DYALOG

The tool of thought for expert programming

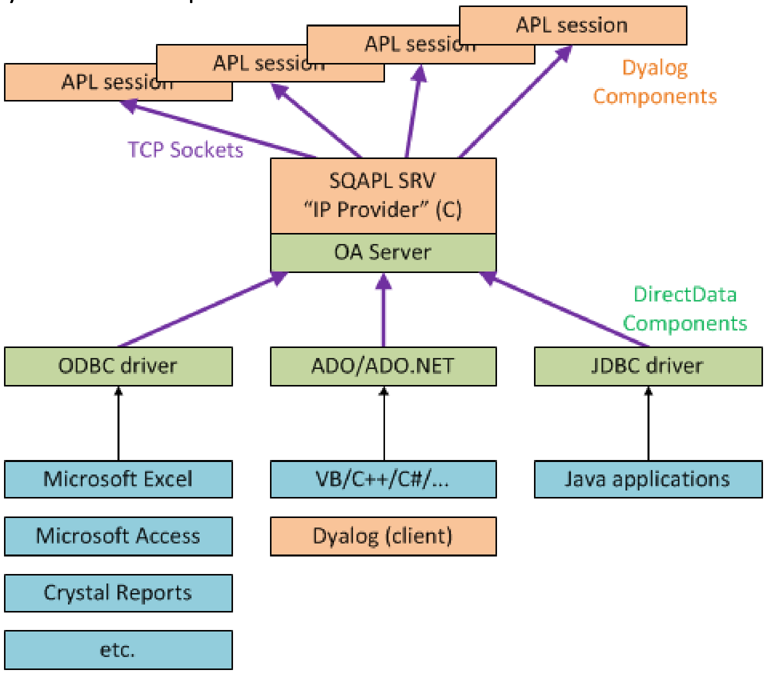
Dyalog SQAPL Server

The Dyalog SQAPL Server can make any APL application SQL-accessible from applications that are ODBC, JDBC, OLE DB or ADO.NET compliant within a very short time.

Dyalog has joined the ranks of over one hundred independent software vendors who have used the Progress® DataDirect® OpenAccess™ SDK to create custom drivers. The Dyalog SQAPL Server is an implementation of this SDK; it extends the OpenAccess framework so that the required data access functions can be implemented in APL. Embedding an APL application in the Dyalog SQAPL Server opens the application's data through open standards and dramatically improves accessibility to it from widely-used reporting, analysis and development tools.

HIGHLIGHTS

- ***Solid technical foundation***
- ***Designed for developers***
- ***Support for a large portion of SQL 92***
- ***Optimised query execution***
- ***Specification compliance resulting in application compatibility***
- ***Broad platform support***
- ***Customisable finished driver***



Dyalog SQAPL Server Components

Features

- **Solid technical foundation** – the underlying OpenAccess SDK has been used by over one hundred organisations to build drivers for different types of data sources.

- **SQAPL is designed for APL developers** – although the purpose of the SQAPL Server is to make APL data available to client developers using any programming language, the server wraps the OpenAccess API in array-oriented covers that make it easy for APL developers to efficiently deliver data to clients. The API can be used in a loop-free fashion, making high-level calls using data arrays of any size.

- **The included SQL engine supports a large portion of SQL 92** – this allows queries with joins, unions, nested queries, stored procedures, inserts, update, deletes, group bys, order bys and other SQL syntax to be executed over any data source.

- **Optimised query execution** – this means that the work to execute the query can either be performed in its entirety by the OpenAccess SQL engine or it can be selectively pushed down to your APL code.

- **OpenAccess ODBC, JDBC, OLE DB, and ADO.NET APIs offer a high level of compliance to their respective specifications** – this guarantees compatibility with any application written to the driver specification and with applications that are compatible with Microsoft's SQL Server, Oracle, Sybase, DB2 and other commercial RDBMSs.

- **Broad platform support** – the driver can support access to data sources on Microsoft Windows, Linux and UNIX, with consistent functionality across all supported platforms.

- **The finished driver can be completely customised** – possible customisations include the driver name, error message pre-fix, error messages and configuration dialog boxes.

- **A fully worked example is included** – providing sample data source implementation including schema functions, index or key column specifications, selections, inserts and updates and stored procedures.

Implement SQL Support in Days

Up to 99% of the code required to create a standards-based driver with SQL support is included in the Dyalog SQAPL Server. When implemented, the 22 data-source-specific APL functions that are provided handle reading from and writing to your data source and exposing the schema. The OpenAccess layer manages the implementation of the ODBC, JDBC, OLE DB and ADO.NET APIs for the SQL processing and the client/server networking framework. The Dyalog SQAPL Server layer offers a simple framework for APL developers to provide application-specific connectors.

The Dyalog SQAPL Server includes a fully worked example implementation of an APL data source that can be customised to your application. If suitable data access functions are already available for your application, then a simple implementation of SQL access can be up and running in a few hours or days.

Unparalleled Flexibility and Universal Access to Your Data

SQAPL is supported on all platforms on which Dyalog APL and the OpenAccess SDK are both available, including 32-bit and 64-bit Microsoft Windows and Linux and 64-bit AIX.

Immediate Results – High Performance Final Solution

The included SQL parser and execution engine provide a high degree of flexibility. As each query arrives, the application code is presented with a parse tree of the incoming SQL statement. At run-time, the application can declare to the framework how the current query should be handled; optionally, this decision can be based on inspection of the parse tree.

In order of increasing difficulty (and decreasing performance) the options are:

- Return the entire contents of each table involved in the query to the framework (the framework then performs all SQL language processing)
- Receive information about filters on key columns and return reduced data sets
- Process all filters in the query in the APL code
- Process the SQL statement including JOINs but excluding GROUP BY and HAVING statements
- Process the entire SQL statement and return the exact result set that should be delivered to the client

In each of these options except the last one, the SQL engine verifies that all filters have been completely processed. This means that the APL application can return a superset of the requested information and the SQL engine can do the last few steps if doing so makes the overall process simpler and more efficient.

The Dyalog SQAPL Server framework and the OpenAccess SDK combine to provide a highly flexible environment for the implementation of an SQL data source; this means that you can have your system running in a few hours as well as having the possibility of building a high performance database engine if the additional effort is justified.

Technical Details

To implement a new data source in APL, the 22 data-source-specific APL functions that are provided need to be inspected (and, possibly, modified). These functions are described in the following table.

Function(s)	Description
LOGIN	Validate supplied user credentials and establish new session environment
TRANSACT	Commit/rollback transactions (if supported by the application)
CATALOGS, SCHEMAS, TABLETYPES	Return fundamental schema information
TABLES, COLUMNS, STAT, FKEY	Return information about individual tables, columns, indices and foreign keys
SELECT, PASSTHROUGH, FIRSTBLOCK, NEXTBLOCK	Different phases of (optionally) optimised query handling
INSERT	Insert records
UPDATE, SELECTUPDATE	Perform updates
DELETE, SELECTDELETE	Perform deletions
PROCS, PROCCOLS	Schema information for stored procedures
PRECEDURE, NEXTSET	Execute stored procedures, return result sets

The Dyalog SQAPL Server package includes a fully worked example that can be used as a starting point for a new server implementation. Architectural advice and consulting is available from Dyalog and its partners.