New Features Summary

Sybase Event Stream Processor 5.0

Contents

Sybase Event Stream Processor 5.0 Succeeds Aleri Streaming Platform 3.2

Sybase® Event Stream Processor 5.0 is the next generation of the Aleri Streaming Platform.

Version 5.0 also includes many features from Sybase CEP R4.

This document summarizes new and changed functionality. For more information on migrating from Aleri Streaming Platform, see the *Migration Guide*.

CCL

Continuous Computation Language (CCL) replaces AleriML as the primary event processing language of Sybase Event Stream Processor.

CCL is based on Structured Query Language (SQL), but is adapted for event stream processing. Functionally it is similar to AleriML, but has these differences:

- Uses SQL-like statements instead of XML tags
- Let you express complex queries that are not limited to AleriML primitives

Sybase Event Stream Processor 5.0 CCL is the next generation of the language, simplified and streamlined. CCL comes from Sybase CEP R4, but does not support all of the syntax and semantics of Sybase CEP R4. For CCL syntax and usage details, see the *CCL Programmers Guide*.

CCL retains the ability to embed SPLASH code. You can use SPLASH to define custom functions and operators, called Flex operators (Flex streams in Aleri). The SPLASH language is unchanged. For SPLASH details, see the *SPLASH Programmers Reference* and the *SPLASH Tutorial*.

Event Stream Features

CCL event streams offer greater flexibility than AleriML:

- Streams and windows are distinct. In AleriML all event streams were called streams, but they had state, making them more like windows. In CCL, both streams and windows process events. Windows have state, meaning they can retain and store data, while streams are stateless and cannot.
- Streams do not have primary keys.
- Streams support only inserts. Windows, depending on their type, support inserts, updates, deletes, and upserts.
- Delta streams, like regular streams, are stateless, but can have primary keys, and can process updates and deletes. You can use delta streams for optimization in some use cases,

such as downstream from a window where updates and deletes must be preserved, but there is no need to retain state.

- Streams and windows can be input, local, or output. Local streams and windows are visible only inside the Event Stream Processor project; only output streams and windows are visible externally, simplifying choices for users implementing downstream applications.
- A Flex operator can produce either a stream or a window.

Flexible Continuous Queries

CCL can define a stream or window to contain a continuous query. The query is not "typed," that is, you need not identify it as a join, aggregate, or other stream type as in Aleri. For ease of use, the standard query types are available, but generic stream and window objects can contain unconstrained CCL queries.

CCL supports inner joins, in addition to other join types supported by AleriML.

Note: The Event Stream Processor compiler is more strict with joins than the Aleri compiler. Some cases may migrate, but not compile on the Event Stream Processor Server.

Compilation

CCL queries are converted to an executable form by the CCL compiler. Compilation is typically performed within Event Stream Processor Studio, but it can also be performed by invoking the CCL compiler from the command line.

Modularity

New modularity features facilitate reuse and team development, and improve your ability to maintain large, complex projects.

When you create and load modular CCL code:

- **CREATE MODULE** statement defines a module.
- Modules can contain streams and windows, which can contain continuous queries.
- Modules can reference declared parameters, where the parameters are set when the module is loaded.
- **IMPORT** statement lets you reference declarations, modules, libraries of external C/C++ and Java functions, and other elements defined in separate `.ccl` files. **IMPORT** is similar to `#include` in C/C++.
- **LOAD MODULE** statement invokes a module, sets parameters, and defines bindings to streams and windows that supply data to the module, and receive data from the module.
- You can load the same module multiple times in a single project.

Error Monitoring

Specialized error streams capture errors and report on the records that caused them.

Error streams can assist in debugging errors during development. It a production environment, they can provide real-time monitoring for your projects.

Error streams are identical to other user-defined streams with two key exceptions:

- An error stream receives records from its source stream or window only when there is an error on the source stream or window. The record it receives is the input to the source stream or window that caused the error.
- An error stream has a predefined schema that cannot be modified.

See *Advanced CCL Programming Techniques > Error Streams* in the *CCL Programmers Guide*.

Projects

ESP projects replace Aleri data models.

Similar to Aleri models, ESP projects hold stream and window definitions, and any additional SPLASH code. However, they are much more flexible, for example, in the event processing options they support, and their runtime configurability.

An ESP project contains:

- One or more `.ccl` files, including a main file, and others that are imported into the main file
- A `.cclnotation` file that contains the diagram that visually represents the CCL
- One or more `.ccr` files that contain the project configuration to specify runtime options
- A `.ccx` executable file for compiled projects

ESP Studio

ESP Studio expands the editing and testing facilities of Aleri Studio.

Visual and Text Authoring

Use the Visual editor or the CCL editor to edit projects in ESP Studio.

The Visual editor lets you create and edit projects without learning CCL syntax. Similar to the editor in Aleri Studio, it adds support for new functionality, including new stream and window types, simple query objects that are comparable to Aleri Stream types, and advanced features such as modularity.

The CCL editor lets you edit CCL files as text, and provides syntax completion options, syntax checking, and error validation. Experienced programmers can work in the CCL editor exclusively, or use it as a supplement to the Visual editor.

You can easily switch between the Visual editor and the CCL editor at any time. Changes made in one editor are reflected in the other.

Run-Test Environment

The Studio Run-Test environment has been restructured with a navigation tool that lets you start and stop projects on multiple ESP servers, and run the test tools against any live project.

Cluster Architecture

A new, advanced clustering architecture supports more flexible deployments. You can start and stop projects in the cluster without managing physical resources.

While Aleri was limited to a single live project per server, Event Stream Processor 5.0 clustering architecture offers these features:

- An ESP server can now be a virtual server, called a *cluster*, that spans any number of physical machines.
- An ESP server can simultaneously run any number of projects, subject to the capacity of the available hardware resources.
- You start a project on the ESP Server, not on a specific machine. The server assigns it to a machine.
- Connections to streams or windows in projects are made through a URL that consists of `server.workspace.project.element`. The cluster resolves the URL.
- Projects run in workspaces, providing a namespace.
- Projects can be started and stopped independently of all other projects running on the server.
- Projects can be designated for warm standby, so that if the machine the project is running on fails, it restarts automatically on another machine in the cluster.
- Hot standby is also available, where a machine fails over to a live backup instance, avoiding the lag time of a restart.
- The cluster manager is fully redundant. Multiple cluster managers run as peers, avoiding any single point of failure.

Dynamic Continuous Queries

The new cluster architecture supports very dynamic environments.

You can add new continuous queries to a running server at any time in the form of new projects. You can stop projects that are no longer needed at any time without affecting other projects (other than downstream projects).

For example, to perform a new aggregation on an existing output stream in a live project, you can create a new project that subscribes to the existing output stream, and applies the desired aggregation (plus filtering and any other computations). You then add this project to the server. It becomes live immediately, subscribing to the existing project, and runs until stopped.

Datatypes

Event Stream Processor has several new datatypes.

New and Renamed Datatypes

- `bigdatetime` – a timestamp with microsecond granularity.
- `interval` – with microsecond granularity.
- `binary` – represents a raw binary buffer. Maximum length of value is platform-dependent; there is no absolute limit. NULL characters are permitted.
- `boolean` – true or false. The following values are converted, and are case-insensitive:

Values	Converted To
1, On/ON, Yes/YES/Y, True/TRUE/T	true
0, Off/OFF, No/NO/N, False/FALSE/F	false

For more about supported datatypes, see the *CCL Programmers Guide*. See the *Adapters Guide* for datatype mappings for specific adapters.

Note: The names of some existing datatypes have changed. For details, see the *Migration Guide*.

Money Datatype Enhancements

You can now set the `money` datatype precision for each use, rather than globally, using the `money (`*`n`*`)` format, where *n* can be a value from 1 to 15. The supported range of values depends on the specified precision. For example:

- `money(1)`: -922337203685477580.8 to 922337203685477580.7

- `money(15)`: -9223.372036854775808 to 9223.372036854775807

Specify explicit precision for money constants with **Dn** syntax, where*n* represents the precision. For example, 100.1234567D7, 100.12345D5.

The `money` datatype with global precision is also supported.

For usage and conversion details for money datatypes, see the *CCL Programmers Guide*.

Pub/Sub API

Event Stream Processor includes a new Pub/Sub API.

Any custom integration that uses the Aleri 3.x Pub/Sub API must be updated to use the new API.

Some features of the new API include:

- Better performance.
- Simpler – fewer steps to set up a project, and start publishing or subscribing.
- The C++ API has been replaced by a C API to avoid compiler dependencies, allowing you to use the compiler of your choice.
- A choice of programming models for receiving events: direct, call back, or select mode.
- Threading options.

For full details, see the Event Stream Processor SDK documentation.

Adapters

Event Stream Processor includes several adapter enhancements.

- The database adapter supports a wide range of databases connecting by ODBC.
- The JMS adapter supports any JMS provider.
- Most adapters are available on all supported platforms.

For a full list of supported adapters, see *Adapters Supported by Event Stream Processor > Adapter Summary* in the *Adapters Guide*.

Analytic Functions

The Event Stream Processor library of built-in functions has over 70 new functions.

- **Numeric** – `atan2`, `compare`, `cosd`, `cosh`, `distance`, `distancesquared`, `log2`, `nextval`, `pi`, `power`, `random`, `sign`, `sind`, `sinh`, `tand`, and `tanh`
- **Bitwise operations** – `bitand`, `bitclear`, `bitflag`, `bitmask`, `bitnot`, `bitor`, `bitset`, `bitshiftleft`, `bitshiftright`, `bittest`, `bittoggle`, and `bitxor`
- **Sets** – `avgof`, `coalesce`, `maxof`, and `minof`
- **Aggregation** – `corr`, `covar_pop`, `covar_samp`, `exp_weighted_avg`, `meandeviation`, `median`, `regr_avgx`, `regr_avgy`, `regr_count`, `regr_intercept`, `regr_r2`, `regr_slope`, `regr_sxx`, `regr_sxy`, `regr_syy`, `var_pop`, `var_samp`, and `vwap`
- **String functions** – `ascii`, `char`, `left`, `regexp_firstsearch`, `regexp_replace`, `regexp_search`, `trim`, and `upper`
- **Date and time** – `dateceiling`, `datefloor`, `dateround`, `dayofmonth`, `dayofweek`, `dayofyear`, `timeToUsec`, `hour`, `microsecond`, `minute`, `month`, `second`, `usecToTime`, `to_bigdatetime`, `to_date`, `to_timestamp`, `to_interval`, and `year`
- **Binary** – `extract`, `concat`, `length`, `to_binary`, `hex_binary`, `base64_binary`, `to_string`, `string`, `hex_string`, `base64_string`, `tonetbinary`, and `fromnetbinary`
- **Column access** – `get*columnbyname(record, colname)`, `get*columnbyindex(record, colindex)`, `isInsert`, and `isDelete`

Project Configuration

Project configuration data is separate from business logic.

All physical deployment configuration elements, such as host names, are defined in project configuration files that are separate from the CCL that defines the business logic. This makes it easier to move projects from a test environment to a production environment, or to deploy a project in multiple locations.

Installation and Licensing

Event Stream Processor uses SySAM licensing mechanism that includes a 30-day grace period.

For information about supported platforms and operating systems, see the *Installation Guide*.