Systems

This release supports the following operating systems:

- Red Hat Enterprise Linux 4 (32 bit)
- Red Hat Enterprise Linux 4 (64 bit)
- Solaris 9 SPARC
- Solaris 10 SPARC
- Solaris 10 x64 and x86

- Windows 2003 Server (32 bit)

Software Requirements

In order to use the 5800 system C API, the following software is required:

- All platforms
  - GNU make
  - Perl
- Solaris: Sun Studio
- Linux: `gcc`
- Windows 2003 server:
  - Microsoft Visual C++ Express Edition 2005
  - Microsoft Windows SDK
  - Cygwin (See `http://www.cygwin.com`)

SDK Installation Instructions

The 5800 system SDK is compressed in a file with a name similar to `StorageTek5800_SDK_1_1-20.zip`, where `1_1-20` refers to the version (`1_1`) and build number (`20`). This `.zip` file expands into a directory `StorageTek5800_SDK_1_1-20`, which contains the entire SDK.

5800 System Semantics Overview

This section provides an overview of 5800 system semantics.

Data and Metadata

The 5800 system stores two types of data: *arbitrary object data* and *structured metadata records*. Every metadata record is associated with exactly one data object. Every data object has at least one metadata record. A unique object identifier (OID) is returned when a metadata record is stored. The OID can later be used to retrieve the metadata record or its object data. In addition, metadata records can be retrieved by a query:

`OID` ↔ `Metadata Record` → `Object Data`

The two types of metadata are *system metadata* and *user metadata*. The names and types of system metadata are predefined and cannot be overridden by the user.

Metadata Schema

A user metadata record consists of a set of attributes. The set of possible attributes is defined by the schema. Only one schema exists for each 5800 system. For information on configuring a schema, see "Configuring Metadata and File System Views" in *Sun StorageTek 5800 System Administration Guide*.

The 5800 system emulator loads its schema from an XML file. The `metadata_merge_config` utility can be used to extend the default schema, appending the new entries to the original schema, which is then read by the emulator. For additional information, see Chapter 2, "Example Applications," and Chapter 3, "Sun StorageTek 5800 System Emulator."

Attributes

For information on the valid data types that you can use when specifying attributes in a schema, see "Configuring Metadata and File System Views" in *Sun StorageTek 5800 System Administration Guide*.

Namespace

A `namespace` is a container that holds a set of typed attribute names. Attribute names must be unique within a given `namespace`. Defining a namespace is a way to organize and group together a set of related attribute names for a given application. For more information, see "Configuring Metadata and File System Views" in *Sun StorageTek 5800 System Administration Guide*

5800 System Query Language

The 5800 system Java and C APIs both have a `Query` method that passes a query string to the 5800 system. Queries are presented to the name-value metadata cache.

For more information on the 5800 system query language, see Chapter 4, "Sun StorageTek 5800 System Query Language," in *Sun StorageTek 5800 System Client API Reference Guide*.

SDK Application Deployment

Java API

You must deploy `honeycomb-client.jar` with any application using the Java API. If you are using the SDK example applications, you must also deploy `honeycomb-sdk.jar`.

C API

The files in the `lib` directory must be deployed with any application using the C API. Different `lib` directories exist for different operating systems. You must use the correct `lib` directory for the OS under which the application is deployed. The `LD_LIBRARY_PATH` environment variable must be set accordingly for Unix systems. In the Windows environment, you must update the PATH environment variable.

◆ ◆ ◆ CHAPTER 2

Example Applications

This chapter provides information about the Java and C example applications that are provided with the 5800 system SDK.

The following topics are discussed:

Example Application Summary

Table 2–1 summarizes the Java and C example applications provided with the 5800 system SDK. All code examples are command-line applications.

For detailed information about the Java example applications, see "Java Example Applications" on page 18.

For detailed information about the C example applications, see "C Example Applications" on page 28.

TABLE 2–1 5800 system Client SDK Example Applications

Application	Java Version	C Version
`StoreFile` – Enables you to specify a file from the command line to store on a 5800 system server as data. You can also specify metadata to include with the file.	"`StoreFile`" on page 20	"`StoreFile`" on page 30

TABLE 2–1 5800 system Client SDK Example Applications *(Continued)*

Application	Java Version	C Version
`CheckIndexed` – Checks if the metadata for an object is present in the query engine, and inserts it if not.	"`CheckIndexed`" on page 21	"CheckIndexed" on page 31
`RetrieveData` – Enables you to retrieve data from the 5800 system server by Object ID. The data object is sent to standard output or to a file specified on the command line.	"`RetrieveData`" on page 22	"`RetrieveData`" on page 32
`RetrieveMetadata` – Enables you to retrieve a metadata record from the 5800 system server associated with the supplied OID. The metadata record is printed to standard output.	"`RetrieveMetadata`" on page 23	"`RetrieveMetadata`" on page 32
`AddMetadata` – Enables you to specify name-value pairs from the command line to store on a 5800 system server as a metadata record.	"`AddMetadata`" on page 23	"`AddMetadata`" on page 33
`DeleteRecord` – Deletes a record from the 5800 system server for an OID supplied on the command line.	"`DeleteRecord`" on page 25	"`DeleteRecord`" on page 34
`Query` – Queries a 5800 system server for specified metadata. The query string is provided on the command line. `Query` can print out the results as a list of name-value pairs or as a list of OIDs. You can specify the maximum number of results returned.	"`Query`" on page 25	"`Query`" on page 35
`RetrieveSchema` – Retrieves the schema from a 5800 system server, printing it to standard output.	"`RetrieveSchema`" on page 27	"`RetrieveSchema`" on page 36
`GetDate`– Gets the date.	"`GetDate`" on page 27	N/A

Java Example Applications

This section provides detailed information about the Java example applications provided with the 5800 system client SDK.

Examples Overview

Included with the 5800 system SDK are several command-line example applications that demonstrate the Java API. The example applications are located in the SDK in the `java/examples` directory. These example applications come complete with a build script.

Software Requirements

The Java example applications require at minimum the JDK™ 5.0 software.

Note – For Windows 2003, add `C:\Program Files\Java\`*jdk version*`\bin` to the `PATH` environment variable.

Running the Applications

UNIX (`.sh`) and Windows (`.bat`) scripts for running the example applications are provided in the `java/scripts/` directory. These scripts must be run from the `java/scripts/` directory. The scripts illustrate the `CLASSPATH` environment variable required. The `.jar` files are in the `java/lib/` directory.

Note – The usage messages printed by the applications omit the `CLASSPATH` environment variable for the sake of readability. If you are running the example application without using the provided scripts, then you must set the `CLASSPATH` environment variable manually.

Building the Java Example Applications

To build the Java example applications, go to the `java/examples/` directory and execute the `master-build.sh` script for Solaris and Linux environments or the `master-build.bat` script for Windows environments. These scripts build the examples and put them in the `honeycomb-sdk.jar` archive located in the `java/lib` directory.

Running a Java Example Application

Once you have built the Java example applications, go to the `java/scripts` directory and execute the `.sh` scripts for Solaris and Linux environments or `.bat` scripts for Windows environments:

Syntax: *script_name arguments*

See the scripts for details of how the applications are run.

About the Example Application Source Code

The Java example applications are all simple applications that follow the same basic structure. First, `Commandline.parse` is called to parse the argument list. Next, the appropriate method in the `NameValueObjectArchive` class is called to communicate with the 5800 system server. Finally, output is delivered back to either standard output, a file, or both. Refer to the comments in the sample code for further details.

Example Applications

The following sections describe the Java example applications that are included with the 5800 system client SDK:

`StoreFile`

Stores a file and associated metadata to a 5800 system server.

Synopsis

```
java StoreFile <IP | HOST> <FILE> [OPTIONS]
```

Description

Stores a file and its associated metadata record. If no `-m` options are specified, a metadata record without user content is generated. The `OID` of the metadata record is printed to `stdout`.

Note – `StoreFile` interprets the time in metadata arguments as local time zone unless the T..Z format indicating UTC is used. For example, 1952-10-27T00:30:29.999Z.

Options

```
-m <name>=<value>
```

Any number of `-m` options can be specified. Each `-m` option specifies a single (`name,value`) pair.

<name> should be specified in the format <namespace>.<attribute>. Use double quotes if <value> is a string containing spaces.

```
-h
```

Print this message.

Examples

```
java StoreFile 10.152.0.12 myFile
  java StoreFile server myFile.jpg \
       -m filesystem.mimetype="image/jpeg"
  java StoreFile server myFile \
       -m system.test.type_char="do re mi"
  java StoreFile server myFile \
       -m system.test.type_string="fa so la"
  java StoreFile server myFile \
       -m system.test.type_long=123
  java StoreFile server myFile \
       -m system.test.type_double=1.23
  java StoreFile server myFile \
       -m system.test.type_binary=0789abcdef
  java StoreFile server myFile \
       -m system.test.type_date=2010-10-20
  java StoreFile server myFile \
       -m system.test.type_time=23:30:29
  java StoreFile server myFile \
       -m system.test.type_timestamp="2010-10-20 23:30:29.999"
  java StoreFile server myFile \
       -m name1=value1 -m name2="value 2"
```

Source Code

`java/examples/StoreFile.java`

CheckIndexed

Ensure an object is queryable. Checks if the metadata for an object is present in the query engine, and inserts the metadata if it is not present.

Synopsis

```
java CheckIndexed <IP | HOST> <OID> [OPTIONS]
```

Description

Check with the 5800 systemserver to determine if the specified OID has become queryable. If not, attempt to make it queryable.

A short message about the supplied OID is printed to stdout:

```
Object OID was already queryable.
Object OID not yet queryable.
Object OID has now been made queryable.
```

Options

```
-h
```

Print this message.

Examples

```
java CheckIndexed server \
      0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
java CheckIndexed 10.152.0.12 \
      0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`java/examples/CheckIndex.java`

RetrieveData

Retrieves data from a 5800 system server.

Synopsis

```
java RetrieveData <IP | HOST> <OID> [FILE] [OPTIONS]
```

Description

Retrieves data from the 5800 system. The `OID` specifies what data to retrieve. Data is written to `FILE`, if specified, otherwise to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
java RetrieveData archivehost \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
    /archive/log.1
java RetrieveData 10.152.0.12 \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
    /archive/log.2
```

Source Code

`java/examples/RetrieveData.java`

RetrieveMetadata

Retrieves data (and metadata) from a specified 5800 system server.

Note – `RetrieveMetadata` always displays the time in the time zone of the shell running the program.

Synopsis

```
java RetrieveMetadata <IP | HOST> <OID> [OPTIONS]
```

Description

Retrieves a data record and metadata from the 5800 system server. The metadata record identified by the supplied `OID` is printed to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
java RetrieveMetadata 10.152.0.12 \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`java/examples/RetrieveMetadata.java`

AddMetadata

Adds a metadata record to an already stored object.

Synopsis

```
java AddMetadata <IP | HOST> <OID> [OPTIONS]
```

Description

Adds a new metadata record to an existing data object.

Options

```
-m <name>=<value>
```

Any number of `-m` options can be specified. Each option specifies a single (`name, value`) pair.

`<name>` should be specified in the format `<namespace>.<attribute>`. Use double quotes if `<value>` is a string containing spaces.

```
-h
```

Print this message.

Examples

```
java AddMetadata server \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m filesystem.mimetype="image/jpeg"
java AddMetadata server \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_char="do re mi"
java AddMetadata server
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_string="fa so la"
java AddMetadata server
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_long=123
java AddMetadata server
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_double=1.23
java AddMetadata
    server 0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_date=1952-10-27
java AddMetadata server \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_time=23:30:29
java AddMetadata server \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m system.test.type_timestamp="1952-10-27 23:30:29.999"
java AddMetadata server \
    0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000@ \
    -m name1=value1 -m name2="value 2"
```

Source Code

`java/examples/AddMetadata.java`

DeleteRecord

Deletes the Sun 5800 system metadata record associated with an OID.

Synopsis

```
java DeleteRecord <IP | Host> <OID> [OPTIONS]
```

Description

Deletes the record with the specified `OID`. If this record is the only record pointing to the data, the data will also be deleted.

Options

`-v`

Print deleted OID to `stdout`.

`-h`

Print this message.

Examples

```
java DeleteRecord server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`java/examples/DeleteRecord.java`

Query

Queries a 5800 system server for metadata records that match the query string passed on the command line.

Note – `Query` requires the T..Z UTC format. For example, 1952-10-27T00:30:29.999Z.

Synopsis

```
java Query <IP | HOST> <QUERY> [OPTIONS]
```

Description

Queries for metadata records. QUERY is of the form:

```
<name1>='<value1>' AND <name2>='<value2>' OR ...
```

See the examples below for formatting of various types of values. The OID and any specified fields of metadata records that match the query are printed to stdout.

<name> should be specified in the format <namespace>.<attribute>.

Note that names that are keywords need to be enclosed in escaped double quotes ("\"<name>\"='<value>'"). Refer to the list of keywords in Chapter 4, "Sun StorageTek 5800 System Query Language," in *Sun StorageTek 5800 System Client API Reference Guide*. Also note that some shells such as csh might not accept the escaped quotes because they are embedded in other quotes.

Options

```
-s <FIELD>
```

Specifies a field to be retrieved, much like an SQL select clause. To retrieve multiple fields, repeat this option. By default, the results are returned as a list of OIDs.

```
-n <number of results>
```

The maximum number of metadata records or OIDs that will be returned. The default is 1000.

```
-h
```

Print this message.

Examples

In the following examples, "first" is a keyword.

```
java Query server "book.author='King'"
java Query server "\"first\"='a'"
java Query 10.152.0.12 "mp3.artist='The Beatles' AND mp3.album='Abbey Road'"
java Query 10.152.0.12 "mp3.artist='The Beatles'" -s mp3.album -s mp3.title
java Query 10.152.0.12 system.test.type_char="'do re mi'"
java Query 10.152.0.12 system.test.type_string="'fa so la'"
java Query 10.152.0.12 system.test.type_long=123
java Query 10.152.0.12 system.test.type_double=1.23
java Query 10.152.0.12 system.test.type_binary="x'0789abcdef'"
java Query 10.152.0.12 system.test.type_date="'2010-10-20'"
java Query 10.152.0.12 system.test.type_time="'23:30:29'"
java Query 10.152.0.12 system.test.type_timestamp="{timestamp'2010-10-20T23:30:29.123Z'}"
```

Source Code

`java/examples/Query.java`

RetrieveSchema

Returns the schema defined on a 5800 system server to standard output.

Synopsis

```
java RetrieveSchema [OPTIONS] <IP | HOST>
```

Description

Retrieves the metadata schema from a 5800 system server, printing it to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
java RetrieveSchema archivehost
```

Source Code

`java/examples/RetrieveSchema.java`

GetDate

Gets the date.

Synopsis

```
java GetDate <IP | HOST> [OPTIONS]
```

Description

Gets the current date used to compute time setting and checking during store and delete operations.

Options

```
-h
```

Print this message.

Examples

```
java GetDate server
```

Source Code

`java/examples/GetDate.java`

C Example Applications

This section provides detailed information on the C example applications provided with the 5800 system client SDK.

Example Overview

Included with the 5800 system SDK are several command-line example applications that demonstrate the use of the C API. The example applications are located in the SDK under `c/examples`. Appropriate libraries are supplied for Solaris SPARC, Solaris x86, Red Hat Linux, and Windows.

Software Requirements

- In Solaris and Linux, set the `LD_LIBRARY_PATH` environment variable to *SDK directory name*`/c/`*OS*`/lib` where *SDK directory name* is the directory into which you unzipped the 5800 system SDK and *OS* is either Solaris or Linux. For example:

  ```
  StorageTek500_SDK_1_1_82/c/Solaris/lib
  ```

- In Microsoft Windows 2003:
  - The example programs will run if they are run as installed in the default build directory. However, if you have moved them elsewhere, the location of the library files (DLLs) must be added to the PATH environment variable as follows:

    ```
    C:\StorageTek500_SDK_1_1_82\c\Win32\lib
    ```

    To edit the windows `PATH` environment variable:

    1. Click the Start button.
    2. Right-click `My Computer` and select properties to launch System Properties.
    3. Click the Advanced tab.
    4. Click Environment Variables.
    5. Under System Variables, scroll down and click Path.
    6. Click Edit to launch Edit System Variable.

7. Add the full path name of the SDK `lib` directory to the `PATH` environment variable. Make certain each path name is separated by a semicolon (`;`).
8. Click OK to close each window.

– Add `C:\Program Files\Java\`*jdk_version*`\bin` to the `PATH` environment variable where *jdk_version* is the version, for example, `jdk5.0`.

To edit the windows `PATH` environment variable:

1. Click the Start button.
2. Right-click `My Computer` and select properties to launch System Properties.
3. Click the Advanced tab.
4. Click Environment Variables.
5. Under System Variables, scroll down and click Path.
6. Click Edit to launch Edit System Variable.
7. Add the Java path name to the `PATH` environment variable. Make certain each path name is separated by a semicolon (`;`).
8. Click OK to close each window.

Also see "Software Requirements" on page 13.

Building the C Example Applications

To build the C example applications, go to the `c/examples` directory and run `make`. This will build the example applications and put them in the `c/examples/`*OS*`/build` directory.

Each C example application can be built separately by running `make` *program_name*.

Running a C Example Application

Once you have built the C example applications, go to the `c/examples/<OS>/build` directory and execute the binary file with the appropriate command line. The examples depend on `libhoneycomb.so`.

About the C Example Application Source Code

First, the function `parseCommandline` is called to parse the command line and store the information in a `struct` called `Commandline`. Next, any files that contain data to be sent to the 5800 system server are opened. The appropriate 5800 system C API is then called. Finally, output is delivered back to either standard output, a file, or both. Refer to the comments in the sample code for further details.

Example Applications

The following C example applications are included with the 5800 system client SDK:

StoreFile

Stores a file and associated metadata on a 5800 system server.

Synopsis

```
StoreFile <IP | HOST> <FILE> [OPTIONS]
```

Description

Stores a file and its associated metadata record. If no `-m` options are specified, a metadata record without user content is generated. The OID of the metadata record is printed to `stdout`.

Options

`-m <name>=<value>`

Any number of `-m` options can be specified. Each option specifies a single (`name,value`) pair.

`<name>` should be specified in the format `<namespace>.<attribute>`. Use double quotes if `<value>` is a string containing spaces.

`-h`

Print this message.

Examples

```
StoreFile server /var/log/messages
StoreFile server ~/journal
StoreFile server myfile.jpg -m filesystem.mimetype="image/jpeg"
StoreFile 10.152.0.12 myfile -m system.test.type_char="do re mi"
StoreFile 10.152.0.12 myfile -m system.test.type_string="fa so la"
```

```
StoreFile 10.152.0.12 myfile -m system.test.type_long=123
StoreFile 10.152.0.12 myfile -m system.test.type_double=1.23
StoreFile 10.152.0.12 myfile -m system.test.type_binary=0789abcdef
StoreFile 10.152.0.12 myfile -m system.test.type_date=2010-10-20
StoreFile 10.152.0.12 myfile -m system.test.type_time=23:30:29
StoreFile 10.152.0.12 myfile \
     -m system.test.type_timestamp="2010-10-20T23:30:29.999"
StoreFile 10.152.0.12 myfile -m name1=value1 -m name2="value 2"
```

Source Code

`c/examples/StoreFile.c`

CheckIndexed

Ensure an object is queryable. Checks if the metadata for an object is present in the query engine, and inserts the metadata if it is not present.

Synopsis

```
CheckIndexed <IP | HOST> <OID> [OPTIONS]
```

Description

Check with the 5800 systemserver to determine if the specified OID has become queryable. If not, attempt to make it queryable.

A short message about the supplied OID is printed to stdout:

```
Object OID was already queryable.
Object OID not yet queryable.
Object OID has now been made queryable.
```

Options

```
-h
```

Print this message.

Examples

```
CheckIndexed archivehost \
       0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
CheckIndexed 10.152.0.12 \
       0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`c/examples/CheckIndexed.c`

RetrieveData

Retrieves a data object from a 5800 system server.

Synopsis

```
RetrieveData <IP | HOST> <OID> <FILE> [OPTIONS]
```

Description

Retrieves data from the 5800 system. The `OID` specifies what data to retrieve. Data is written to `FILE`, if specified, otherwise to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
RetrieveData storagetek \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     /archive/log.1
```

Source Code

`c/examples/RetrieveData.c`

RetrieveMetadata

Retrieves a metadata record from a specified 5800 system server.

Synopsis

```
RetrieveMetadata <IP | HOST> <OID> [OPTIONS]
```

Description

Retrieves metadata from the 5800 system. The `OID` specifies what data to retrieve. Metadata is printed to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
RetrieveMetadata archivehost \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
RetrieveMetadata 10.152.0.12 \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`c/examples/RetrieveMetadata.c`

AddMetadata

Adds a metadata record to existing data.

Synopsis

```
AddMetadata <IP | HOST> <OID> [OPTIONS]
```

Description

Adds a new metadata record to an existing data object.

Options

```
-m <name>=<value>
```

Any number of `-m` options can be specified. Each option specifies a single (`name, value`) pair.

`<name>` should be specified in the format `<namespace>.<attribute>`. Use double quotes if `<value>` is a string containing spaces.

`-h`

Print this message.

Examples

```
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m filesystem.mimetype="image/jpeg"
```

```
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_char="do re mi"
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_string="fa so la"
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_long=123
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_double=1.23
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_date=1992-10-27
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_time=23:30:29
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m system.test.type_timestamp="1992-10-27T23:30:29"
AddMetadata server \
     0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000 \
     -m name1=value1 -m name2="value 2"
```

Source Code

`c/examples/AddMetadata.c`

DeleteRecord

Deletes a record associated with an `OID`.

Synopsis

```
DeleteRecord <IP | HOST> <OID> [OPTIONS]
```

Description

Deletes a record associated with an `OID`. The `OID` specifies which record to delete. The record consists of all metadata associated with the `OID`, or the data if it is a data `OID`. The `OID` itself becomes inaccessible. If this OID is the last `OID` associated with the data, the data is also deleted.

Options

`-v`

Print deleted OID to `stdout`.

`-h`

Print this message.

Examples

```
DeleteRecord server 0200004f75ee01094cc13e11dbbad000e08159832d000024d40200000000
```

Source Code

`c/examples/DeleteRecord.c`

Query

Queries a 5800 system server for metadata records that match the query string passed on the command line.

Note – `Query` requires the T..Z UTC format. For example, 1952-10-27T00:30:29.999Z.

Synopsis

```
Query <IP | HOST> <QUERY> [OPTIONS]
```

Description

Queries for metadata records. `QUERY` is of the form:

`<name1>='<value1>' AND <name2>='<value2>' OR ...`

See the examples below for formatting of various types of values. The `OID` and any specified fields of metadata records that match the query are printed to `stdout`.

`<name>` should be specified in the format `<namespace>.<attribute>`.

Note that names that are keywords need to be enclosed in escaped double quotes ("`\"<name>\"='<value>'`"). Refer to the list of keywords in Chapter 4, "Sun StorageTek 5800 System Query Language," in *Sun StorageTek 5800 System Client API Reference Guide*. Also note that some shells such as `csh` may not accept the escaped quotes because they are embedded in other quotes.

Options

`-s <FIELD>`

Print out results as metadata name-value records. Use as many `-s` switches as needed to define all fields that will be printed to `stdout`.

`-n <number of results>`

The maximum number of metadata records or `OID`s that will be returned. The default is 1000.

`-h`

Print this message.

Examples

In the following examples, “first” is a keyword.

```
Query archivehost "book.author='King'"
Query archivehost "\"first\"='a'"
Query archivehost system.test.type_char="'do re mi'"
Query archivehost system.test.type_string="'fa so la'"
Query archivehost system.test.type_long=123
Query archivehost system.test.type_double=1.23
Query archivehost system.test.type_binary="x'0789abcdef'"
Query archivehost system.test.type_date="'2010-10-20'"
Query archivehost system.test.type_time="'22:10:29'"
Query archivehost system.test.type_timestamp="{timestamp'2010-10-20T23:30:29.123Z'}"
Query 10.152.0.12 "mp3.artist='The Beatles' AND mp3.album='Abbey Road'"
Query 10.152.0.12 "mp3.artist='The Beatles'" -s mp3.album -s mp3.title
Query 10.152.0.12 "system.test.type_timestamp={timestamp '1952-10-27T08:30:29.999Z'}"
```

Source Code

`c/examples/Query.c`

RetrieveSchema

Prints metadata attributes to `stdout`.

Synopsis

```
RetrieveSchema <IP | HOST> [OPTIONS]
```

Description

Retrieves the schema from a 5800 system server, printing it to `stdout`.

Options

```
-h
```

Print this message.

Examples

```
RetrieveSchema archivehost
```

Source Code

`c/examples/RetrieveSchema.c`

◆ ◆ ◆ CHAPTER 3

Sun StorageTek 5800 System Emulator

This chapter provides information on running the 5800 system emulator.

The following topics are discussed:

Introduction to the Emulator

The 5800 system emulator mimics the behavior of a 5800 system server. This feature enables you to test software being developed for a 5800 system without having to access a 5800 system server. The emulator stores all data and metadata on the local hard drive. This data is by default stored in the `emulator/var` directory.

Software Requirements

The 5800 system emulator requires at minimum the JDK 1.5 software.

Emulator Startup

The 5800 system emulator can run on Solaris, Linux, or Windows environments. To start the emulator, go to the `emulator/bin` directory and execute the `start.sh` file for Solaris and Linux environments or the `start.bat` file for the Windows environment. The emulator will be listening on port 8080 of the machine you start it on.

See "Emulator Configuration File" on page 41 for directions on changing the port number.

Emulator Shutdown

Connect to port 8080 of the machine running the 5800 system emulator with your web browser (`http://localhost:8080` if you are on your local system). Click on `HttpContext [/admin]`, then `Click here` to shutdown the emulator.

Schema Modification

To add a custom schema to the 5800 system, create an XML file detailing your custom schema. Refer to the XML files in the `config` directory for examples of a custom schema. Also, see "Configuring Metadata and File System View" in *Sun StorageTek 5800 System Administration Guide*.

Once your XML schema file is complete, stop the emulator and run the `metadata_merge_config.sh` script (`metadata_merge_config.bat` on Windows) located in the `bin` directory. This script takes one command-line argument, which is the full path to your XML schema file. The emulator should be stopped while updating the schema.

To use a new schema in the emulator you should first do a manual wipe of the emulator contents.

▼ To Manually Clear the Emulator Contents

1. **Shut down the emulator if it is currently running.**
2. **Remove the emulator `var` directory and all of its contents.**

   For example:

   `rm -fr` *SDK directory name*`/emulator/var`

   where *SDK directory name* is the directory into which you unzipped the 5800 system SDK.
3. **Run the `metadata_merge_config` program to activate the new schema.**

4 **Restart the emulator.**

Emulator Event Log

The emulator event log is automatically stored under `/emulator/logs/emulator.log`. The event log contains all the transactions that the emulator has processed, as well as startup and shutdown information. This log can be very helpful in debugging your 5800 system client applications.

Emulator Configuration File

The emulator configuration file is located at `/emulator/config/emulator.config`. This file contains settings for configuring the emulator. To configure the emulator to listen on a different port, add the line:

`honeycomb.protocol.port =` *port_number*

to the `emulator.config` file, where *port_number* is the port the emulator will listen on. For further information on the configuration file, refer to the *Sun StorageTek 5800 Administrator's Guide*.

Index

W