SYBASE®

Examples Guide

Sybase CEP Option R4

DOCUMENT ID: DC01026-01-0400-01
LAST REVISED: February 2010

Sybase, Inc., One Sybase Drive, Dublin, CA 94568

Contents

Preface

The *Examples Guide* describes most of the examples that ship with the current version of Sybase® CEP and contains use cases of Sybase CEP applications. The examples demonstrate how Sybase CEP can be used in a variety of domains, such as network security, process monitoring, RFID, and finance.

If you are not already familiar with Sybase CEP, it is recommended that you first read the following documents:

- *Sybase CEP Technology Overview*
- *Sybase CEP Frequently Asked Questions (Basic)*

and complete the *Sybase CEP Getting Started* tutorial.

For Sybase CEP Server and Sybase CEP Studio installation and configuration instructions, please see the *Sybase CEP Installation Guide*.

For help with using the Sybase CEP Studio interface, please see the *Sybase CEP Studio Guide*.

For more advanced Sybase CEP Engine and CCL programming concepts, please see the :

- *Sybase CEP Programmer's Guide*.
- *Sybase CEP CCL Reference*.
- *Sybase CEP Frequently Asked Questions (Advanced)*.
- *Sybase CEP Integration Guide*.

Working with Examples

Sybase CEP examples are located in the `examples` subdirectory of the Adapters Base Folder designated during installation.

By default, the Adapters Base Folder is SybaseC8Repository. On Microsoft Windows, the default location is:

```
C:\Documents and Settings\My Documents\
user-name\
SybaseC8Repository\3.0
```

On UNIX-like operating systems, the default location is in the home directory of the user under which Sybase CEP was installed.

Within the `examples` directory, the various Sybase CEP examples are organized by type (such as Finance or NetworkAndSecurity). Each example directory may contain subdirectories:

- The data subdirectory contains .csv files that are used to simulate data input received by the example Sybase CEP project.
- The modules subdirectory contains additional (nested) query modules, when the example uses more than one module.
- The schema subdirectory contains schema .ccs files used by the streams and named windows in the example.
- The src subdirectory contains source files for any examples that use one of the SDKs.

To run an example, load its project .ccp file into Sybase CEP Studio and run the project. See the *Sybase CEP Studio Guide* for more instructions on loading and running projects.

Reinstalling Examples

If you have modified examples and want to reinstall the original clean version, delete the `examples` directory and restart Sybase CEP Studio. The original examples are automatically recreated.

Note that, even if you uninstall and reinstall Sybase CEP, your examples directory survives if you have made any changes to it. To get fresh examples, you must delete the examples directory. When you install a new version of Sybase CEP Engine, it will install a new set of examples into a new `examples` directory under the new version number.

Feature Examples

The FeatureExamples directory contains example code covering most of the features of Sybase CEP Engine. The directory contains both CCL and SDK examples, and each source code file includes embedded comments to explain the logic of the example.

To examine and run the CCL examples, load them into Sybase CEP Studio (see the *Sybase CEP Studio Guide* for more information) or Eclipse (see the *Sybase CEP Eclipse Plug-In Guide* for more information).

The examples in FeatureExamples are organized into the following subdirectories:

- Adapters: These examples demonstrate how to use adapters that Sybase CEP provides as well as how to use the provided SDKs to build in-process (C) and out-of-process (C, Java 1.5, .NET3, Perl, and Python) adapter. Use the make file found directly under FeatureExamples to compile the source code for the SDK examples. The file readme.txt under InProcess and OutOfProcess contains complete instructions.
- Basic: These examples demonstrate basic Sybase CEP Engine features with CCL.
- DatabasesAndRPC: These examples show how to interact with databases and remote procedures. Some external setup needs to be performed in order for these examples to work. Complete setup instructions are listed in the comments in the specific CCL modules.
- FunctionsAndOperators: These examples demonstrate the Sybase CEP predefined functions and operators, as well as showing how to define your own function with the C/C++ SDK. Use the make file found directly under FeatureExamples to compile the source code for the SDK examples. The file readme.txt under UserDefinedFunctions contains complete instructions.
- Patterns: These examples illustrate event pattern matching with CCL.
- XML: These examples demonstrate XML parsing with Sybase CEP Engine and are explained in detail in the technical article *Sybase CEP XML Cookbook*.
- Windows: These examples demonstrate different types of windows, including public windows, and a variety of ways that the contents of windows can be used and manipulated.

Games Examples

The `games` subdirectory of the `examples` directory contains games examples, including Asteroids and Breakout.

Asteroids

A description of the games example that runs similar to the Asteroids arcade game.

The Asteroids project contains the following submodules:

- The InitClient module initializes the client by creating macros.
- The MainLoop module manages the main game loop, and contains the game logic.
- The ExtendedMsgsProcessor helper module converts high-level commands into low level c8_sdl_client commands.

Breakout

A description of the games example that runs similar to the Breakout game.

AI mode may be toggled on and off by pressing the "A" key in the game screen. In AI mode, the paddle is controlled automatically by the CCL statements.

Getting Started

The `GettingStarted` subdirectory of the `examples` directory contains files used by the *Sybase CEP Getting Started* tutorial.

For more information, please see the *Sybase CEP Getting Started* tutorial.

Miscellaneous Examples

The `miscellaneous` subdirectory of the `examples` directory contains several examples, described in the sections below.

Factorial

An example that computes the factorial for the number in the input stream.

Fibonacci

An example that computes the Fibonacci series from *M* to *N*, where *M* and *N* are configurable.

To request the series, a row is sent to stream InputRequests. In response, the Fibonacci series segment from Lower to Upper is generated in stream FilteredResultsStream.

Pythagorean

An example that searches for Pythagorean triples in randomly generated data.

Sorting

An example that demonstrates how to sort rows in a stream.

The stream OrdersIn has rows associated with purchase orders coming in random order, unsorted by StepID. After the rows are processed by the queries in this example, they are published to stream OrdersOut sorted by StepID.

A Window NextStep is used to remember the next StepID for which the query is looking for, for every order. An incoming row triggers a new row if its StepID is equal to NextStep.StepID for this order. This, in turn, can trigger the next row, the row after that, and so on.

A KEEP ALL window is used to enable the processing of early step IDs that arrive late in the query's execution. In a real-world application, a KEEP ALL window would probably be impractical (it would use up too many resources). In such an application the window definition would have a time or row limit, for example: KEEP 1000 ROWS.

UniversalTuringMachine

An example of an implementation of the Universal Turing Machine (UTM).

Network and Security Examples

The `NetworkAndSecurity` subdirectory of the `examples` directory contains several examples.

GangliaAlerts

An example that consists of two submodules that send alerts when the percentage of free disk space is smaller than 10%, or when the Metric column of the InData stream contains the string of "part_max_used" and the corresponding value in the Value column exceeds 90.00.

RestaurantFraudDetection

An example that shows how to track unusual events within a large-scale restaurant for real-time fraud detection.

This example shows how to track unusual events within a large-scale restaurant for real-time fraud detection. It contains three submodules:

The AveragesAlerts submodule detects statistical anomalies in restaurant events. The queries in this module analyze certain ratios for each cashier, relative to the number of started checks. These include the number of no sales, voided checks, split checks, transfer checks, reopen checks, and recalled checks. When the average of any of these for a given cashier, or for all cashiers exceeds the average by more than one standard deviation, an alert is issued.

The ManualPatternsAlerts submodule detects the occurrence of the following pattern by using windows and joins. An alert is issued if the pattern occurs:

1. Start check [event 3000].
2. Ring item(s) [amount > 0 when the check is canceled].
3. Cancel Check (requires manager override) [event 3999].
4. No Sale [event 5101].
5. No Receipt Printed (absence of event 5110).

The C8PatternsAlerts submodule monitors for the same pattern using CCL's pattern (**MATCHING** clause) syntax.

SecurityEventsCorrelation

An example project that contains submodules that implement two methods for security events correlation. The two data input streams for the project, InVirusAlerts and InIDSAlerts,

simulate alerts coming from a virus detection system and an intrusion detection system, respectively.

The SecurityPatterns submodule uses the CCL pattern (MATCHING clause) syntax to detect a series of events arriving in the InVirusAlerts and InIDSAlerts streams. An alert is issued if the pattern is detected.

The SecuritySimpleJoin submodule uses a simple join to detect possible intrusions into the system. The query looks for common SourceIP addresses from which attacks originate. This is accomplished by using the following query structure:

```
INSERT INTO OutRepeatAttackAlerts
SELECT I1.SourceIP AS SourceIP,
      I1.AttackKind AS AttackKind,
      V1.Virus AS Virus,
FROM   InIDSAlerts KEEP 30 SECONDS AS I1,
      InVirusAlerts KEEP 30 SECONDS AS V1
WHERE  I1.SourceIP=V1.SourceIP
FROM   InIDSAlerts As I1 KEEP 30 SECONDS,
      InVirusAlerts As V1 KEEP 30 SECONDS
WHERE  I1.SourceIP=V1.SourceIP ;
```

SecurityLogAnalysis

SecurityLogAnalysis is a network security demo that models a common way to look at authentication alerts in three log files (`syslog`, `ftp` authentication, and `/var/log/secure` log). This example demonstrates various kinds of alerts.

- **Alert 1:** five (5) failures that have:
  - The same source.
  - The same destination.
  - The same user name.

  Generates a low priority alert. Somebody may have forgotten their password, or is trying to guess a password.
- **Alert 2:** seven (7) or more failures that have:
  - The same source.
  - Different destinations.
  - The same user name.

  Generates a low priority alert. Somebody may have forgotten the same password for multiple systems, or is trying to guess a password.
- **Alert 3:** four (4) or more failures that have:
  - The same source.
  - The same, or different destinations.
  - Different user names.

Generates a medium priority alert. Somebody is clearly trying to guess the user name and password.

- **Alert 4:** three (3) or more failures that have:
  - The same source.
  - The same, or different destinations.
  - Different user names.

  Followed by one (1) event that has:
  - The same source as before.
  - The same destination as before.
  - One of the user names tried before.

  Generates a high priority alert. Somebody tried to guess the name and password, and has succeeded.

SlowScanDetection

An example that detects slow scan activity by identifying patterns of syslog entries.

1. An IDS warning that a machine has been attacked by the worm.
2. An excessive number of NETBIOS packets appears within the time threshold.

The Module is parameterized:

BackDoorWatchDuration	Number of hours after an attack during which the attacked machine should be monitored.
AlertThresholdCount	Number of NETBIOS packets that are considered to be "excessive".
AlertThresholdDuration	Number of microseconds during which AlertThresholdCount must be exceeded in order for NETBIOS packets to be considered excessive.
AlertThresholdCountPlus1	Used in creating a window in one of the queries in the module.

Production environments generate a large number of syslog entries for many machines. At any time, a relatively small number of machines are attacked by viruses, and are therefore under suspicion. The module is structured into several queries in order to minimize the load on the Sybase CEP Engine.

The first query creates a stream containing only the attack entries from the IDS. This prefiltering is used to make the second query's WatchWindow size smaller, as WatchWindow will now include information about only those machines that have actually been attacked, instead of storing data about all machines.

The second query filters in only NETBIOS activity from machines that have been attacked within $BackDoorWatchDuration hours. This minimizes the window size in the third query.

The third query creates alerts when the $AlertThresholdCount suspicious rows have occurred within $AlertThresholdDuration.

The fourth query formats the alert.

Process Monitoring Examples

The `ProcessMonitoring` subdirectory of the `examples` directory contains several examples.

BPELTranslator

An example that provides the files for simulating a BPEL to CCL conversion.

For instructions on how to use the files, please read the `readme.txt` file in the `BPELTranslator` directory.

ExpensesApproval

An example that simulates an automatic system for approving expense reports that match given criteria.

This project contains two submodules: the DataGen submodule generates random source data and the ApproveExpenses submodule performs the approval.

FiniteStateMachine

An example that is a structurally simple implementation of a Deterministic Finite State Machine. The example uses a project with a single main module.

For this example, the state transition table is assumed to exist in an Oracle, Ingres, or another database. The `fsmDb.sql` file in the `data` subdirectory contains the SQL necessary to set up the Oracle example. The SQL table has the following form:

```
create table dfsm_state_transitions (
 from_state             varchar2(30) not null,
 to_state               varchar2(30) not null,
 event                  varchar2(30) not null,
 action                 varchar2(30) not null)
```

The database setup is further described in the `readme.txt` file in the data subdirectory.

The data describes the possible paths of RFID-enabled pallets through doors in a three-room building. The incoming data has no timestamp. Instead of using a timestamp column from the data source, the input adapter specifies a Rate to regulate the input rate and uses the current time to assign timestamps to incoming rows.

The StateEventStream in this example receives its data from a `.csv` file, which contains an ID column, to differentiate the multiple finite state machine instantiations, as well as a FromState and an Event column. The FromState and Event data is used in conjunction with a CCL database subquery, which obtains the to_state and action values from the relational database. The retrieved information is passed on to the CurrentStateStream stream.

The SQL queries used in this example notes the absence of an entry in the state transition table by the union with a not-exists subquery. This is used for subsequent filtering and error handling. If there is no entry in the state transition table, the row is fed to the ErrorAction stream.

Valid rows are passed into a named window called States that keeps the last two rows for each finite state machine instantiation.

The TransitionStatus stream receives both valid and invalid state transition data from States. The transition is considered valid if the previous state value of the current row is the same as the previous row's state value (or if the previous row does not exist, because only the first row has been processed). Otherwise data has been dropped or extraneously added and the entire process is suspect. The States window maintains a view on the last two rows to identify such loss of data.

The TransitionStatus.IsValid field is set to either TRUE or FALSE when the data is validated and the row is published to the StandardAction or ErrorAction stream, depending on the value in the TransitionStatus stream.

ProcessTracking

An example that shows how certain processes can be tracked in real-time. The data for this example simulates a large number of processes going through several steps, using three scenarios, some of which include loops.

The three simulated process flows are:

- **Flow 1** (Scenario 1):
  1. Mold.
  2. Trim.
  3. Package.
- **Flow 2A** (Scenario 2, with a loop on steps 2 and 3 repeated):
  1. Premix.
  2. Dye injection.
  3. Dye curing.
  4. Dye injection.
  5. Dye curing.
  6. Varnish.
  7. Polish.

- **Flow 2B** (Scenario 2, with step 4 repeated):
  1. Premix.
  2. Dye injection.
  3. Dye curing.
  4. Varnish.
  5. Varnish.
  6. Polish.
- **Flow 3** (Scenario 3):
  1. Premix.
  2. Dye mix-in.
  3. Dye curing.
  4. Brush.
  5. Polish.

The example then computes:

- The number of completed steps.
- The number of uncompleted steps.
- The average duration of each step.
- The average number of processes that have completed in a given scenario.
- Several other metrics.

Finally, this example shows how to compute complex metrics based on these simple metrics.

RFID and Sensor Network Examples

The `RfidAndSensorNetworks` subdirectory of the `examples` directory contains several examples.

PalletCasesMapping

An example that accepts a stream of RFID readings representing cases and pallets on a conveyor belt, and outputs XML nodes describing the contents of each pallet.

It is assumed that there may be any number of redundant readings for a given case or pallet, and that the closest pallet to a given case is considered to be the one that contains the case. RFID tags of pallets always begin with the characters "RSP" (Regular Shipping Pallet), "OSP" (Old Shipping Pallet), or "CSP" (Cleveland Shipping Pallet); all other prefixes are assumed to represent cases.

ParentChildrenTracking

A version of the parent-children tracking example that takes all its input data from `.csv` files, simulating live RFID data, and publishes the alerts into output stream OutStreamAlerts.

The raw RFID data arrive in input stream InStreamRfid. This stream contains only a timestamp, ReaderID, and PersonID. A second input stream, InStreamReaders, carries information about the RFID reader coordinates, and a third stream, InStreamPersons, contains family and person IDs of the people wearing the RFID sensors, as well as a BOOLEAN column IsChild identifying if the family member is a child. Note that the information in stream InStreamPersons is rather static, and may be contained in a database in a real-world application. However, for this example, person information is represented as a data stream.

The local stream StreamRfidFull joins the information coming in on the three input streams. This stream is then split into two streams: StreamChildren and StreamParents, according to the values in the BOOLEAN column IsChild.

Finally, a join is executed between StreamChildren and StreamParents. The query checks whether a child and a parent from the same family (as defined by FamilyID) are more than a certain distance away from each other. If they are, an alert is published into stream OutStreamAlerts.

ParentChildrenTrackingWithDatabase

A variation of the parent-children tracking example that takes static information from database tables person_data and reader_data. The RFID readings come from a `.csv` file, which simulates live RFID readings.

Because this example illustrates how to access a database, additional steps to set up and populate the database must be completed before running the example. Please see the `readme.txt` file in the `data` subdirectory for more information.

PowerGridMonitoring

An example that monitors the status of 30 switches.

Every switch has four to six sensors associated with it. Every sensor is either a voltage or current sensor. Every switch reports once per second on its state, which may be "on" (closed) or "off" (open); this information arrives in input stream StreamSwitchStates. Every sensor reports its value once per second; these rows arrive in streams StreamVoltage or CurrentAlerts, whichever is appropriate.

The queries in this example generate two kinds of alerts: "swing alerts" and "switch alerts". Swing alerts are issued when the value reported by a single sensor changes by at least a threshold amount within a specified interval. Switch alerts are generated when a switch is thrown, but the voltage or current reported by at least one of the switch's associated sensors does *not* change significantly and immediately.

WarehouseMonitoring

An example that shows how cases and pallets can be tracked in a warehouse, using RFIDs.

The main module of this project contains a local stream StreamFullReadings, which joins the data from the project's two input streams. A number of intermediate and alert streams, based on this stream, are then created. Some of the alerts include notification that a pallet is not unpacked after a certain time interval, a case did not make it to a shelf after a certain time interval, or that a pallet was not shipped after it was packed, and so on.

The DoorAlertsSlow and DoorAlertsVerySlow submodules contain queries that send an alert if traffic through a certain door is too slow. "Too slow" is defined as being more than a certain fraction of the standard deviation less than the average traffic through all doors. This fraction is the parameter called $StdDevInterval. In the DoorAlertsSlow module, this parameter is set to 0.5; in the DoorAlertsVerySlow module, it is set to 1.0.

The CasesNotShipped8sec and CasesNotShipped15sec modules contain queries that issue an alert if an item is not shipped for longer than a certain period of time, in this case 8 or 15 seconds. Remember that this is just a simulation. In a real-world application, this number would be higher.

Web Examples

The `Web` subdirectory of the `examples` directory contains several examples.

AppacheLogAnalysis

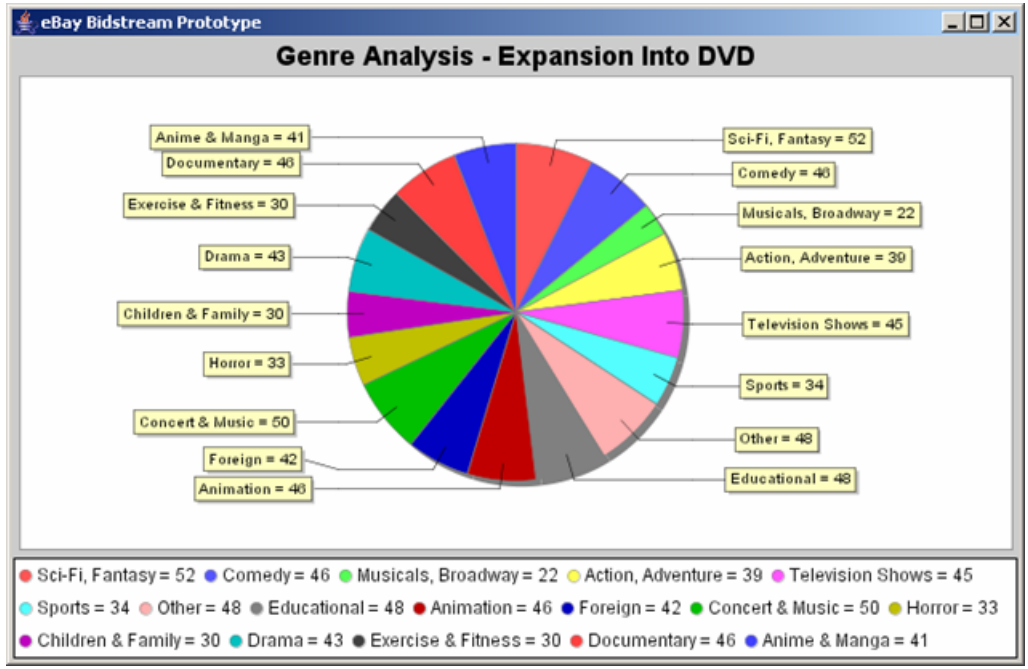
This example gathers click information from Apache logs, calculates "topics" popularity ratings, and then issues alerts for the most popular topics.

eBayBidMonitoring

An example that simulates a stream of eBay movie bid data. The main module of this project contains no CCL statements, although it does contain substantial comments on the two submodules.

The first submodule, DrillDowns, has simple CCL statements that count the number of movie bids for the two available formats, DVD and VHS. For the DVD bids, the number of occurrences for each genre, as defined on eBay is calculated. In addition, the number of bids for each subgenre of the Sports genre is calculated.

By itself, this data is no different from aggregations seen in other examples. In this example, however, a Java (1.5 or later) output adapter is provided. This adapter makes use of jFreeChart to render a Pie Chart for each of the count aggregations. The charts are redrawn every three seconds (as specified by a parameter in the CCL). For example, the genre analysis pie chart appears like this:



The directions for setting up and running the output adapter are contained in the comments in the CCL and in the following file:

```
data-file-base-folder
\
version
\examples\Web\eBayBidMonitoring\setup\adapters\outputAdapter
\readme.txt
```

where *data-file-base-folder* is the full path and name of the Adapters Base Folder on your computer, and *version* is the version of the Sybase CEP Engine.

The second submodule is called Alerts. Alerts is a request/response demonstration that uses an input adapter written in Python (2.4 or later). The module detects when another seller is selling the same item as a requested set of sellers and has a current bid that is higher. A seller whose competition is getting a better price can then examine more closely why this is the case. The input adapter allows the adding of sellers to the set of sellers of interest. It also provides the means to remove sellers from the set of sellers of interest, by using Delete Statements. Alerts are generated only for sellers of interest.

The input adapter provides a simple console-driven user interface:

```
uris are not specified.
Assuming:
To add seller userid:
  ccl://localhost:6789/Stream/Default/Bids/Alerts/MySellers
```

```
To remove seller userid:
  ccl://localhost:6789/Stream/Default/Bids/Alerts/NotMySellers
Enter as many commands as desired, separately, with a Return in
between each.
Commands are either:
add <sellerUserid>
remove <sellerUserid>
exit
add S0905
add S0269
remove S0905
exit
```

See the CCL comments in the main module for suggestions on seller IDs to enter and which streams and windows to view. The directions for setting up and running the input adapter are contained in comments in the CCL and in the following file:

```
data-file-base-folder
\
version
\examples\Web\eBayBidMonitoring\setup\adapters\inputAdapter
\readme.txt
```

LiveJournalAlerts

An example that enables users to subscribe to notifcations for new posts made by one or more users in LiveJournal.com.

This example demonstrates a very common "templatized queries" implementation, when a large number of users want to execute similar queries with different parameters.

MarketingCampaignMonitoring

An example that computes very simple correlations between the number of people who have clicked on an ad for a given product, and the number of people who have bought the product.

The input streams RandomClicks and RandomTransactions are randomly generated. The output stream BadCampaigns contains all campaigns for which the ratio of the number of transactions for the ad to the number of clicks on an ad is less than 0.5.