Galax Version 1.0 Documentation

"The XQuery Implementation for Discriminating Hackers"

Mary Fernández, AT&T Labs Research
Jérôme Siméon, IBM T.J. Watson Research Center

November 10, 2011

Contents

Part I

User's Manual

Chapter 1

Getting Started

1.1 What is XQuery 1.0?

XQuery 1.0 is a query language for XML, defined by the World-Wide Web Consortium (W3C), under the XML activity. XQuery is a powerful language, which supports XPath 2.0 as a subset and includes expressions to construct new XML documents, SQL-like expressions to perform selection, joins, and sorting over collections of XML values, operations on namespaces, and expressions over XML Schema types. XQuery is a functional language, which comes with an extensive library of built-in functions, and allows user to define their own functions. More information about XQuery can be found of the XML Query Working Group Web page[1].

1.2 What is Galax?

Galax is an implementation of XQuery 1.0 designed with the following goals in mind: completeness, conformance, performance, and extensibility. Galax is open-source, and has been used on a large variety of real-life XML applications. Galax relies on a formally specified and open architecture which is particularly well suited for users interested in teaching XQuery, or in experimenting with extensions of the language or optimizations.

Here is a list of the main Galax features.

- Galax implements the January 2007 (W3C Recommendation) set of specifications for XQuery.
- Galax supports Minimal Conformance, as well as the following optional features: Static Typing, Full Axis, Module, and Serialization. The following table lists Galax 1.0 results with respect to the XQuery Tests Suite version 1.0.2:

	Pass	Fail	Total	Percent
Minimal Conformance	14553	71	14637	(99.4%)
Optional Features				
Static Typing Feature	46	0	46	(100%)
Full Axis Feature	130	0	130	(100%)
Module Feature	32	0	32	(100%)

- Galax supports Unicode, with native support for the UTF-8 and ISO-8859-1 character encodings.
- Galax is portable and runs on most modern platforms.

[1] `http://www.w3c.org/XML/Query`

- Galax includes a command-line interface, APIs for OCaml, C, and Java, and a simple Web-based interface.
- Galax includes partial support for the Schema Import and Schema Validation optional features. Unimplemented features of XML Schema include simple types facets and derivation by extension and restriction on complex types.
- Galax supports the XQuery 1.0 Update Facility[2].
- Galax supports the XQueryP scripting extension for XQuery[3].

Alpha features: The following features are experimental.

- Galax inludes support for SOAP and WSDL-based Web Services invocation;
- Galax inludes support for Distributed XQuery development;
- Galax's compiler includes an optimizer that supports state of the art database and programming language optimization, including:
  - type-based optimizations, notably eliminating unnecessary type matching or casting operations and turning dynamic dispatch into static dispatch for comparison and arithmetic operators;
  - tail-recursion optimization;
  - join and query unnesting optimizations, including physical support for XQuery-specific hash and sort join algorithms;
  - tree-pattern detection in query plans and physical support for the TwigJoin and StaircaseJoin algorithms;
  - generation of hybrid streamed/materialized query plans.
- Galax has an *extensible data model* interface. This allows users to provide their own implementation of the XML data model. This can be used to support e.g., XML queries over legacy data.

Limitations: See Chapter 4 for details on Galax's alignment with the XQuery and XPath working drafts.

1.2.1 Changes since the last version

The following lists the main changes included with this version (1.0).

- Galax 1.0 supports the latest OCaml 3.10 compiler and GODI 3.10.
- Support for the XQuery Update Facility 1.0, August 2007 Working Draft. Support for the XQuery 1.0 Update Facility Static Typing Feature.
- Improved implementation of XQueryP.
  - Re-implementation of while loops.
  - Fixed issues with scoping.
- Improved support for modules.
  - Properly implemented nested imports.
  - Added support for module interfaces.

[2] `http://www.w3.org/TR/xquery-update-10/`
[3] `http://www.ximep-2006.org/papers/Paper-Chamberlin-Carey.pdf`

- Support for XQueryX trivial embedding.
- Bug fixes:
  - Fixed problems with support for in-scope namespaces.
  - Fixed problems with checking of cyclic variable declarations.
  - Fixed several problems with the parser, using wrong lexical states inside constructors.
  - Small bug fixes in support for the prolog.

Changes from older versions can be found in Chapter 8.

1.3 Downloading and installing Galax

The official distribution can be downloaded from the main Galax Web site[4]. Detailed installation instructions are provided in Chapter 2.

1.4 How to use Galax

The Galax processor offers the following user interfaces:

- Command-line.
- Application-programming interfaces (APIs) for OCaml, C, or Java.
- Web interface.

1.4.1 Using Galax from the command line

A number of stand-alone command-line tools are provided with the Galax distribution. Assuming the Galax distribution is intalled in `$GALAXHOME`, and that Galax executables are reachable from your `$PATH` environment variable, The following examples show how to use the main command-line tools.

galax-run The main XQuery interpreter (**galax-run**) is the simplest way to use Galax. For instance, the following commands:

```
% echo "<two>{ 1+1 }</two>" > test.xq
% galax-run test.xq
<two>2</two>
```

evaluates the query `<two>{ 1+1 }</two>` and prints the XML result `<two>2</two>`.

galax-parse The Galax XML parser and XML Schema validator can be called as a standalone tool. For instance, the following command validates the document in `hispo.xml` against the schema in `hispo.xsd`:

```
% galax-parse -validate -xmlschema $GALAXHOME/examples/docs/hispo.xsd \
                                   $GALAXHOME/examples/docs/hispo.xml
```

galax-mapschema A stand-alone tool that maps XML Schema documents into the XQuery type system. This tools is useful for checking whether Galax recognizes all the constructs in your XML Schema. It also eliminates a lot of the "noise" in XML Schema's XML syntax.

For instance, this command will print out the XQuery type representation of the schema in `hispo.xsd`:

```
% galax-mapschema $GALAXHOME/examples/docs/hispo.xsd
```

Chapter 5 describes the command-line tools in detail.

[4] `http://www.galaxquery.org`

1.4.2 Web interface

The Web interface is a simple and convenient way to get acquainted with Galax. It allows users to submit a query, and view the result of compilation and execution for that query.

An on-line version is available on-line at: `http://www.galaxquery.org/demo/galax_demo.html`

You can also re-compile the demo from the Galax source and install it on your own system. You will need an HTTP server (Apache is recommended), and follow the compilation instructions in Section 2.4.

1.4.3 Language API's

Galax supports APIs for OCaml, C, and Java. See Chapter 6 for how to use the APIs.

If you have installed the binary distribution of Galax, all three APIs are available.

If you have intalled the source distribution of Galax, you will need to select the language(s) for which you need API support at configuration time. See Chapter 2 for details on compiling Galax from source.

Examples of how to use Galax's APIs can be found in the following directories:
`$GALAXHOME/examples/caml_api/`
`$GALAXHOME/examples/c_api/`
`$GALAXHOME/examples/java_api/`.

Chapter 2

Installation

This chapter contains the following sections:

- Prerequisites (Section 2.1) : prerequisites for Galax installation.
- Source Distribution (Section 2.2) : Installation instructions for the source distribution.
- Running the W3C XQuery Test Suite (XQTS) (Section 2.3).
- Web-form interface (Section 2.4) : installation instructions for Galax's web-form interface.

2.1 Prerequisites

- PCRE 5.0 or higher is required. If you do not have PCRE version 5.0 or higher[1] installed.
- (Optional) If you plan to use Jungle, Galax's secondary storage manager, you will need Berkeley DB version 4.4 or higher[2] installed.

2.2 Source Installation

There are two methods for installing Galax from source:

1. **RECOMMENDED METHOD:** Use GODI installation and configuration tool to automatically install Galax and all the libraries on which it depends.
2. Install Galax and the libraries on which it depends manually.

   This method may be preferable if you are an OCaml user, already have an installation of OCaml and its libraries, and do not want to install a new version of OCaml. Even if you are already an OCaml user, the first method is always recommended.

2.2.1 Installing Galax using GODI installation and configuration tool

Warning: Installing packages using GODI is an ON-LINE process, so preferably, you should be connected to the Net by a fast, hard-wire connection.

[1] `http://www.pcre.org/`
[2] `http://dev.sleepycat.com/downloads/releasehistorybdb.html`

1. Download GODI from : `http://godi.ocaml-programming.de/`

   Follow GODI installation instructions through "bootstrap_stage2" command, adding the `<godi-prefix-path>/bin` and `<godi-prefix-path>/sbin` directories to your PATH as instructed.

   Note: For OCaml users, after GODI bootstrapping phase, the OCaml executables are located in `<godi-prefix-path>/bin` and the OCaml libraries are located in `<godi-prefix-path>/lib/ocaml`.

2. Run `godi_console`

   (a) Select `conf-pcre` package (Packages are listed in alphabetical order.)
   Select [b]uild & install, then e[x]it.
   GODI will give you the PCRE library configure options:

   ```
   [ 1] GODI_BASEPKG_PCRE = no
   [ 2]  GODI_PCRE_INCDIR =
   [ 3]  GODI_PCRE_LIBDIR =
   ```

   Select 2 to set the path to the directory containing pcre.h (version 5.0 or higher).
   Select 3 to set the path to the directory containing libpcre.a (version 5.0 or higher).
   Select e[x]it twice to return to the main menu.

   (b) (Optional) If you are planning to include support for Jungle, Galax's secondary storage manager, then select `conf-bdb`.
   Select [b]uild & install, then e[x]it.
   GODI will give you the BDB library configure options:

   ```
   [ 1] GODI_BDB_INCDIR =
   [ 2] GODI_BDB_LIBDIR =
   ```

   Select 1 to set the path to the directory containing db.h. (version 4.4 or higher).
   Select 2 to set the path to the directory containing libdb.a (version 4.4 or higher).
   Select e[x]it twice to return to the main menu.

   (c) Select `godi-galax` package.
   Select [b]uild & install, then e[x]it.
   This will add all of Galax's dependencies to your GODI configuration. GODI will give you Galax's configure options:

   ```
   [ 1] GODI_GALAX_INCLUDE_JUNGLE = no
   [ 2]  GODI_GALAX_INCLUDE_C_API = no
   [ 3] GODI_GALAX_INCLUDE_JAVA_$ = no
   [ 4]   GODI_GALAX_INCLUDE_UTF8 = yes
   [ 5] GODI_GALAX_INCLUDE_ISO88$ = yes
   [ 6]      GODI_GALAX_JAVA_HOME = unset
   [ 7]       GODI_GALAX_JAVA_BIN = unset
   [ 8]   GODI_GALAX_JAVA_INCLUDE = unset
   [ 9] GODI_GALAX_REGRESSION_SU$ = unset
   ```

   You can change the default Galax configuration as follows:
   Select 1 to include Jungle, Galax's secondary storage manager.
   Select 2 to include Galax's C API.
   Select 3 to include Galax's Java API (requires the C API).
   Select 4(5) to exclude UTF8(ISO88) character set (reduces sizes of executables).
   Select 6(7,8) to set the Java home(bin,include) directories.

Select 9 to set directory where XQuery 1.0 test suite is installed.

Select e[xi]t.

Select [s]tart/continue, which will report all dependencies.

Then select [s]tart/continue, which will report actual dependencies based on your current GODI installation.

Then select [o]k.

3. Take a long coffee break...while GODI downloads and installs packages.

4. Finish

   Add `<godi-prefix-path>/bin` to your `PATH`.

   When installation has completed, Galax will be installed in `<godi-prefix-path>/` with the following subdirectories:

Directory	Sub-directory	Content
`bin/`		Galax's command-line executables
`lib/`		
	`pervasive.xq`	Signatures of XQuery 1.0 Function and Operators
	`c/`	C libraries and API header files
	`java/`	Java libraries and API interfaces
	`ocaml/pkg-lib/galax/`	OCaml libraries and interfaces
`share/galax/`		
	`examples/`	Examples of using C, Java, and OCaml APIs
	`regress/`	Test harness and configuration for the W3C XQuery Test Suite
	`usecases/`	XQuery 1.0 usecases

2.2.2 Installing Galax source by hand

If you already have an OCaml installation, you can install the Galax source by hand. To do so, you need the following versions of the OCaml compiler and libraries, and C libraries:

Library	Version	GODI-Package
OCaml compiler[3]	3.10	godi-ocaml
OCaml Libraries		
findlib[4]	1.1.2pl1	godi-findlib
Pcre-ocaml[5]	5.12.2	godi-pcre
Ocamlnet[6]	2.2.8.1	godi-ocamlnet
Caml IDL[7]	1.05	godi-camlidl
PXP[8]	1.2.0test1	godi-pxp
Camomile[9]	0.7.1	godi-camomile
C Libraries		
Berkeley DB (libdb)	4.3	Only for Jungle
PCRE	>4.5	

1. Download Galax source from .

2. Unzip and untar Galax in a directory of your choice. `gunzip galax-1.0.tar.gz`

   `tar xvf galax-1.0.tar`

3. Configure Galax.

   From the `galax/` you just created, run:

   `./configure -galax-home $GALAXHOME`

Where `$GALAXHOME` is the target installation directory.

If you want the C API, add `-with-c`.

If you want the Java API, add `-with-java -java-home <Top-level Java directory>`.

The configure script will try to find your OCaml installation and other necessary libraries by itself. Review the default configuration before proceeding. If you need to change the default configuration, run `./configure -help` for all options.

4. Build Galax.

   In `galax/` run:

   ```
   make world
   ```

   Go get more coffee...

5. Install Galax.

   In `galax/` run:

   ```
   make install
   ```

6. Finish

   Add `$GALAXHOME/bin` to your `PATH` environment variable.

   When installation has completed, Galax will be installed in `$GALAXHOME` with these subdirectories:

Directory	Content
`bin/`	Command-line executables
`examples/`	Examples of using C, Java, and OCaml APIs
`regress/`	Test harness and configuration for the W3C XQuery Test Suite
`usecases/`	XQuery 1.0 usecases

2.3 XQuery Test Suite

The Galax distribution includes a test harness for the W3C XQuery tests suite. It can be run by following the steps listed below.

1. **Download and install the XQuery Test Suite.** The latest version of the XQuery Test Suite can be found on the W3C XQuery Web page at the following address:

   `http://www.w3.org/XML/Query/test-suite/`

   The Test Suite comes as a `zip` archive, which can be unzipped in a directory of your choice (We use `$XQTS` in the following instructions).

2. **Configure Galax's test harness.** If you followed the GODI installation path, you need to select option 9 as indicated above to specify the directory in which the test suite has been installed. If you followed the "by hand" installation path, you must specify the directory in which the test suite has been installed using the following configure option:

   ```
   -regression $XQTS
   ```

3. **Run the test suite.** You can run the test suite by calling:

   ```
   make
   ```

   from the `$GALAXHOME/regress` directory. This will produce a file called:

   `testresults-W3C.xml`

   which contains the result of running the test suite (following the format required for test results by the XQTS).

4. **Generate a test suite summary.** The XQTS distribution contains an XSLT style sheet which can be used to produced a summary of the test results in HTML form. The corresponding style sheet is located in the directory:

   `$XQTS/ReportingResults`

   The stylesheet can be run by:

   (a) Specifying the path to the test results obtained from the previous step in the configuration file:

   `$XQTS/ReportingResults/Results.xml`

   as follows:

   ```
   <results>
      <result>$GALAXHOME/regress/testresults-W3C.xml</result>
   </results>
   ```

   (b) Changing to the directory:

   `$XQTS/ReportingResults/Results.xml`

   and running one of the two following commands (you will need a working installation of ant, and of an XSLT processor):

   ```
   ant -f Build.xml create
   ant -f Build.xml createsimple
   ```

   which should respectively produce the following HTML files:

   ```
   XQTSReportSimple.html
   XQTSReport.html
   ```

2.4 Web Site Interface

The Galax Web site and on-line demo are bundled with the source distribution (only).

2.4.1 Prerequisites

The Galax Web site has only been tested with Apache Web servers. We recommend you use Apache as some of the CGI scripts might be sensitive to the server you are using. Apache is commonly installed with most Linux distributions, or can be downloaded from: `http://www.apache.org/`.

2.4.2 Installation of Website Interface

1. From the source directory, edit `galax/website/Makefile.config` and initialize the following variables:

   `WEBSITE` Location of the Apache directory for Galax.

   `CGIBIN_PREFIX` Prefix of directory that may contain CGI executables. If empty, then they are placed in `WEBSITE`.

2. Compile the the demo scripts: `make`

3. Install the web site and demo scripts: `sudo make install`

4. Configure your own Apache server: If all the galax demo files (HTML, XML and CGIs) are placed in one directory where the HTTP daemon is expecting to find them, then it is necessary to add the following config info to the proper `http.conf` file:

```
<Directory "/var/www/html/galax">
  Options All
  AllowOverride None
  AddHandler cgi-script .cgi
  Order allow,deny
  Allow from all
</Directory>
```

This permits scripts with suffix .cgi in `/var/www/html/galax` to be executed. The Galax demo is available at `http://localhost/galax`.

5. OR Configure an existing multi-user Apache server.

   Your sysadmin may already have set up an Apache server for general use, and allows CGI programs by any user. You can verify by finding directives similar to the following in httpd.conf (wherever it might be located on your system),

```
AddHandler cgi-script .cgi
<DirectoryMatch "/galax/cgi-bin">
  AllowOverride AuthConfig
  Options ExecCGI
  SetHandler cgi-script
</DirectoryMatch>
```

   In that case, simply follow the comments in `website/Makefile.config` to choose installation destinations for your CGI programs and the HTML documents should suffice. The URL for accessing the installed site will depend on how your webserver is set up. Consult your sysadmin or webadmin for further help.

Chapter 3

Tutorial

3.1 Executing a query

The simplest way to use Galax is by calling the **galax-run** interpreter from the command line. This chapter describes the most frequently used command-line options. Chapter 5 enumerates all the command-line options.

Before you begin, follow the instructions in Section 2.2 and run the following query to make sure your environment is set-up correctly:

```
% echo '<two>{ 1+1 }</two>' > test.xq
% galax-run test.xq
<two>2</two>
```

Galax evaluates expression `<two>{ 1+1 }</two>` in file `test.xq` and prints the result `<two>2</two>`.

By default, Galax parses and evaluates an XQuery main module, which contains both a prolog and an expression. Sometimes it is useful to separate the prolog from an expression, for example, if the same prolog is used by multiple expressions. The `-context` option specifies a file that contains a query prolog.

All of the XQuery use cases in `$GALAXHOME/usecases` are implemented by separating the query prolog from the query expressions. Here is how to execute the Parts usecase:

```
% cd $GALAXHOME/usecases
% galax-run -context parts_context.xq parts_usecase.xq
```

The other use cases are executed similarly, for example:

```
% galax-run -context rel_context.xq rel_usecase.xq
```

3.2 Accessing Input

You can access an input document by calling the `fn:doc()` function and passing the file name as an argument:

```
% cd $GALAXHOME/usecases
% echo 'fn:doc("docs/books.xml")' > doc.xq
% galax-run doc.xq
```

You can access an input document by referring to the context item (the "." dot variable), whose value is the document's content:

```
% echo '.' >dot.xq
% galax-run -context-item docs/books.xml dot.xq
```

You can also access an input document by using the `-doc` argument, which binds an external variable to the content of the given document file:

```
% echo 'declare variable $x external; $x' > var.xq
% galax-run -doc x=docs/books.xml var.xq
```

3.3 Controlling Output

By default, Galax serializes the result of a query in a format that reflects the precise data model instance. For example, the result of this query is serialized as the literal `2`:

```
% echo "document { 1+1 }"> docnode.xq
% galax-run docnode.xq
document { 2 }
```

If you want the output of your query to be as the standard prescribes, then use the `-serialize standard` option:

```
% galax-run docnode.xq -serialize standard
2
```

By default, Galax serializes the result value to standard output. Use the `-output-xml` option to serialize the result value to an output file.

```
% galax-run docnode.xq -serialize standard -output-xml output.xml
% cat output.xml
2
```

3.4 Controlling Compilation

By default, Galax compiles the given query an returns the corresponding result. The following options can be set to print the query as it progresses through the compilation pipeline .

`-print-expr [on/off]` *Print input expression*
`-print-normalized-expr [on/off]` *Print expression after normalization*
`-print-rewritten-expr [on/off]` *Print expression after rewriting*
`-print-logical-plan [on/off]` *Print logical plan*
`-print-optimized-plan [on/off]` *Print logical plan after optimization*
`-print-physical-plan [on/off]` *Print physical plan*

As the output for the compiled query can be quite large, it is often convenient to set the output to verbose using `-verbose on`, which prints headers for each phase. For instance, the following command prints the original query, and the optimized logical plan for the query.

```
% galax-run docnode.xq -verbose on -print-expr on -print-optimized-plan on
```

3.5 Updates and Procedural Extensions

Galax supports several extensions to XQuery 1.0, notably XML updates and a procedural extensions. To enable one of those extensions, you must use the corresponding language level option on the command line:

```
galax-run -language ultf       (: W3C Update Facility :)
galax-run -language xquerybang (: XQuery! Language :)
galax-run -language xqueryp    (: XQueryP Language :)
```

Some examples of each of the three languages are provided in the `$GALAXHOME/examples/extensions` directory.

Part II

Reference Manual

Chapter 4

XQuery 1.0 Conformance

This chapter documents the relationship of Galax to the target W3C working drafts. Galax 1.0 is a prototype implementation, and therefore it is not (yet) completely aligned with the relevant W3C working drafts (WDs). This chapter also document the non-standard features in Galax 1.0 and the known bugs and limitations.

Galax 1.0 implements the January, 2007 XQuery 1.0 and XPath 2.0 Candidate Recommendations, the XML 1.0 Recommendation, the Namespaces in XML Recommendation, and XML Schema Recommendation (Parts 1 and 2).

Galax 1.0 implements the XQuery 1.0 Recommendations:

- XQuery 1.0 and XPath 2.0 Data Model. W3C Recommendation January, 2007. (`http://www.w3.org/TR/xpath-dat`
- XQuery 1.0 : An XML Query Language. W3C Recommendation January, 2007. (`http://www.w3.org/TR/xquery/`)
- XQuery 1.0 and XPath 2.0 Formal Semantics. W3C Recommendation January, 2007. (`http://www.w3.org/TR/xque`
- XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Recommendation January, 2007. (`http://www.w3.org/T`
- XML Query Use Cases. W3C Working Draft 23 March 2007. (`http://www.w3.org/TR/2007/NOTE-xquery-use-cas`
- XML Schema Part 1: Structures and Part 2: Datatypes. W3C Recommendation 2 May 2001. `http://www.w3.org/TR/xmlschema-1/`, `http://www.w3.org/TR/xmlschema-2/`.

4.1 Data Model

Galax 1.0 fully supports the XQuery 1.0 and XPath 2.0 Data Model.

Galax 1.0 implements an xsd:float value as an xsd:double value.

4.2 XQuery

The alignment issues in this section follow the outline of the "Expressions" section in `http://www.w3c.org/TR/xquery`. If a subsection is not listed here, it means that Galax 1.0 implements the semantics described in that section.

XQuery Section 2.1.1 Static Context

Galax 1.0 does not support:

- **XPath 1.0 compatibility modes**.
- **Collations**.

- The **construction mode**. Type annotations on copied elements are always erased/eliminated.
- The **ordering mode**. Input order is always preserved.

XQuery Section 2.1.2 Dynamic Context

The **implicit timezone** is set to the local timezone.

XQuery Section 2.2 Processing Model

Galax's processing model is similar to XQuery's abstract processing model. See Section 7 for more information on Galax's internal processing model.

XQuery Section 2.2.4 Serialization

XQuery Section 2.3.3 Effective Boolean Value

Galax 1.0 does not check that a numeric value is equal to NaN when computing an effective boolean value.

XQuery Section 2.3.4 Input Sources

Galax 1.0 does not support the `fn:collection()` function.

The context item and values for external variables can be specified on the command line or in the API. See Sections 5.1.1 and 6.2.

XQuery Section 2.4.4 SequenceType Matching

Galax requires that all actual types, that is, those types that annotate input documents be in the in-scope schema definitions. Galax will raise a dynamic error if it encounters a type in a document that is not imported into the query by an **import schema** prolog statement.

XQuery Section 2.6 Optional Features

Galax supports the **Schema Import**, **Static Typing**, and **Full Axis** features.

XQuery Section 2.6.4 Module Feature

Galax 1.0 supports the Module feature.

XQuery Section 2.6.5 Pragmas

Galax 1.0 does not support the **Pragmas** feature.

XQuery Section 2.6.6 Must-Understand Extensions

Galax 1.0 does not support must-understand extensions.

XQuery Section 2.6.7 Static Typing Extensions

Galax 1.0 does not support static typing extensions.

XQuery Section 3.1.1 Literals

Galax 1.0 implements an xsd:float value as an xsd:double value.

XQuery Section 3.2.1.1 Axes

Galax 1.0 supports all axes with the exception of the **preceding** and **following** axes.

XQuery Section 3.7 Constructors

When constructing a new element, Galax 1.0always erases/eliminates type annotations on copied elements.

When constructing a new element, Galax 1.0requires that the new element's attributes precede its other content.

XQuery Section 3.7.4 In-scope Namespaces of a Constructed Element

Namespace declarations in input and output documents and in input queries are not handled consistently. We are working on this.

XQuery Section 3.9 Ordered and Unordered Expressions

Galax 1.0 will accept queries that contain the `ordered` or `unordered` expressions, but they have no effect on query evaluation (i.e., they are no-ops).

XQuery Section 4.2 Module Declaration

XQuery Section 4.2 Module Import

Galax 1.0 supports the Module feature.

XQuery Section 4.4 Default Collation Declaration

Galax 1.0 does not support collations.

XQuery Section 4.6 Construction Declaration

Galax 1.0 does not support the construction declaration.

XQuery Section 4.8 Default Ordering Declaration

Galax 1.0 does not support the default ordering declaration.

XQuery Section 4.9 Schema Import

Schema components in an imported schema are mapped into XQuery types according to the mapping rules specified in the XQuery 1.0 Formal Semantics (see below).

4.3 XQuery 1.0 Formal Semantics

The XQuery 1.0 formal semantics defines the mapping of every XQuery expression into an expression in the XQuery core, and it defines the static and dynamic semantics of each core expression. The formal semantics also defines how imported schemas are mapped into internal XQuery types.

Galax 1.0 implements the static and dynamic semantics of core expressions defined in the XQuery 1.0 Formal Semantics.

4.4 Functions and Operators

Galax 1.0 supports most of the functions in the XQuery 1.0 and XPath 2.0 Functions and Operators document. The signatures of supported functions are listed in `$GALAXLIB/pervasive.xq`.

Galax 1.0 does not support the following functions:

```
fn:id
fn:idref
fn:collection
```

Functions and Operators Section 4 The Trace Function

The `fn:trace` function emits its input sequence and message are to standard output.

Functions and Operators Section 5.2 Constructor Functions for User-Defined Types

Galax 1.0does not support constructor functions for user-defined types.

Functions and Operators Section 12 Functions and Operators on base64Binary and hexBinary

Galax 1.0 does not support any functions on binary data.

4.5 Use Cases

See `$GALAXHOME/usecases/STATUS`

4.6 Galax 1.0 extensions

4.6.1 Defining XQuery Types in the Query Prolog

Type values are available in a query by either importing a predefined XML schema using the `import schema` declaration in the query prolog or by defining XQuery types explicitly in the query prolog.

Galax 1.0 supports the definition of XQuery types in the query prolog using the internal type syntax defined in the XQuery 1.0 Formal Semantics. The grammar is provided here for reference:

```
TypeDeclaration ::=    ("define" "element" QName "{" TypeDefn? "}")
                     | ("define" "attribute" QName "{" TypeDefn? "}")
                     | ("define" "type" QName "{" TypeDefn? "}")

TypeDefn        ::=    TypeUnion
                     | TypeBoth
                     | TypeSequence
                     | TypeSimpleType
                     | TypeAttributeRef
                     | TypeElementRef
                     | TypeTypeRef
                     | TypeParenthesized
                     | TypeNone

TypeUnion        ::= TypeDefn  "|"  TypeDefn
```

```
TypeBoth          ::= TypeDefn  "&"  TypeDefn
TypeSequence      ::= TypeDefn  ","  TypeDefn
TypeSimpleType    ::= QName OccurrenceIndicator
TypeAttributeRef  ::= "attribute" NameTest ("{" TypeDefn? "}")? OccurrenceIndicator
TypeElementRef    ::= "element" NameTest ("{" TypeDefn? "}")? OccurrenceIndicator
TypeTypeRef       ::= "type" NameTest OccurrenceIndicator
TypeParenthesized::= "(" TypeDefn? ")" OccurrenceIndicator
TypeNone          ::= "none"
```

4.6.2 Galax specific functions

Galax-only functions are put in the Galax namespace (http://www.galaxquery.org), which is bound by default to the glx: prefix.

See `$GALAXHOME/lib/pervasive.xq` for a complete list of functions in the Galax namespace.

Chapter 5

Command Line Tools

Galax supports the following stand-alone command-line tools:

galax-run The Galax XQuery interpreter.

galax-parse XML document parser and validator.

galax-mapschema XML Schema validator. Outputs XML Schema in XQuery type system.

galaxd The Galax network server. Executes Galax query plans issued from remote client (Under development).

5.1 `galax-run` : The Galax XQuery interpreter

The simplest way to use Galax is by calling the 'galax-run' interpreter from the command line. The interpreter takes an XQuery input file, evaluates it, and yields the result on standard output.

Usage: `galax-run` *`options`* `query.xq`

For instance, the following commands from the Galax home directory:

```
%  echo "<two>  1+1  </two>" > test.xq
%  $(GALAXHOME)/bin/galax-run test.xq
<two>2</two>
```

evaluates the simple query `<two> { 1+1 } </two>` and prints the XML value `<two>2</two>`.

The query interpreter has eight processing stages: parsing an XQuery expression; normalizing an XQuery expression into an XQuery core expression; static typing of a core expression; rewriting a core expression; factorizing a core expression; compiling a core expression into a logical plan; selecting a physical plan from a logical plan; and evaluating a physical plan.

Parsing, factorization, and compilation are always enabled. By default, the other phases are:

```
-normalize on
-static off
-rewriting on
-optimization on
-dynamic on
```

By default, all result values (XML result, inferred type, etc.) are written to standard output.

The command line options permit various combinations of phases, printing intermediate expressions, and redirecting output to multiple output files. Here are the available options. Default values are in `code` font.

-help,–help Display this list of options

5.1.1 Input options

-base-uri Sets the default base URI in the static context

-var `x`=literal, Binds the global variable `x` to literal constant

-doc `x`=filename, Binds the global variable `x` to document in filename

-context-item Binds the context item ("`.`" variable) to the document in filename or '-' for stdin

-context Load the context query from file or '-' for stdin

-xml-whitespace Preserves whitespace in XML documents [on/`off`]

-xml-pic Preserves PI's and comments in XML documents [on/`off`]

5.1.2 Output options

-serialize Set serialization kind [wf, canonical, or `xquery`]

5.1.3 Evaluation options

-dynamic Evaluation phase [`on`/off]

-execute-normalized-expr All files are assumed to be normalized expressions for execution [on/`off`]

-execute-logical-plan All files are assumed to be logical plans for execution [on/`off`]

-print-xml Print XML result [`on`/off]

-output-xml Output XML to result file

-print-expr Print input expression [on/`off`]

-output-expr Output expr to file.

5.1.4 Normalization options

-normalize Normalization phase [`on`/off]

-print-normalized-expr Print normalized expression [on/`off`]

-output-normalized-expr Output normalized expression to file.

5.1.5 Static typing options

-static Static analysis phase [on/`off`]

-typing Static typing behavior [`none`, weak, or strong]

-print-type Print type of expression [on/`off`]

-output-type Output type to file.

-print-typed-expr Print typed expression [on/`off`]

-output-typed-expr Output typed expression to file.

5.1.6 Rewriting options

-rewriting Rewriting phase (use with -static on) [`on`/off]

-sbdo Document-order/duplicate-elimination optimization behavior [remove, preserve, sloppy, tidy, `duptidy`]

remove Remove all distinct-docorder (ddo) calls from the query plan. May result in faulty evaluation plans.

preserve Preserve all distinct-doc-order calls (no optimization).

sloppy Delay duplicate removal and sorting until the last step in the path expression. May cause the number of duplicates in intermediate results to increase exponentially since they are not removed. Also, subsequent steps are evaluated multiple times when duplicates are generated in a step.

tidy Removes all distinct-doc-order operations hat can be removed while maintaining duplicate-freeness and doc-order after each step.

duptidy Only sorts and removes duplicates after a step if duplicates are generated. Intermediate results can be out of document order.

-print-rewritten-expr Print rewritten expression [on/`off`]

-output-rewritten-expr Output rewritten expression to file.

5.1.7 Factorization options

-factorization Factorization phase [`on`/off]

-print-factorized-expr Print factorized expression [on/`off`]

-output-factorized-expr Output factorized expression to file.

5.1.8 Compilation options

-print-comp-annotations Print compilation annotations [on/`off`]

-print-logical-plan Print logical plan [on/`off`]

-output-logical-plan Output logical plan to file.

5.1.9 Optimization options

-optimization Optimization phase [`on`/off]

-nested-loop-join Turns off sort/hash joins and uses only nested-loop joins [`on`/off]

-print-optimized-plan Print optimized plan [on/`off`]

-output-optimized-plan Output optimized plan to file

-print-physical-plan Print physical plan [on/`off`]

-output-physical-plan Output physical algebraic plan to file

5.1.10 Miscellaneous printing options

-verbose Emit descriptive headers in output [on/`off`]

-print-global Prints everything : prologs, exprs, etc.

-output-all Output everything to file.

-print-error-code Print only the error code instead of the full error message

-output-err Redirect error output to file.

-print-context-item Serializes the context item at the end of query evaluation

-output-context-item Output the context item to the given file

-print-annotations Print expression annotations [on/`off`]

-print-prolog Print query prolog [on/`off`]

-print-materialize Print whenever data materialization occurs [on/`off`]

-force-materialized Force materialization of values in variables [on/`off`]

5.1.11 Document projection options

-projection Document projection behavior [`none`, standard, or optimized]

-print-projection Prints the projection paths

-output-projection Output the projections paths to a file.

-print-projected-file Prints back the input document after projection

-output-projected-file Output the input document after projection into file.

5.1.12 Miscellaneous options

-version Prints the Galax version

-debug Emit debugging [on/`off`]

-monitor Monitors memory and CPU consumption [on/`off`]

-monitor-mem Monitors memory consumption [on/`off`]

-monitor-time Monitors CPU consumption [on/`off`]

-output-monitor Output monitor actibity to file

-internal-encoding Set the input character encoding representation, e.g., `utf8, iso88591`.

-output-encoding Set the output character encoding representation

5.2 `galax-parse`: XML parser and XML Schema validator

`galax-parse` parses an XML document and optionally validates the document against an XML Schema.
Usage: `galax-parse` *`options`* `document.xml`

-help,–help Display this list of options

-xml-whitespace Preserves whitespace in XML documents

-xml-pic Preserves PI's and comments in XML documents

-validate Set validation on

-xmlschema Schema against which to perform validation

-dm Also builds the data model instance

-monitor Monitors memory and CPU consumption

-monitor-mem Monitors memory consumption

-monitor-time Monitors CPU consumption

-output-monitor Output monitor to file

-serialize-namespaces Set serialization of namespace nodes [strip/`preserve`]

-serialize Set serialization kind [canonical, `wf`, or xquery]

-print-error-code Print only the error code instead of the full error message

-output-encoding Set the output encoding representation

5.3 `galax-mapschema` : XML Schema validator

`galax-mapschema` maps XML schemas in xmlschema(s) into XQuery type expressions.
Usage: `galax-mapschema` *`options`* `schema.xsd`

-prefix Namespace prefix

-verbose Set printing to verbose

-import-type Set XML Schema import

-normalize-type Set XQuery type normalization

-print-type Set printing ofXQuery type

-print-normalized-type Set printing of normalized XQuery type

-output-type Output XQuery type in file

-output-normalized-type Output normalized XQuery type in file

5.4 `galaxd` : The Galax network server

`galaxd` is a server that allows Galax to be invoked over the network.
Usage: `galaxd [options] [query.xq]`

-port *n* Listen on port *n*. If the -port option is not used, the port defaults to 3324.

-s *dir* Simulate the directory *dir*. The server will act as if it is a part of a virtual network specified by *dir*. Each file *host.xq* in *dir* defines a server *host* in the virtual network. The virtual network is implemented on the localhost at ports *3324*, *3325*, ... Ports are assigned to the *host.xq* files in lexicographic order.

The query file *query.xq* should define a function named local:main(). An XQuery program can get the result of local:main() on *host* by calling doc("dxq://host/"). If the server is using a non-default port *port*, then use doc("dxq://host:port/").

5.4.1 Running simulations

It is sometimes useful to simulate a network of Galax servers on a single host. The -s option makes this possible. The way to set this up is to create a directory with a query file for each simulated host. For example, create a directory *example* with query files *a.xq*, *b.xq*, and *c.xq*. Each .xq file should define a local:main() function. Also, these files can refer to each other's local:main() functions using doc("dxq://a/"), doc("dxq://b/"), and doc("dxq://c/"). Then start up the three servers as follows:

```
galaxd -s example -port 3324
galaxd -s example -port 3325
galaxd -s example -port 3326
```

`galaxd` uses the -s option to find out what the virtual network will look like: it will have hosts a, b, and c, operating on non-virtual ports 3324, 3325, and 3326 on localhost. The first invocation of `galaxd` above uses port 3324, so it uses a.xq to define its local:main() function. Similarly, the second and third invocations use b.xq and c.xq, respectively.

Chapter 6

Application Programming Interfaces (APIS)

The quickest way to learn how to use the APIs is as follows:

1. Read Section 6.1 "Galax API Support".
2. Read Section 6.2 "Quick Start to the Galax APIs".
3. Read the example programs in the `galax/examples/` directory while reading Section 6.2.

Every Galax API has functions for:

- Converting values in the XQuery data model to/from values in the native programming language (O'Caml, C or Java);
- Accessing values in XQuery data model from the native programming language;
- Loading XML documents into the XQuery data model;
- Creating and modifying the query evaluation environment (also known as the **dynamic context**);
- Evaluating queries given a dynamic context; and
- Serializing XQuery data model values as XML documents.

This chapter describes how to use each kind of functions.

6.1 Galax API Functionality

Galax currently supports application-program interfaces for the O'Caml, C, and Java programming languages.

All APIs support the same set of functions; only their names differ in each language API. This file describes the API functions. The interfaces for each language are defined in:

O'Caml `$GALAXHOME/lib/caml/galax.mli`

C `$GALAXHOME/lib/c/{galax,galax_util,galax_types,itemlist}.h`

Java `$GALAXHOME/lib/java/doc/*.html`

If you use the C API, see Section 6.5.1 “Memory Management in C API”.

Example programs that use these APIs are in:

O’Caml `$GALAXHOME/examples/caml_api`

C `$GALAXHOME/examples/c_api`

Java `$GALAXHOME/examples/java_api`

To try out the API programs, edit examples/Makefile.config to set up your environment, then execute: `cd $GALAXHOME/examples; make all`.

This will compile and run the examples. Each directory contains a ”test” program that exercises every function in the API and an ”example” programs that illustrates some simple uses of the API.

The Galax query engine is implemented in O’Caml. This means that values in the native language (C or Java) are converted into values in the XQuery data model (which are represented are by O’Caml objects) before sending them to the Galax engine. The APIs provide functions for converting between native-language values and XQuery data-model values.

6.1.1 Linking and Running

There are two kinds of Galax libraries: byte code and native code. The C and Java libraries require native code libraries, and Java requires dynamically linked libraries. Here are the libraries:

O’Caml libraries in `$GALAXHOME/lib/caml`:

galax.cma Byte code

galax.cmxa Native code

C libraries in `$GALAXHOME/lib/c`:

libgalaxopt.a Native code, statically linked

libgalaxopt.so Native code, dynamically linked

Java libraries in `$GALAXHOME/lib/java`:

libglxoptj.so Native code, dynamically linked

Note that Java applications MUST link with a dynamically linked library and that C applications MAY link with a dynamically linked library.

For Linux users, set `LD_LIBRARY_PATH` to `$GALAXHOME/lib/c:$GALAXHOME/lib/java`.

The Makefiles in `examples/c_api` and `examples/java_api` show how to compile, link, and run applications that use the C and Java APIs.

6.2 Quick Start to using the APIs

The simplest API functions allow you to evaluate an XQuery statement in a string. If the statement is an update, these functions return the empty list, otherwise if the statement is an Xquery expression, these functions return a list of XML values.

The example programs in `$(GALAXHOME)/examples/caml_api/example.ml`, `$(GALAXHOME)/examples/c_api/example` `$(GALAXHOME)/examples/java_api/Example.java` illustrate how to use these query evaluation functions.

Galax accepts input (documents and queries) from files, string buffers, channels and HTTP, and emits output (XML values) in files, string buffers, channels, and formatters. See `$(GALAXHOME)/lib/caml/galax_io.mli`.

All the evaluation functions require a processing context. The default processing context is constructed by calling the function `Processing_context.default_processing_context()`:

```
val default_processing_context : unit -> processing_context
```

There are three ways to evaluate an XQuery statement:

```
val eval_statement_with_context_item :
  Processing_context.processing_context -> Galax_io.input_spec ->
    Galax_io.input_spec -> item list
```

Bind the context item (the XPath "." expression) to the XML document in the resource named by the second argument, and evaluate the XQuery statement in the third argument.

```
val eval_statement_with_context_item_as_xml :
  Processing_context.processing_context -> item ->
    Galax_io.input_spec -> item list
```

Bind the context item (the XPath "." expression) to the XML value in the second argument and evaluate the XQuery statement in the third argument.

```
val eval_statement_with_variables_as_xml :
  Processing_context.processing_context ->
    (string * item list) list ->
      Galax_io.input_spec -> item list
```

The second argument is a list of variable name and XML value pairs. Bind each variable to the corresponding XML value and evaluate the XQuery statement in the third argument.

Sometimes you need more control over query evaluation, because, for example, you want to load XQuery libraries and/or main modules and evaluate statements incrementally. The following two sections describe the API functions that provide finer-grained control.

6.3 XQuery Data Model

6.3.1 Types and Constructors

In the XQuery data model, a value is a **sequence** (or list) of **items**. An item is either an **node** or an **atomic value**. An node is an **element**, **attribute**, **text**, **comment**, or **processing-instruction**. An **atomic value** is one of the nineteen XML Schema data types plus the XQuery type **xs:untypedAtomic**.

The Galax APIs provide constructors for the following data model values:

- lists/sequences of items
- element, attribute, text, comment, and processing instruction nodes
- xs:string, xs:boolean, xs:int, xs:integer, xs:decimal, xs:float, xs:double, xs:anyURI, xs:QName, xs:dateTime, xs:date, xs:time, xs:yearMonthDuration, xs:dayTimeDuration, and xs:untypedAtomic.

Atomic values

The constructor functions for atomic values take values in the native language and return atomic values in the XQuery data model. For example, the O'Caml constructor:

```
val atomicFloat   : float -> atomicFloat
```

takes an O'Caml float value (as defined in the Datatypes module) and returns a float in the XQuery data model. Similarly, the C constructor:

```
extern galax_err galax_atomicDecimal(int i, atomicDecimal *decimal);
```

takes a C integer value and returns a decimal in the XQuery data model.

Nodes

The constructor functions for nodes typically take other data model values as arguments. For example, the O'Caml constructor for elements:

```
val elementNode : atomicQName * attribute list * node list * atomicQName -> element
```

takes a QName value, a list of attribute nodes, a list of children nodes, and the QName of the element's type. Simliarly, the C constructor for text nodes takes an XQuery string value:

```
extern galax_err galax_textNode(atomicString str, text *);
```

Sequences

The constructor functions for sequences are language specific. In O'Caml, the sequence constructor is simply the O'Caml list constructor. In C, the sequence constructor is defined in galapi/itemlist.h as:

```
extern itemlist itemlist_cons(item i, itemlist cdr);
```

6.3.2 Using XQuery data model values

The APIs are written in an "object-oriented" style, meaning that any use of a type in a function signature denotes any value of that type or a value derived from that type. For example, the function **Dm_functions.string_of_atomicvalue** takes any atomic value (i.e., xs_string, xs_boolean, xs_int, xs_float, etc.) and returns an O'Caml string value:

```
val string_of_atomicValue  : atomicValue -> string
```

Similarly, the function `galax_parent` in the C API takes any node value (i.e., an element, attribute, text, comment, or processing instruction node) and returns a list of nodes:

```
extern galax_err galax_parent(node n, node_list *);
```

6.3.3 Accessors

The accessor functions take XQuery values and return constituent parts of the value. For example, the `children` accessor takes an element node and returns the sequence of children nodes contained in that element:

```
val children : node -> node list      (* O'Caml *)
extern galax_err galax_children(node n, node_list *); /* C */
```

The XQuery data model accessors are described in detail in `http://www.w3c.org/TR/query-datamodel`.

6.3.4 Loading documents

Galax provides the `load_document` function for loading documents.

The `load_document` function takes the name of an XML file in the local file system and returns a sequence of nodes that are the top-level nodes in the document (this may include zero or more comments and processing instructions and zero or one element node.)

```
val load_document : Processing_context.processing_context ->
  Galax_io.input_spec -> node list (* O'Caml *)

extern galax_err galax_load_document(char* filename, node_list *);
extern galax_err galax_load_document_from_string(char* string, node_list *);
```

6.4 Query Evaluation

The general model for evaluating an XQuery expression or statement proceeds as follows (each function is described in detail below):

1. Create default processing context:

   ```
   let proc_ctxt = default_processing_context() in
   ```

2. Load Galax's standard library:

   ```
   let mod_ctxt = load_standard_library(proc_ctxt) in
   ```

3. (Optionally) load any imported library modules:

   ```
   let library_input = File_Input "some-xquery-library.xq" in let mod_ctxt = import_library_module
   pc mod_ctxt library_input in
   ```

4. (Optionally) load one main XQuery module:

   ```
   let (mod_ctxt, stmts) = import_main_module mod_ctxt (File_Input "some-main-module.xq") in
   ```

5. (Optionally) initialize the context item and/or global variables defined in application (i.e., external environment):

   ```
   let ext_ctxt = build_external_context proc_ctxt opt_context_item var_value_list in let mod_ctxt
   = add_external_context mod_ctxt ext_ctxt in
   ```

6. Evaluate all global variables in module context:

   ```
   let mod_ctxt = eval_global_variables mod_ctxt
   ```

   ** NB: This step is necessary if the module contains *any* global variables, whether defined in the XQuery module or defined externally by the application. **

7. Finally, evaluate a statement from the main module or one defined in the application or call some XQuery function defined in the module context:

   ```
   let result = eval_statement proc_ctxt mod_ctxt stmt in
   ```

   ```
   let result = eval_statement_from_io proc_ctxt mod_ctxt (Buffer_Input some-XQuery-statement)
   in
   ```

   ```
   let result = eval_query_function proc_ctxt mod_ctxt "some-function" argument-values in
   ```

6.4.1 Module context

Every query is evaluated in a **module context**, which includes:

- the built-in types, namespaces, and functions;
- the user-defined types, namespaces, and functions specified in any imported library modules; and
- any additional context defined by the application (e.g., the values of the context item and any global variables).

The functions for creating a module context include:

```
val default_processing_context : unit -> processing_context
```

The default processing context, which just contains flags for controlling debugging, printing, and the processing phases. You can change the default processing context yourself if you want to print out debugging info.

```
val load_standard_library : processing_context -> module_context
```

Load the standard Galax library, which contains the built-in types, namespaces, and functions.

```
val import_library_module : processing_context ->
  module_context -> input_spec -> module_context
```

If you need to import other library modules, this function returns the module_context argument extended with the module in the second argument.

```
val import_main_module    : processing_context ->
  module_context -> input_spec ->
    module_context * (Xquery_ast.cstatement list)
```

If you want to import a main module defined in a file, this function returns the module_context argument extended with the main module in the second argument and a list of statements to evaluate.

The functions for creating an external context (context item and global variable values):

```
val build_external_context : processing_context -> (item option) ->
  (atomicDayTimeDuration option) -> (string * item list) list ->  external_context
```

The external context includes an optional value for the context item (known as "."), the (optional) local timezone, and a list of variable name, item-list value pairs.

```
val add_external_context : module_context -> external_context -> module_context
```

This function extends the given module context with the external context.

```
val eval_global_variables : processing_context -> xquery_module -> xquery_module
```

This function evaluates the expressions for all (possibly mutually dependent) global variables. It must be called before calling the eval_* functions otherwise you will get an "Undefined variable" error at evaluation time.

Analogous functions are defined in the C and Java APIs.

6.4.2 Evaluating queries/expressions

The APIs support three functions for evaluating a query: `eval_statement_from_io`, `eval_statement`, and `eval_query_function`.

Note: If the module context contains (possibly mutually dependent) global variables, the function `eval_global_variables` must be called before calling the eval_* functions otherwise you will get an "Undefined variable" error at evaluation time.

```
val eval_statement_from_io : processing_context -> xquery_module -> Galax_io.input_spec -> item list
```

Given the module context, evaluates the XQuery statement in the third argument. If the statement is an XQuery expression, returns Some (item list); otherwise if the statement is an XQuery update, returns None (because update statements have side effects on the data model store, but do not return values).

```
val eval_statement     : processing_context -> xquery_module -> xquery_statement -> item list
```

Given the module context, evaluates the XQuery statement

```
val eval_query_function  : processing_context -> xquery_module -> string -> item list list -> item li
```

Given the module context, evaluates the function with name in the string argument applied to the list of item-list arguments. **Note:** Each actual function argument is bound to one item list.

Analogous functions are defined in the C and Java APIs.

6.4.3 Serializing XQuery data model values

Once an application program has a handle on the result of evaluating a query, it can either use the accessor functions in the API or it can serialize the result value into an XML document. There are three serialization functions: `serialize_to_string`, `serialize_to_output_channel` and `serialize_to_file`.

```
val serialize  : processing_context -> Galax_io.output_spec -> item list -> unit
```

Serialize an XML value to the given galax output.

```
val serialize_to_string : processing_context -> item list -> string
```

Serializes an XML value to a string.

Analogous functions are defined in the C and Java APIs.

6.5 C API Specifics

6.5.1 Memory Management

The Galax query engine is implemented in O'Caml. This means that values in the native language (C or Java) are converted into values in the XQuery data model (which represented are by O'Caml objects) before sending them to the Galax engine. Similarly, the values returned from the Galax engine are also O'Caml values – the native language values are "opaque handles" to the O'Caml values.

All O'Caml values live in the O'Caml memory heap and are therefore managed by the O'Caml garbage collector. The C API guarantees that any items returned from Galax to a C application will not be de-allocated by the O'Caml garbage collector, unless the C appliation explicitly frees those items, indicating that they are no longer accessible in the C appliation. The C API provides two functions in galapi/itemlist.h for freeing XQuery item values:

```
extern void item_free(item i);
```

Frees one XQuery item value.

```
extern void itemlist_free(itemlist il);
```

Frees every XQuery item value in the given item list.

6.5.2 Exceptions

The Galax query engine may raise an exception in O'Caml, which must be conveyed to the C application. Every function in the C API returns an integer error value :

- 0 if no exception was raised or
- -1 if an exception was raised.

The global variable galax_error_string contains the string value of the exception raised in Galax. In future APIs, we will provide a better mapping between error codes and Galax exceptions

6.6 Java API Specifics

6.6.1 General Info

The Galax query engine is implemented in O'Caml. This means that values in the native language (C or Java) are converted into values in the XQuery data model (which represented are by O'Caml objects) before sending them to the Galax engine.

The Java API uses JNI to call the C API, which in turn calls the O'Caml API (it's not as horrible as it sounds).

There is one class for each of the built-in XML Schema types supported by Galax and one class for each kind of node:

Atomic	Node	Item
xsAnyURI	Attribute	
xsBoolean	Comment	
xsDecimal	Element	
xsDouble	ProcessingInstruction	
xsFloat	Text	
xsInt		
xsInteger		
xsQName		
xsString		
xsUntyped		

There is one class for each kind of sequence:

- ItemList
- AtomicList
- NodeList
- AttributeList

There is one class for each kind of context used by Galax:

- ExternalContext
- ModuleContext
- ProcessingContext
- QueryContext

Finally, the procedures for loading documents, constructing new contexts and running queries are in the `Galax` class.

6.6.2 Exceptions

All Galax Java API functions can raise the exception class GalapiException, which must be handled by the Java application.

6.6.3 Memory Management

All Java-C-O'Caml memory management is handled automatically in the Java API.

6.7 Operation-System Notes

Currently, Galax is not re-entrant, which means multi-threaded applications cannot create multiple, independent instances of the Galax query engine to evaluate queries.

6.7.1 Windows

The C API library `libgalaxopt.a,so` does not link properly under MinGW. A user reported that if you have the source distribution, you can link directly with the object files in `galapi/c_api/*.o` and adding the library `-lasmrun` on the command line works.

Chapter 7

For Galax Developers

7.1 Galax Source Code Architecture

The Galax source-code directories roughly correspond to each phase of the query processor. (Put link to Jerome's tutorial presentation here)

The processing phases are:

Document processing

```
  Document Parsing =>
 [Schema Normalization (below) =>]
    Validation =>
      Loading =>
        Evaluation (below)
```

Schema processing

```
  Schema Parsing =>
    Schema Normalization =>
      Validation (above)
      Static Typing (below)
```

Query processing

```
  Query Parsing =>
    Normalization =>
   [Schema Normalization (above) =>]
      Static Typing (optional phase) =>
        Rewriting =>
          Compilation =>
         [Loading (above) =>]
            Evaluation =>
              Serialization
```

7.1.1 General

Makefile

- Main Makefile

base/

- Command-line argument parsing
- Global variables (conf.mlp)
- XQuery Errors
- String pools
- XML Whitespace handling

ast/

- All ASTs: XQuery User & Core, XQuery Type User & Core
- Pretty printers for all ASTs

config/
monitor/

- CPU &/or memory monitoring of each processing phase

toplevel/

- Main programs for command-line tools (see 'Generated executables' below)

website/

- Local copy of Galax web site

7.1.2 Datamodel

datatypes/ (*** Doug)

- XML Schema simple datatypes – Lexers and basic operations
- datatypes_lexer.mll To learn about O'Caml lex, read: `http://caml.inria.fr/ocaml/htmlman/manual026.html` Sections 12.1 and 12.2 Other examples of lexers in lexing/*.mll

  We are going to extend this module to include lexer for: xsd:date, xsd:time, xsd:dateTime, xs:yearMonthDuration, xs:dayTimeDuration (Skip Gregorian types for now, xsd:gDay, xsd:gMonth, etc)
- dateTime.ml,mli This module will implement the datatypes and basic operations

namespace/

- XML Qualified Names (prefix:localname) – Lexer and basic operations – QName resolution prefix =¿ URI
- Names of builtin functions & operators

dm/ (*** Doug)

- Abstract data model interface for Nodes
- Concrete data model implementation for AtomicValues

datamodel/

- Main-memory implementation of abstract data model for Nodes (Document-Object Model or DOM)

jungledm/

- Secondary storage implementation of Galax datamodel (Jungle)

physicaldm/

- Physical data model

streaming/

- XML parser to untyped and typed SAX streams
- export datamodel to SAX stream

7.1.3 Processing Model

procctxt/

- Processing context contains all query-processor state:
  - Parse context
  - Normalization context
  - Static context
  - Rewrite context
  - Dynamic context

procmod/

- Processing model dynamically "glues" together phases (controlled by command-line arguments or API)

7.1.4 Query Parsing

lexing/

- Lexers for XQuery (excludes all simple datatypes)

parsing/

- Parsing context
- Parsing phase

7.1.5 Normalization

normalization/

- Normalization context
- Normalization phase (XQuery AST =¿ XQuery Core AST)
- Overloaded functions

7.1.6 Static Typing

fsa/

- Finite-state Automata for checking sub-typing relation

typing/

- Static-typing context
- Static-typing phase

7.1.7 Schema/Validation

schema/

- Schema-validation context
- Schema normalization phase (XML Schema =¿ XQuery Core Types)
- Document validation phase
- Judgments(functions) for comparing XQuery types

7.1.8 Rewriting

cleaning/

- Logical optimization/rewriting phase
- Sort-by-document order (DDO) optimization

rewriting/

- Generic AST rewriter

7.1.9 Compilation

compile/

- Compilation phase

algebra/

- AST for compiled algebra
- Dynamic context
- Implementations (dynamic) of most built-in functions & operators

7.1.10 Evaluation

evaluation/

- Evaluation phase

stdlib/

- Static typing of built-in functions & operators
- Implementations (dynamic) of built-in functions fn:doc, fn:error
- Signatures of built-in functions & operators (pervasive.xqp) Corresponds to sections in `http://www.w3c.org/TR/xpa`

7.1.11 Serialization

serialization/

- Serialize SAX stream to XML document (in O'Caml formatter)

7.1.12 Testing

usecases/

- Tests of XQuery Usecases Implements examples in `http://www.w3.org/TR/xquery-use-cases/`

examples/

- Tests of O'Caml, C & Java APIs

regress/

- Regression tests (needs separate xqueryunit/ CVS package)

7.1.13 APIs

galapi/

- O'Caml, C & Java APIs to Galax processor

7.1.14 External libraries & tools

tools/

Required tools:

- http
- pcre
- pxp-engine
- netstring

Optional supported tools:

- Jungle

Optional unsupported tools:

- glx_curl
- jabber

7.1.15 Extensions

extensions/

- apache
- jabber

7.1.16 Experimental Galax extensions

projection/

- Document projection

wsdl/
wsdl_usecases/

- Web-service interfaces

7.1.17 Documentation

- Changes Change log!! Protocol: always document your changes in Changes file; use log entry as input message to 'cvs commit'
- BUGS Out of date
- LICENSE
- README
- STATUS
- TODO

7.1.18 Generated executables

ocaml-galax O'Caml top-level interpretor that loads Galax library. Usage: ocaml-galax -I $(HOME)/Galax/lib/caml-devel

galax-run Complete XQuery engine For Usage: galax-run –help See also: all: rule in usecases/Makefile

galax.a Library versions of Galax

galax.cma byte code

galax.cmxa machine code

galax-parse Syntax checking on query, validation on a document

galax-compile Parsing, normalization, optimization, and prints resulting expression

galax-mapschema Takes XML Schema and prints out internal XQuery type

Auxiliary research tools:

galax-mapwsdl Imports/exports Galax queries as WSDL Web Services

xquery2soap

galax-project Takes XQuery query and figures out what fragments of documents are necessary to evaluate the query

Chapter 8

Release Notes

8.1 Galax 1.0 (March 2008)

Galax version 1.0 implements the XQuery 1.0 Recommendation from January, 2007 (`http://www.w3c.org/TR/xquery`) and the XQuery Update Facility 1.0 from August, 2007 (`http://www.w3.org/TR/xquery-update-10/`). It also implements XQueryP, an imperative scripting language that extends XQuery with updates with mutable variables, while loops, and sequential expressions (`http://www.ximep-2006.org/papers/Paper-Chamberlin-Carey.pdf`

- Galax 1.0 must be built with O'Caml 3.10
- Implementation of the latest XQuery Update Facility 1.0 (`http://www.w3.org/TR/xquery-update-10/`), including static typing.
- Improved implementation of XQueryP.
  - Re-implementation of while loops.
  - Fixed issues with scoping.
- Improved support for modules.
  - Properly implemented nested imports.
  - Added support for module interfaces.
  - Support for XQueryX trivial embedding.
  - Bug fixes:
    * Fixed problems with support for in-scope namespaces.
    * Fixed problems with checking of cyclic variable declarations.
    * Fixed several problems with the parser, using wrong lexical states inside constructors.
    * Small bug fixes in support for the prolog.
  - Conformance results on XQTS 1.0.2:

Feature	Pass	Fail	Total	Perc
Minimal Conformance	14555	69	14637	(99.4%) Optional Featu
Static Typing Feature	46	0	46	
Full Axis Feature	130	0	130	
Module Feature	32	0	32	

8.2 Galax 0.7.2 (February 2007)

Galax version 0.7.2 is a minor release, and should be considered as a beta release. Galax 0.7.2 implements the XQuery 1.0 Recommendation from January, 2007.

This is a development release. Notably static typing and some of the new compiler optimizations are not fully tested.

- Compiler:
  - We're still working on Join detection...A never-ending saga.
- Current testing results on XQTS 1.0.2:

Feature	Galax Pass	Fail	Total	Percent
Minimal Conformance	14514	110	14637	(99.1%)
Optional Features				
Schema Import Feature	0	0	174	
Schema Validation Feature	0	0	25	
Static Typing Feature	46	0	46	
Full Axis Feature	130	0	130	
Module Feature	0	0	32	
Trivial XML Embedding Feature	0	0	4	

8.3 Galax 0.6.8 (August 2006)

Galax version 0.6.8 is a minor release, and should be considered as a beta release. Galax 0.6.8 implements the XQuery 1.0 candidate recommendation working drafts from January, 2007.

This is a development source-only release. Notably static typing and some of the new compiler optimizations are not fully tested.

- Language:
  - More bug fixes to align with the January, 2007CR working drafts.
  - Support for the XQuery! language is fixed (was broken in the previous release).
  - Support for the XML update facility (`http://www.w3.org/TR/xquery-update-10/`).
  - Support for the XQueryP extension to XQuery (`http://www.ximep-2006.org/papers/Paper-Chamberlin-Ca`
- Compiler:
  - Join detection is back! Some bug fixes and improvements to the robustness of the optimizer in the presence of rewritings.

8.4 Galax 0.6.5 (May 2006)

Galax version 0.6.5 is a major release, and should be considered as an alpha release. Galax 0.6.5 implements the XQuery 1.0 candidate recommendation working drafts from January, 2007.

This is a development source-only release. Notably static typing and some of the new compiler optimizations are not fully tested.

- Language:
  - Alignment with the January, 2007CR working drafts.
  - Support for XML updates based on the XQuery! language [Alpha] (See `http://xquerybang.cs.washington.e`
  - Support for both strong and weak typing. [Alpha].

  - Complete implementation of XQuery 1.0 Functions and Operators, notably full support for date and time, URI, and QName operations.
  - Countless bug fixes...

- Environment:
  - New configure script for automated configuration, compilation, and installation.
  - Galax is now a GODI package(`http://godi.ocaml-programming.de/`).
  - Added test harness for the W3C XQuery test-suite. Galax passes **7852** out of **7917** tests on version **1.0.2** of the XQuery 1.0 Test Suite. The test suite contains a total of **10,055** tests.

- Compiler:
  - Support for a variety of physical algorithms for evaluating path expressions, including twig joins, staircase joins, and one-pass evaluation over XML token streams. [Alpha]
  - Known problems: Support for hash/sort joins is temporarily disabled.

8.5 Galax 0.5.0 (February 2005)

Galax version 0.5 is a major release, and should be considered as an alpha release. Galax 0.5.0 implements the XQuery 1.0 working draft published in October 2004.

Among the most noticeable changes:

- Alignment with the XQuery 1.0, October 2004 working drafts.
- A much faster XML parser, based on Gerd Stolpmann's PXP, fixing many XML 1.0 conformance bugs as well.
- A completely new compiler, including a query optimizer that supports join optimizations and should deliver much better performances than the previous versions of Galax.
- "Document projection" is finally part of the main release, allowing to process queries over large documents. (see the galax-project command-line tool).
- Improved support for sorting by document order and duplicate removal in the compiler.
- The Windows port is back, based on the MinGW compilers.
- New port for MacOS X.

Have contributed to this release: Mary Fernández, Nicola Onose, Philippe Michiels, Christopher Ré, Jérôme Siméon, Michael Stark.

8.6 Galax 0.4.0 (August 2004)

Galax version 0.4 is a major release, and should be considered as an alpha release. Galax 0.4.0 implements the latest XQuery 1.0 working draft published in July 2004. It contains many improvements from the previous version, as well as new features.

Among the most noticeable improvements and new features: Galax now comes bundled with Jungle, a simple native XML store. It now supports XML Schema and "named typing". Finally, it contains some prototype support for Web services.

Have contributed to this release: Mary Fernández, Vladimir Gapeyev, Nicola Onose, Philippe Michiels, Doug Petkanics, Christopher Ré, Jérôme Siméon, Avinash Vyas.

Main changes over the previous version are listed below.

Language changes:

- Support for the latest XQuery 1.0's specifications (July 2004 Working Drafts).
- Support for XML Schema 1.0: schema import, validate, named typing, sequence types and type tests.
- Preliminary support for modules. Import module statements without recursion are supported. Details of the semantics will be fixed when issues around modules are addressed by the XML Query working group.
- Support for dates and time.
- Support for string regular expressions.

Environment changes:

- A new set of command-line tools replace the old ones: galax-run: The XQuery execution engine galax-parse: XML parsing and XML Schema validation galax-project: To apply XML document projection galax-mapschema: To map XML Schema to the XQuery type system ocaml-galax: The OCaml interpretor bundled with Galax
- New command-line tools for Web services: galax-mapwsdl: To map WSDL interfaces to an XQuery module xquery2soap: To deploy an XQuery module as an apache Web service.
- Revisions to the Caml, C, C++ and Java APIs.

Architectural changes:

- A new extensible data model, making Galax easy to use over 'virtual' XML documents.
- A completely new query compiler and evaluation engine that supports an hybrid SAX-tuple-tree algebra. The new compilation infrastructure should already show improved performances, although it performs little optimizations yet. Expect more work in this area in future versions of the system

New features:

- Alpha support for native XML storage with Jungle (on top of Sleepycat's BerkeleyDB).
- Alpha support for calling Web services from within XQuery.

Portability:

- Added Makefile for Mac OSX in config/Makefile.osx.
- Fixed numerous problems with Win32.

8.7 Galax 0.3.5 (December 2003)

This is a bug-fix release:

- Now compiles with OCaml 3.07.
- Numerous bug fixes to the XML updates support.
- Added glx:document-save() function allowing to save the result of a query (notably useful in the case of an existing document that has been updated).
- Fixed bug in the release of Caml values in the C-API.
- Fixed bug in pretty-printing of function application, and added pretty-printing for the query prolog.
- Fixed index bug in fn:substring and fn:translate.

- Fixed serialization in canonical form.
- Fixed bug in parsing of PIs in the document root.
- Fixed bug in validation/casting of atomic values not dealing with whitespace properly.
- Fixed bug in key/keyref support introduced with the new API.
- Fixed bug in parsing of DTD declarations with the PUBLIC keyword.

8.8 Galax 0.3.1 (June 2003)

Language extensions:

- Alpha support for XML updates!

Command line:

- External variables and context items can be bound from the command-line.

API:

- Brand new, hopefully complete Caml, C, and Java APIs. Check them out!

Parsing:

- Switched for good to the SAX parser. glx:document-sax is removed.
- Complete new support for character encodings. Now detects encoding in XML declaration properly. Support for UTF-16.

8.9 Galax 0.3.0 (January 2003)

Bugs

- All reported and fixed bugs are documented in 'Bugs' file.

Language: Numerous changes to align with Nov. 2002 and upcoming Feb 2003 WDs.

- Grammar alignment: 'as' SequenceType in type declarations, function signatures, typeswitch
- Implements element & attribute constructor semantics of Nov. 2002 WDs.
- Added positional variables to FLWOR
- Added support for order-by in FLWOR
- Implemented complete semantics of path expressions, including document order and removing duplicates.
- Implemented dynamic function dispatch, promotion of arguments to arithmetic operators
- Transitive 'eq' operators

Galax features:

- Command-line options for monitoring memory and CPU usage.

Data model:

- Updated terminology to align with WDs.

Function library:

- Changed xf: to fn: prefix
- Added fn:error()
- Added support for fn:base-uri() and fn:lang()
- New semantics for fn:data() and fn:boolean()

8.10 Galax 0.2.0 (October 2002)

Parsing

- Fixed very large number of bugs. Support for entities and DTDs is still missing.
- Support for ISO-8859-1 and UTF-8 character encodings.
- Factorized XML and XQuery parsers. Results in more compact code, easier to improve and maintain.
- Updated XQuery parser to align with latest grammar.
- Alpha support for SAX-based parsing.

Data model:

- Support for node identity.
- Fixed support for text nodes.

Language:

- Major revision based on August 16th 2002 working drafts.
- Full support for XPath expressions. Notably XPath parent, ancestor, ancestor-or-self axis. See STATUS for some remaining deviations.
- Support for node-identity related operations (is, isnot distinct-nodes, union, intersect, except, etc.).
- Support for type promotion in function calls, arithmetics, etc.
- Fixed many bugs in the semantics, through normalization.
- Added dynamic semantics for type operations through (simplified) form of matching. Still some bugs there.

Namespaces:

- Fixed many bugs in namespace support and printing of namespaces.
- Implemented support for default function namespace.

XML Schema:

- *Very* alpha support for XML Schema import and validation. Basic datatypes are now supported.

Type system:

- Updated type inference with the new language.

- Made sure all expressions are type checked, necessary for optimization.
- Fixed bugs in typing for typeswitch.

Function library:

- Most of F&O functions are now implemented. Some limitations apply to XML Schema types not yet supported (notably date and time).

Optimizer:

- Support for simple query simplification.

Compilation:

- Removed dependency to the stdlib file.
- Removed dependency to anything but OCaml compilers and standard unix tools.
- Fixed compilation for both Cygwin and MinGW.

Tests:

- Added large number of regression tests.

Tools and interfaces:

- Removed Java API for now. Will be back soon.
- Added very limited user-level api (Galapi) in Caml.
- Changed galax command-lined interpretor. See the new syntax and options in ./README
- Major revision of the pretty-printer for types and XQuery expressions. Added support for precedence in both cases.

Documentation:

- Added examples of calls to the Caml and C API's in ./example.

Chapter 9

General

9.1 The Galax Team

Contributors:

- Mary Fernández (Lead)
- Trevor Jim
- Philippe Michiels
- Kristi Morton
- Nicola Onose
- Chris Rath
- Christopher Ré
- Jérôme Siméon (Lead)
- Michael Stark

Alumni:

- Byron Choi
- Vladimir Gapeyev
- Amélie Marian
- Douglas Petkanics
- Gargi Sur
- Avinash Vyas
- Philip Wadler

9.2 Feedback

Feedback and bug reports can be sent by mail to: `galax-users@research.att.com`
You can be subscribe to the Galax mailing list at: `galax-users-request@research.att.com`
You can report bugs at `http://bugzilla.galaxquery.net/`.

9.3 Copyright and License

Galax version 1.0 is distributed under the terms of the LUCENT PUBLIC LICENSE VERSION 1.0 - see the LICENSE file for details.

```
Lucent Public License Version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS PUBLIC
LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE
PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a.in the case of Lucent Technologies Inc. ("LUCENT"), the Original
Program, and

b.in the case of each Contributor,

i.changes to the Program, and

ii.additions to the Program; where such changes and/or additions to
the Program originate from and are "Contributed" by that particular
Contributor.

A Contribution is "Contributed" by a Contributor only (i) if it was
added to the Program by such Contributor itself or anyone acting on
such Contributor's behalf, and (ii) the Contributor explicitly
consents, in accordance with Section 3C, to characterization of the
changes and/or additions as Contributions.

"Contributor" means LUCENT and any other entity that has Contributed a
Contribution to the Program.

"Distributor" means a Recipient that distributes the Program,
modifications to the Program, or any part thereof.

"Licensed Patents" mean patent claims licensable by a Contributor
which are necessarily infringed by the use or sale of its Contribution
alone or when combined with the Program.

"Original Program" means the original version of the software
accompanying this Agreement as released by LUCENT, including source
code, object code and documentation, if any.

"Program" means the Original Program and Contributions or any part
thereof

"Recipient" means anyone who receives the Program under this
Agreement, including all Contributors.

2. GRANT OF RIGHTS
```

a.Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b.Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. The patent license granted by a Contributor shall also apply to the combination of the Contribution of that Contributor and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license granted by a Contributor shall not apply to (i) any other combinations which include the Contribution, nor to (ii) Contributions of other Contributors. No hardware per se is licensed hereunder.

c.Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d.Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A. Distributor may choose to distribute the Program in any form under this Agreement or under its own license agreement, provided that:

a.it complies with the terms and conditions of this Agreement;

b.if the Program is distributed in source code or other tangible form, a copy of this Agreement or Distributor's own license agreement is included with each copy of the Program; and

c.if distributed under Distributor's own license agreement, such license agreement:

i.effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii.effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits; and

iii.states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party.

B. Each Distributor must include the following in a conspicuous location in the Program:

Copyright (C) 2003, Lucent Technologies Inc. and others. All Rights Reserved.

C. In addition, each Contributor must identify itself as the originator of its Contribution, if any, and manifest its intent that the additions and/or changes be a Contribution, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution. Once consent is granted, it may not thereafter be revoked.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Distributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for Contributors. Therefore, if a Distributor includes the Program in a commercial product offering, such Distributor ("Commercial Distributor") hereby agrees to defend and indemnify every Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Distributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Distributor in writing of such claim, and b) allow the Commercial Distributor to control, and cooperate with the Commercial Distributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Distributor might include the Program in a commercial product offering, Product X. That Distributor is then a Commercial Distributor. If that Commercial Distributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Distributor's responsibility alone. Under this section, the Commercial Distributor would have to defend claims against the Contributors related to those performance claims and warranties, and if a court requires any Contributor to pay any damages as a result, the Commercial Distributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. EXPORT CONTROL

The Recipient acknowledges that the Program is "publicly available" as the term is defined under the United States export administration regulations and is not subject to export control under such laws and regulations. However, if the Recipient modifies the Program to change (or otherwise affect) such publicly available status, the Recipient agrees that Recipient alone is responsible for compliance with the United States export administration regulations (or the export control laws and regulation of any other countries) and hereby indemnifies the Contributors for any liability incurred as a result of the Recipients actions which result in any violation of any such laws and regulations.

8. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Lucent Technologies Inc. may publish new versions (including revisions) of this Agreement from time to time. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. No one other than Lucent has the right to modify this Agreement. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

9.4 Bugs and Limitations

9.5 Known Bugs and Limitations

Galax's error messages are often uninformative. We are working on this.

Namespace declarations in input and output documents and in input queries are not handled consistently. We are working on this.

Although module declarations and module import statements are supported, they are not well tested.