XINS

A framework for
distributed applications

Version 4. April 2006
Ernst de Haan

Audience

- Intended for:
  - managers
  - developers
  - system administrators

- Experience with XINS is not required

Goals

<u>Inform about:</u>

- frameworks:
  - concept
  - applicability, pros/cons
- XINS:
  - history
  - design principles
  - features
  - qualification

Agenda

- Frameworks
- Fundamentals
- Features
- Qualification

Developing an application

- Where do you start?
- Requirements!

Application requirements

- Functional:
  - driven by the project
  - specific per application
  - cannot be generalized, only tunneled
- Non-functional:
  - typically equal for all applications

Non-functional requirements

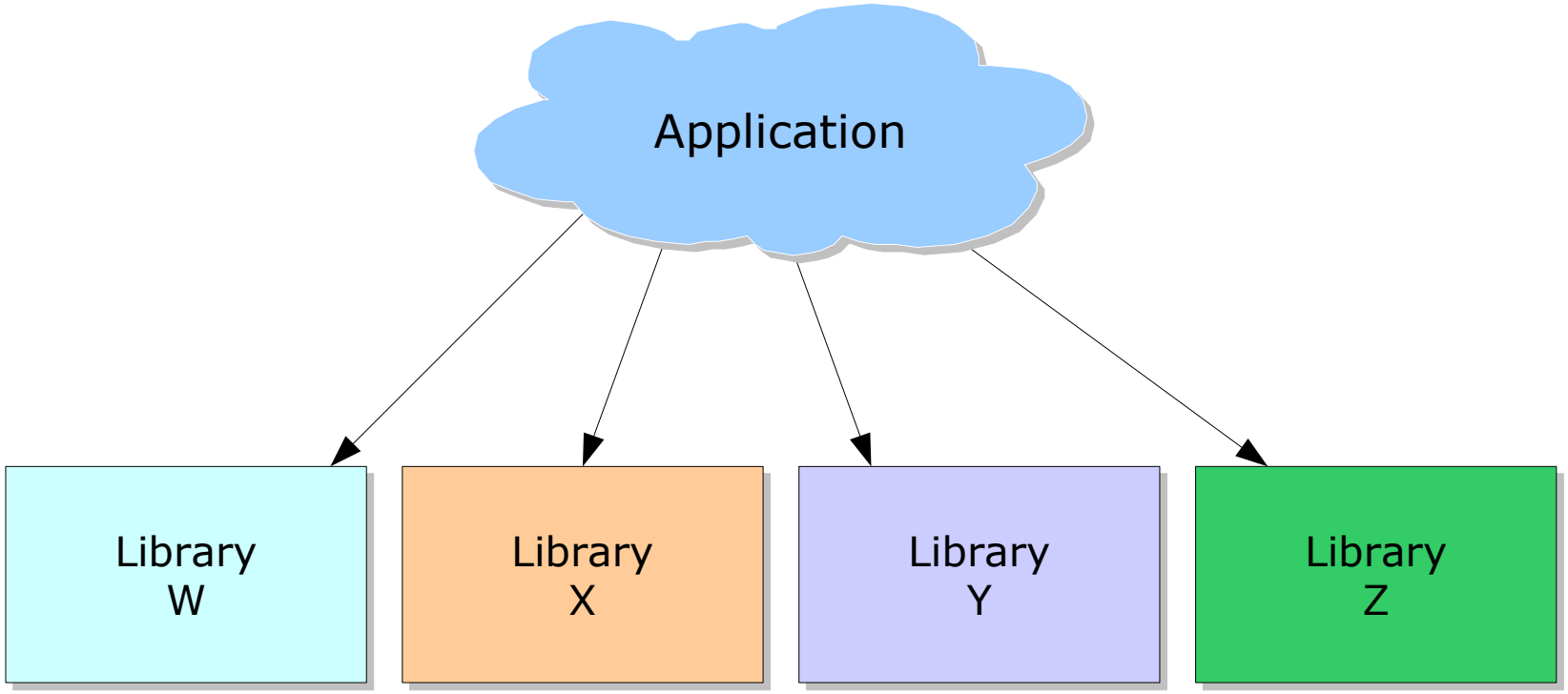
- For example:
  - packaging
  - deployment
  - configuration management
  - monitoring
  - performance statistics
  - transaction logging
  - error detection
  - ...

Non-functional requirements

- Options:
  - Rebuild every time
  - Copy/paste
  - Use libraries

Step 1: Libraries

- Share functionality, avoid duplication



Limitations of libraries

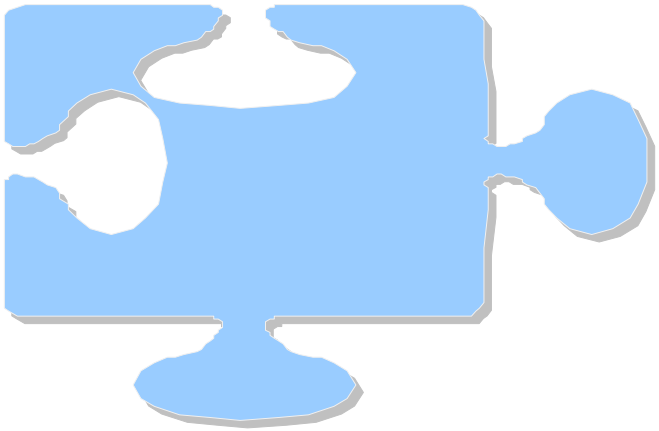
- Only provide functionalities
- Glue still needed
- Libraries may overlap or even conflict
- By definition cannot solve certain issues:
  - unified packaging
  - unified deployment
  - unified testing
  - …

Step 2: Framework

- Combine several libraries
- Add some glue and tools
- Ta-da! Unified approach

Framework

Application



What is a framework?

- Functionalities + glue + tools
- No conflicts
- Unified approach to certain non-functional requirements
- Best practices
- It guides and supports, from start to finish
- Domain-specific

Definition

<u>Frame'work</u> (*n*) =

- a systematic approach for developing a certain type of software applications,
- typically including:
  - support programs
  - code libraries

Application architecture

- Architecture borders predefined
- Frozen spots:
  - define overall architecture
  - static: remain same with every application
- Hot spots:
  - dynamic to individual applications

Unification

Unified approach to certain non-functional requirements, for example:

- coding
- configuration management
- logging
- testing
- monitoring
- …

Having no framework

- Features set of applications differ
- Feature implementations differ
- Time needed to build features that are required but non-functional
- Project pushes, code/test period limited
- Innovation expensive, limited
- Duplication of code (copy/paste ?)

Pros

- Quicker from idea to working code
- Extensive feature set
- Same features and behavior everywhere
- Well-tested
- Fuels innovation
  (if the framework is controlled)

Cons

- Framework is domain-specific
- Enforces an approach
  - process
  - programming language
  - packaging
  - restricts use of other libraries?
  - restricts use of other frameworks?

Agenda

- Frameworks
- Fundamentals
- Features
- Qualification

History

Apr 2002 : Project initiated

Jan 2003 : Open-sourced (SourceForge)

Nov 2004 : 1.0 (after 212 pre-releases)

Jan 2005 : 1.1

May " : 1.2

Nov " : 1.3

Apr 2006 : 1.4.0-beta3

May " : 1.4.0-final

XINS as a framework

Domain:

- Distributed applications

Main constraints:

- RPC style
- HTTP
- Java

XINS goals

- Easy to develop distributed applications
  - easy to understand
  - good time to market
  - avoid bugs
- Easy to monitor and operate
- Consistent
- High quality
- Stable

Separation of concerns

Separate:

- specification and implementation
- data and presentation
- logging, code and translations

Simplicity

- Simplicity is key
- Makes it easier to
  - understand
  - tune
  - change

Based on selected standards

- All definitions in XML
- All communication over HTTP
- Avoid dependency on complex standards (e.g. SOAP)

RPC

- Function-oriented (e.g. “GetCart”)
- Server-side:
  - one XINS function
    = one Java method to implement
- Client-side
  - one XINS function
    = one Java method to invoke

DOD

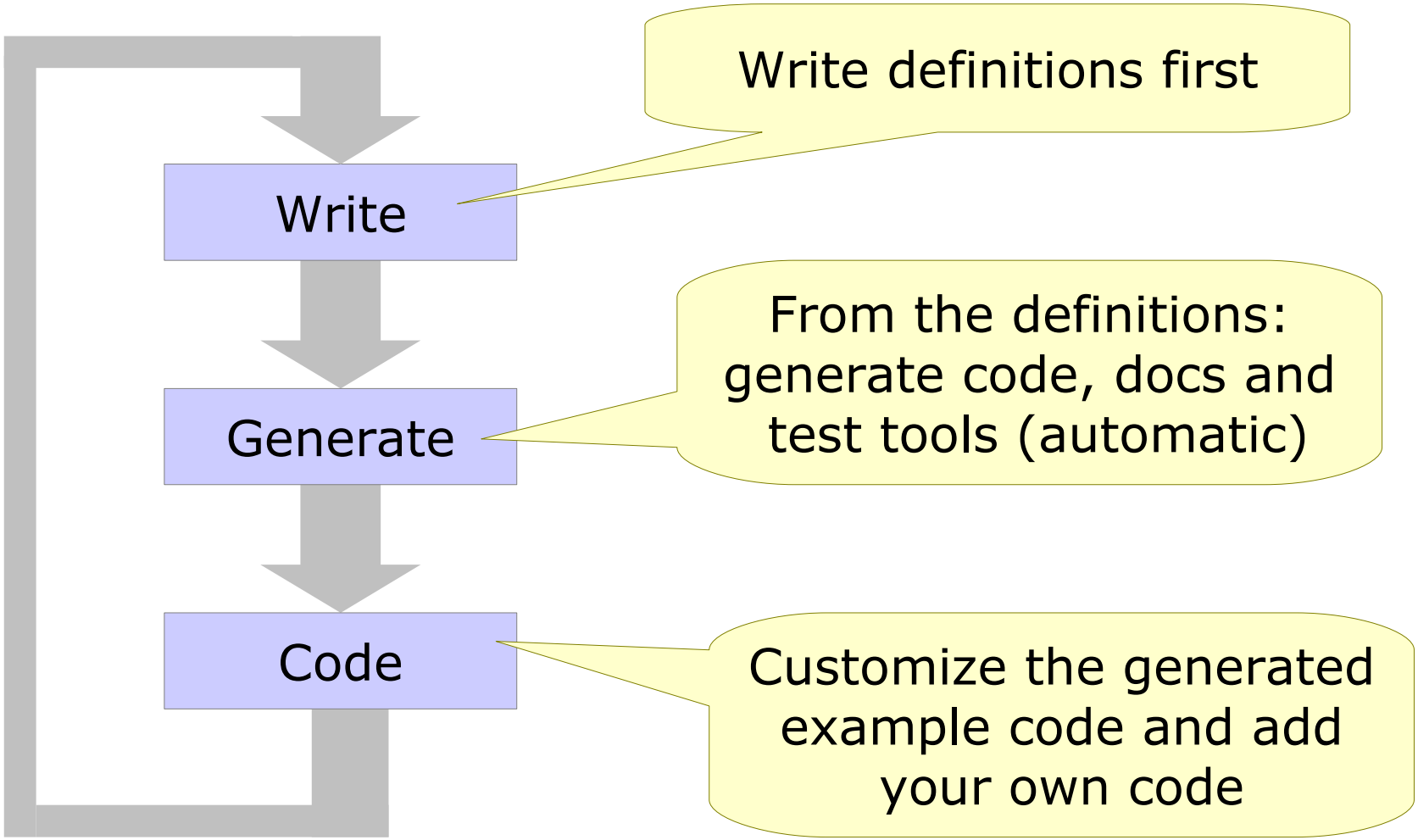
<u>Definition-oriented development:</u>

- Focus on definitions, not on code
- Definitions are mandatory
- Definitions are leading
- Start with definitions before coding

Advantages of DOD

- Easier to re-use:
  - generate code
  - use as run-time configuration
- Examples:
  - behavior (validation, business logic, etc.)
  - documentation
  - tools or tool configurations (test forms, etc.)
  - other kinds of definitions (WSDL, etc.)

DOD process


Write
Generate
Code
Write definitions first
From the definitions: generate code, docs and test tools (automatic)
Customize the generated example code and add your own code

Example: Function definition

```
<function name="Hello">

  <description>Greets the indicated person.</description>

  <input>
    <param name="name" required="true">
      <description>Person to be greeted.</description>
    </param>
  </input>

  <output>
    <param name="greeting" required="true">
      <description>Greeting for the person.</description>
    </param>
  </output>

</function>
```

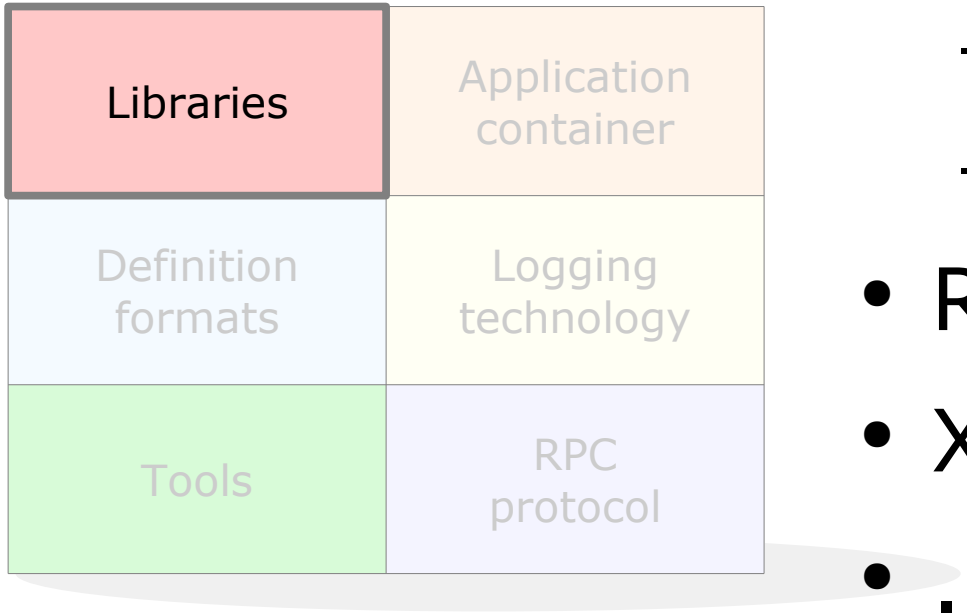
Agenda

- Frameworks
- Fundamentals
- Features
- Qualification

XINS components

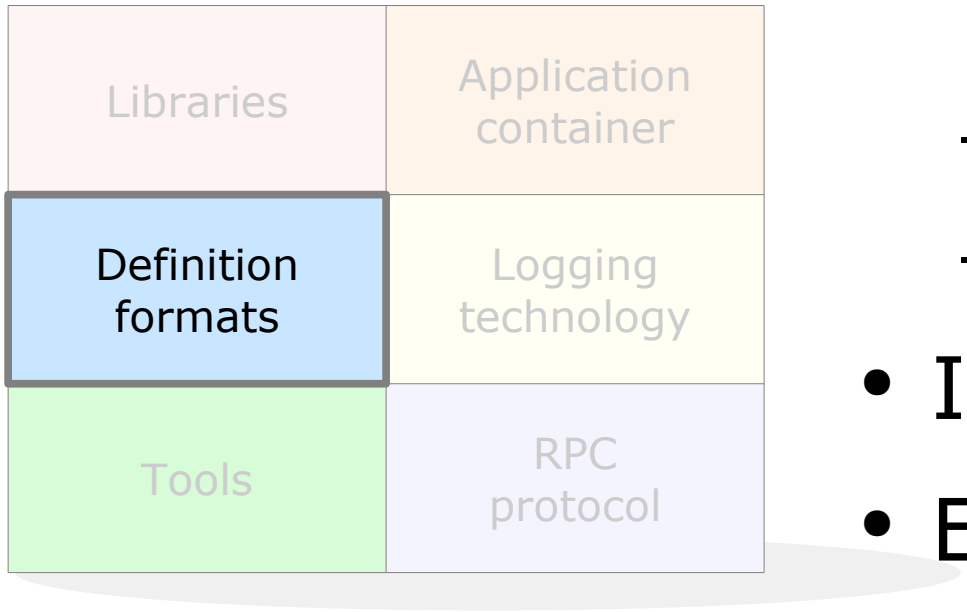
Libraries	Application container
Definition formats	Logging technology
Tools	RPC protocol

Libraries



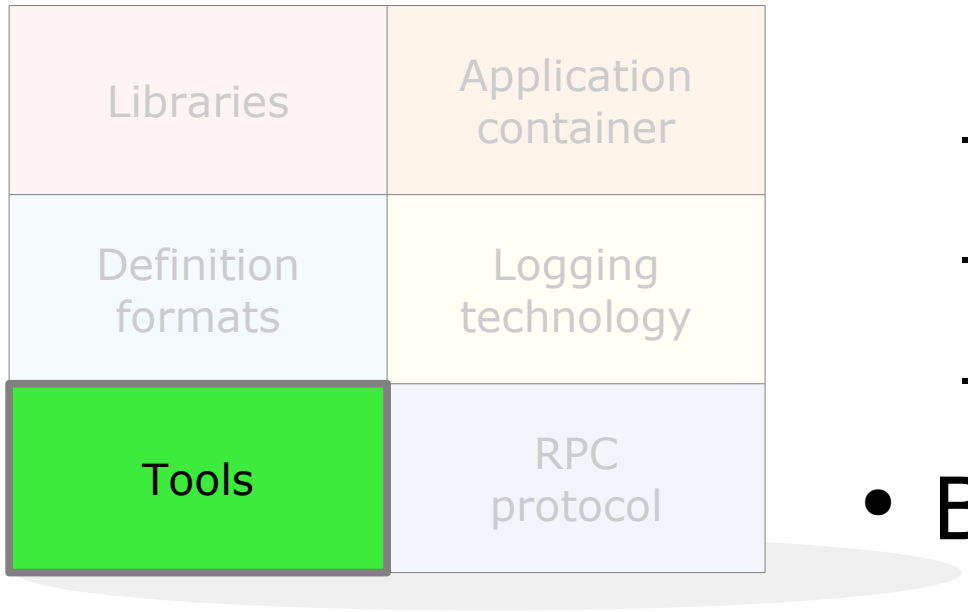
- Client-side 'caller'
  - load-balancing
  - fail-over
  - logging
- Regular expressions
- XML encoding
- ...

Definition formats



- Interface
  - APIs, functions, types, error codes
  - parameters in/out
  - validation rules
- Implementation
- Environments
- Authors
- …

Tools



- Generate from specs:
  - Server- and client-side code
  - Docs (HTML, ODF)
  - Test forms
  - etc.
- Build package (WAR)
- Run/test application
- ...

Specdocs: API index

API index - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://xins.sourceforge.net/demo/specdocs/index.html

API INDEX | OVERVIEW | FUNCTION | TEST FORM | TYPE | RESULT CODE

API index

This project defines the following API specifications:

API	Description	Function count
myproject	My first XINS project	1

Version 1.8. Generated using XINS 1.3.0-rc2.

Done 8.410s

Specdocs: API overview

API overview: myproject - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://xins.sourceforge.net/demo/specdocs/myproject/index.html

API Index | Overview | Function | Test form | Type | Result code

API overview *myproject* (Download)

My first XINS project.

Functions

Function	Version	Status	Description
MyFunction	1.6		A simple function that return a "hello" message to the person

Types

Type	Version	Status	Description
Gender	1.2		A gender
LastName	1.3		Last name of a person

Result codes

Result code	Version	Status	Description
NoVowel	1.3		The name does not contain any vowels

API Owner

No API Owner has been assigned to this API.

Done 2.794s

Specdocs: Function

MyFunction - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://xins.sourceforge.net/demo/specdocs/myproject/MyFunction.html

API INDEX | OVERVIEW | FUNCTION | TEST FORM | TYPE | RESULT CODE

Function *MyFunction*

A simple function that return a "hello" message to the person.

Input section

Input parameters

Parameter	Type	Description	Required
gender	Gender	The gender of the person.	yes
personLastName	LastName	The last name of the person.	yes

Data section

This function defines no input data section.

Output section

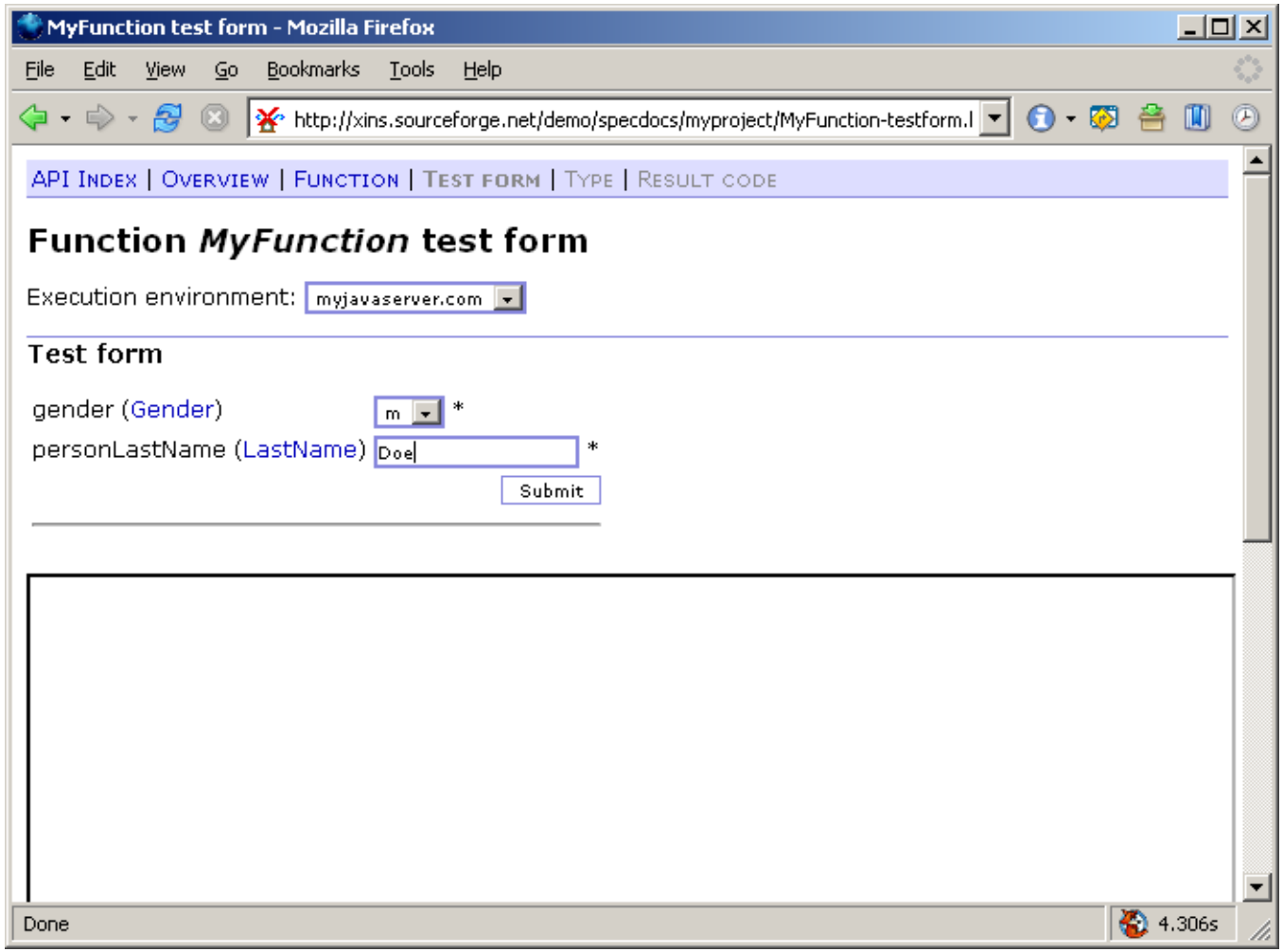
Result codes

A result code is returned when an error occurs during the execution of the implementation.

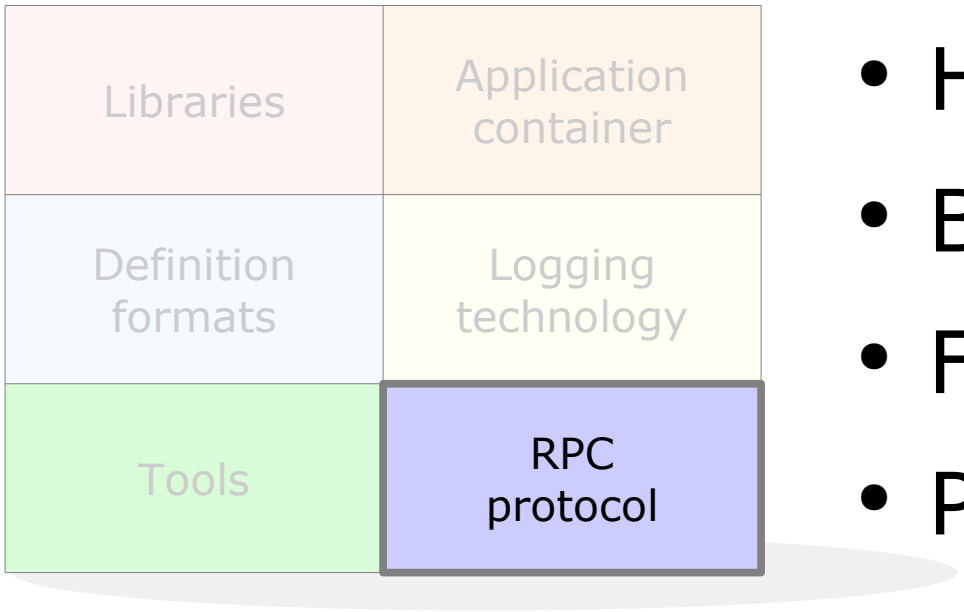
Name	Description
_DisabledFunction	The function is currently disabled.

Done 1.271s

Test form


MyFunction test form - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://xins.sourceforge.net/demo/specdocs/myproject/MyFunction-testform.l
API INDEX | OVERVIEW | FUNCTION | TEST FORM | TYPE | RESULT CODE
Function MyFunction test form
Execution environment: myjavaserver.com
Test form
gender (Gender) m *
personLastName (LastName) Doe *
Submit
Done
4.306s

RPC protocol



- "POX-RPC"
- Simple
- HTTP-based
- Browser-compatible
- Function-oriented
- Params in/out
- Error codes
  - various standard codes

POX-RPC call

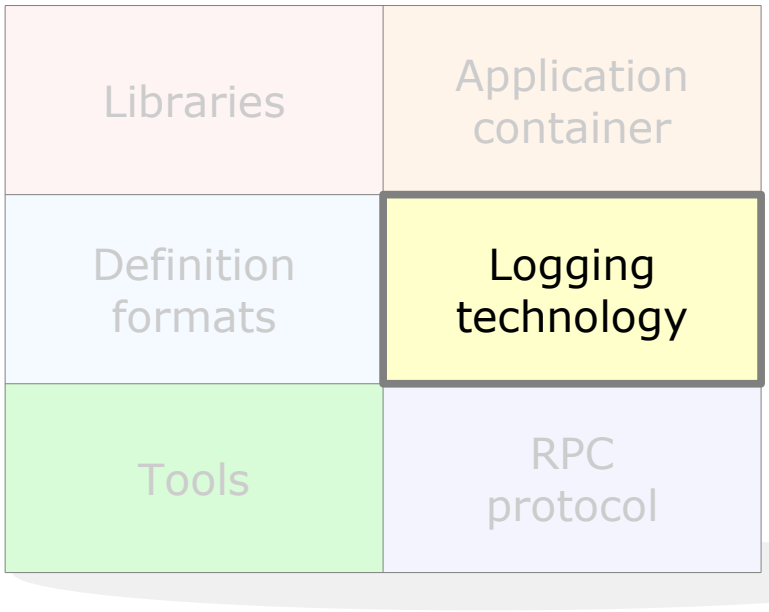
```
GET /?_function=GetCart&cart=1563 HTTP/1.1
Host: test.rest-rpc.org
Accept: text/xml
Connection: close
```

request

response

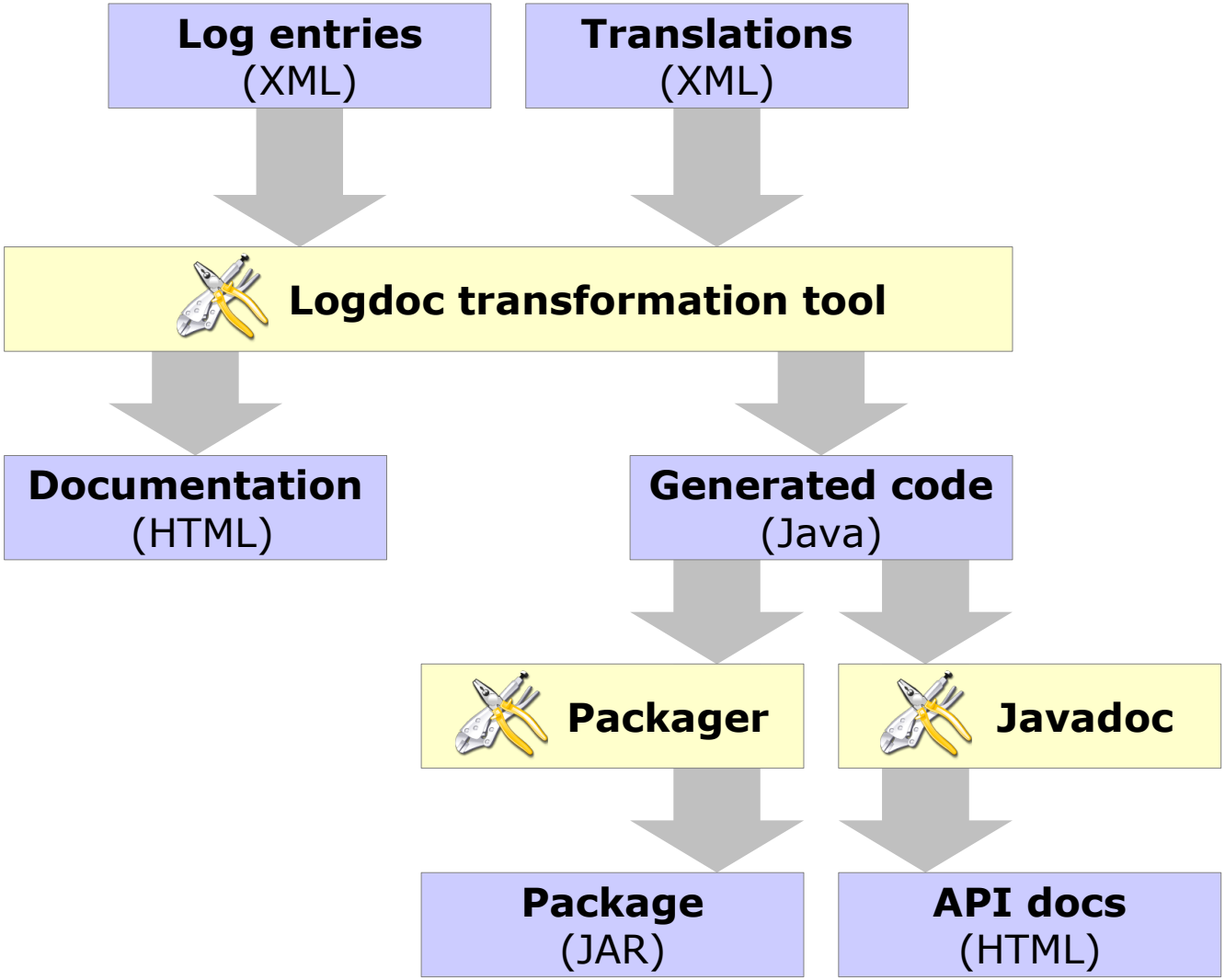
```
HTTP/1.1 200 OK
Content-Length: 114
Content-Type: text/xml
Connection: Close

<result>
   <param name="id">10732</param>
   <param name="remainder">60.5</param>
   <data>
      <item product="8923" price="12" amount="3"/>
      <item product="2108" price="24.5" amount="1"/>
   </data>
</result>
```

Logging technology



- "Logdoc"
- Separates:
  - code
  - log entries
  - translations
  - log levels
- Docs for all log entries
- Filter any log entry

Logdoc


Log entries
(XML)
Translations
(XML)
Logdoc transformation tool
Documentation
(HTML)
Generated code
(Java)
Packager
Javadoc
Package
(JAR)
API docs
(HTML)

Log4J vs Logdoc: Code

Log4J:

```
Logger log = Logger.getLogger("connect.init");
log.error(exception,
     + "Connection "
     + connID
     + " could not be created. Received error "
     + errorID
     + '.');
```

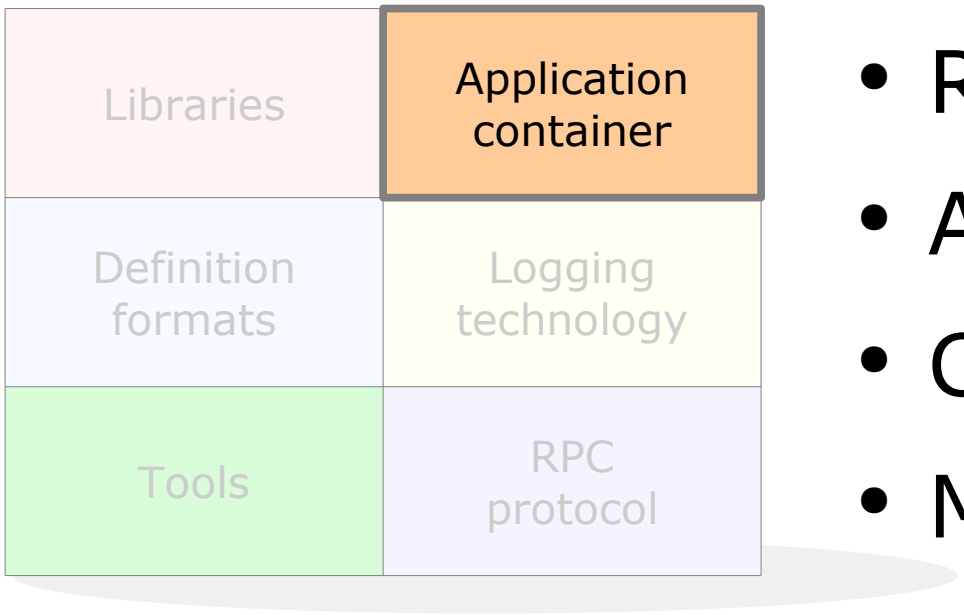
Logdoc:

```
Log.log_30012(exception, connID, errorID);
```

Example: Log definition

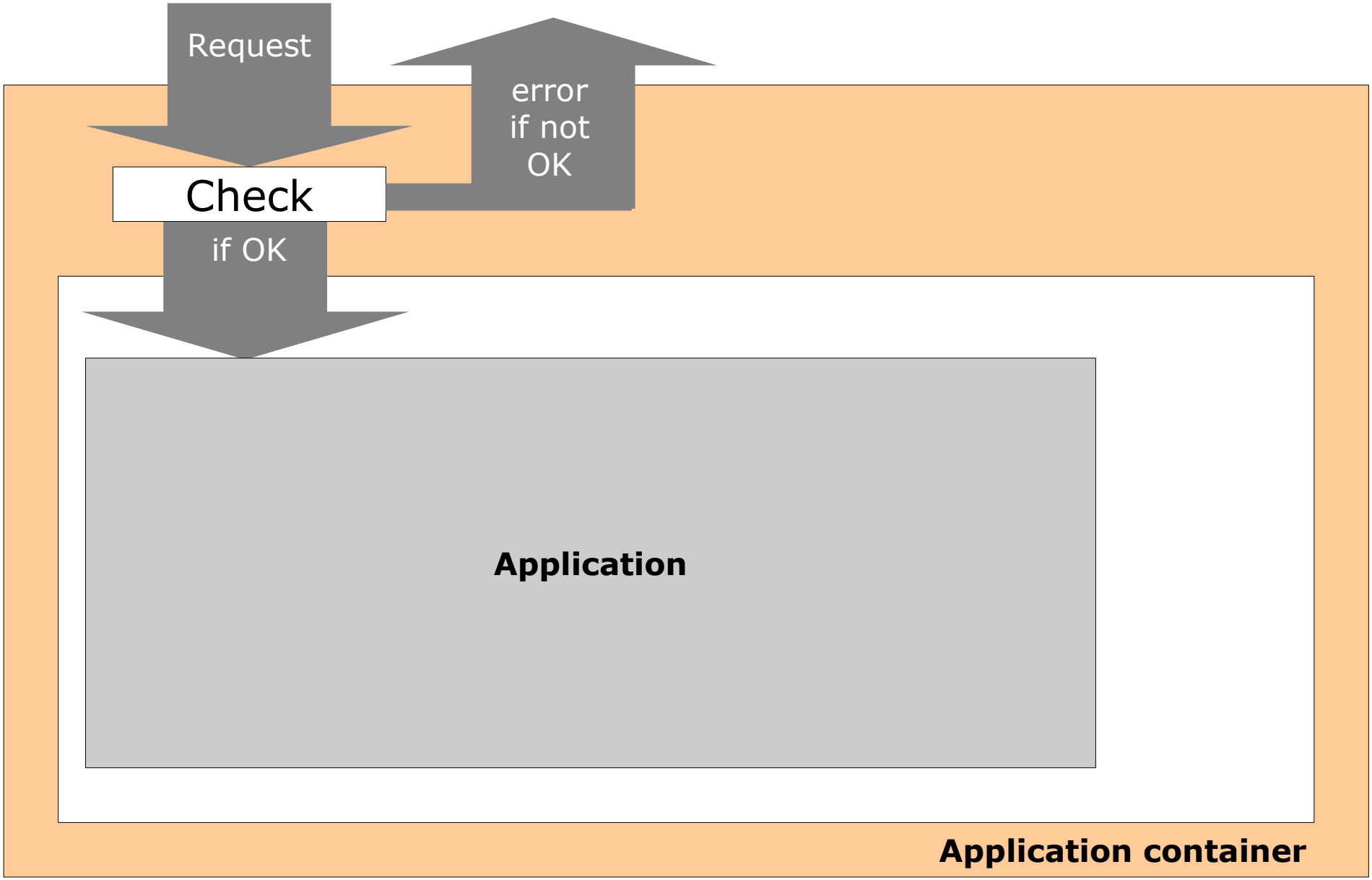
```
<log>
  <translation-bundle locale="en_US"/>

  <group id="conn" name="Connection initialization">

    <entry id="31000" level="INFO">
      <description>Connection succeeded.</description>
      <param name="id"/>
      <param name="num" nullable="false" type="int32"/>
    </entry>

    <entry id="31001" level="ERROR" exception="true">
      <description>Connection failed.</description>
      <param name="id"/>
    </entry>

  </group>
</log>
```

Application container

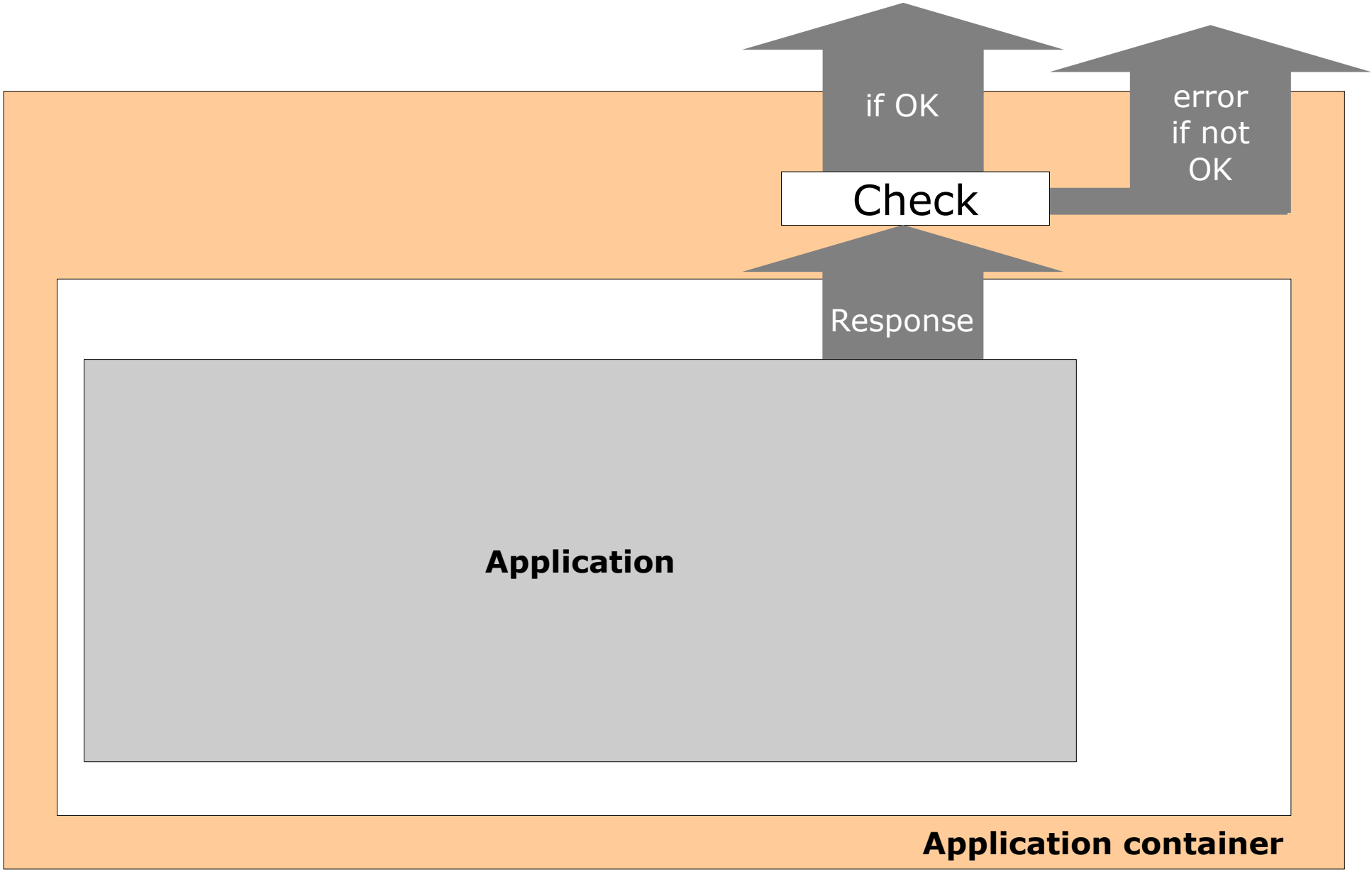


- In servlet container
- Sandbox
- Runtime config
- ACLs
- Calling conventions
- Meta-functions
- Logging
- Context identifiers

Validation of input



Validation of output



Sandbox

- Input checked against specs
  - Invalid? → *_InvalidRequest* error
- Application code is encapsulated
  - Exception thrown? → *_InternalError* error
  - Invalid response? → *_InvalidResponse* error
- Everything is logged

Runtime config

- External to application
  - tested package unchanged to production
- Text file
- Key-value pairs
- Automatically reloaded

Calling conventions

- Abstraction of protocol
- HTTP-based
- Built-in:
  - POX-RPC
  - SOAP
  - XML-RPC
  - XSLT
- Custom:
  - extend Java class: *CustomCallingConvention*

Meta-functions

- *_NoOp*
- *_GetVersion*
- *_GetSettings*
- *_GetStatistics* and *_ResetStatistics*
- *_ReloadProperties*
- *_CheckLinks*
- *_GetFunctionList*
- *_EnableFunction* and *_DisableFunction*

Logging

- Logging in application container:
  - Extensive
  - Completely Logdoc-based
  - Transaction logging
  - Fine-tuned during last 3 years

Example: Start-up log

```
3200 NOTICE Bootstrapping XINS/Java Server Framework 1.4.0-beta3-dev. Servlet
   container: "Orion/2.0.3". JVM: "Sun Microsystems Inc. Java HotSpot(TM) Client VM
   1.5.0_06-b05". OS: "Linux 2.6.12-gentoo-r9/i386".

3227 WARN   XINS/Java Server Framework 1.4.0-beta3-dev is not a production release.

3212 INFO   Package for "allinone" API, version "1.6", was built on zaphod at
   2006.04.06 14:58:04.891, using XINS 1.4.0-beta3-dev.

3228 WARN   Package was built with XINS 1.4.0-beta3-dev, which is not a production
   release.

3245 INFO   Default calling convention is "_xins-soap".

3225 INFO   XINS/Java Server Framework 1.4.0-beta3-dev is bootstrapped.

3405 INFO   Initializing API.

3429 INFO   Access rule 0 is "allow 127.0.0.1/24 *".

3429 INFO   Access rule 1 is "allow 10.0.0.0/24 *".

3427 INFO   Successfully loaded access rule list with 2 rule(s).

3406 INFO   Initialized API.

3441 INFO   XSLT template cache in the XSLT calling convention is disabled.
```

Transaction logging

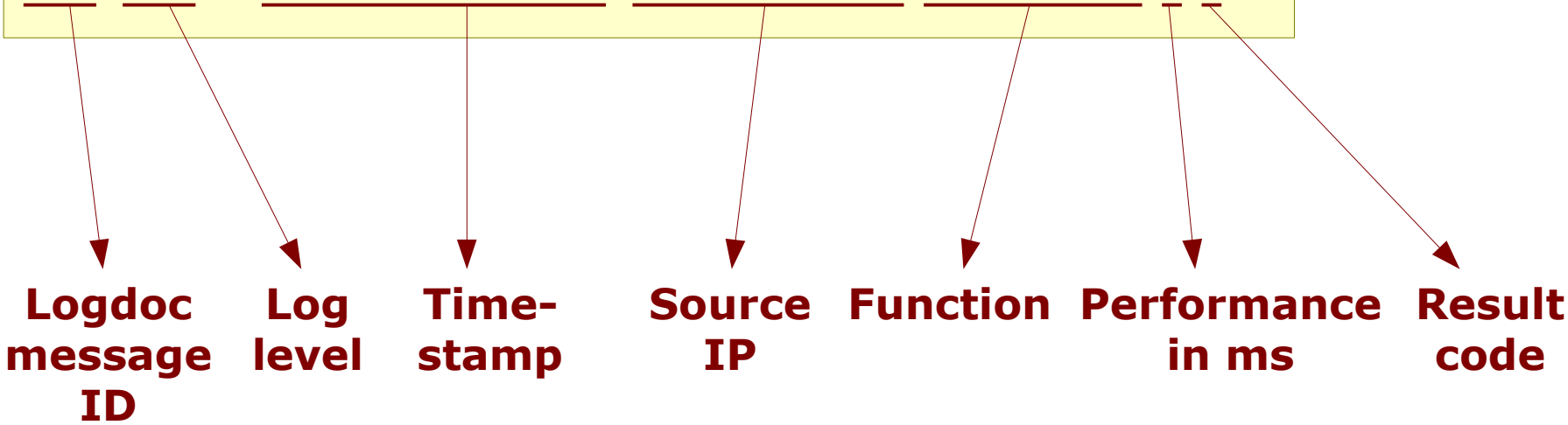
- Logs every incoming request
  - Timestamp
  - Source IP
  - Function name
  - Performance
  - Result code (0 for success)
- Choose:
  - 3540: with params in/out
  - 3541: without

Example: Transaction log

```
3521 INFO   Received HTTP GET request from 194.134.168.69, path is "/", query
   string is "_function=_GetVersion&_convention=_xins-std".

3552 DEBUG  Request from 194.134.168.69 to function _GetVersion does not match
   access rule 0 ("allow 127.0.0.1/24 *").

3552 DEBUG  Request from 194.134.168.69 to function _GetVersion does not match
   access rule 1 ("allow 10.0.0.0/24 *").

3550 DEBUG  Allowing call from 194.134.168.69 to function _GetVersion. Request
   matches access rule 2 ("allow 194.134.168.69/32 *").

3540 INFO   20060407-135658472 194.134.168.69 _GetVersion 1 0
   java.version=1.5.0_06&xins.version=1.4.0-beta3-dev&api.version=1.6

3541 INFO   20060407-135658472 194.134.168.69 _GetVersion 1 0
```

Example: Transaction log

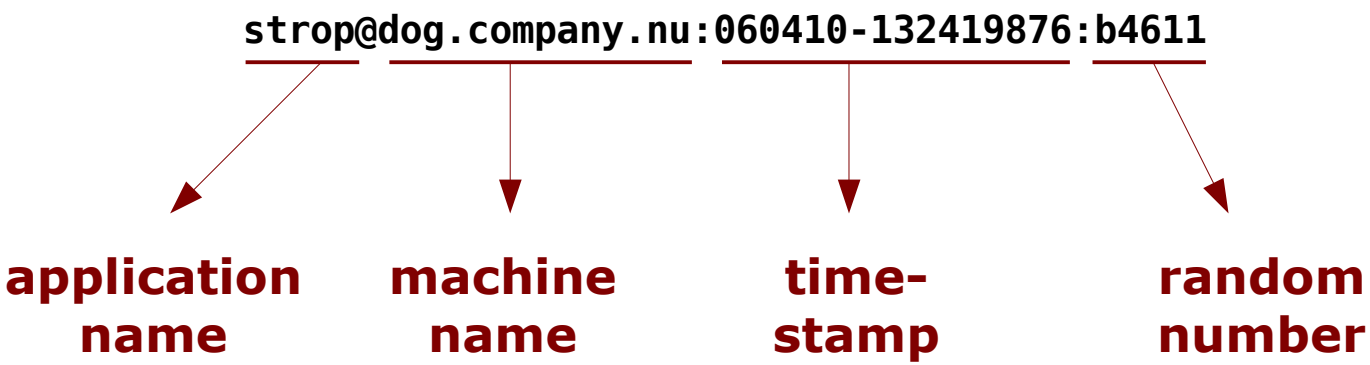
```
3541 INFO   20060407-135658472 194.134.168.69 _GetVersion 1 0
```



Context identifiers

- For diagnosing issues across systems
- Front system generates ID
- ID is passed to all underlying systems
- Systems log ID with selected messages

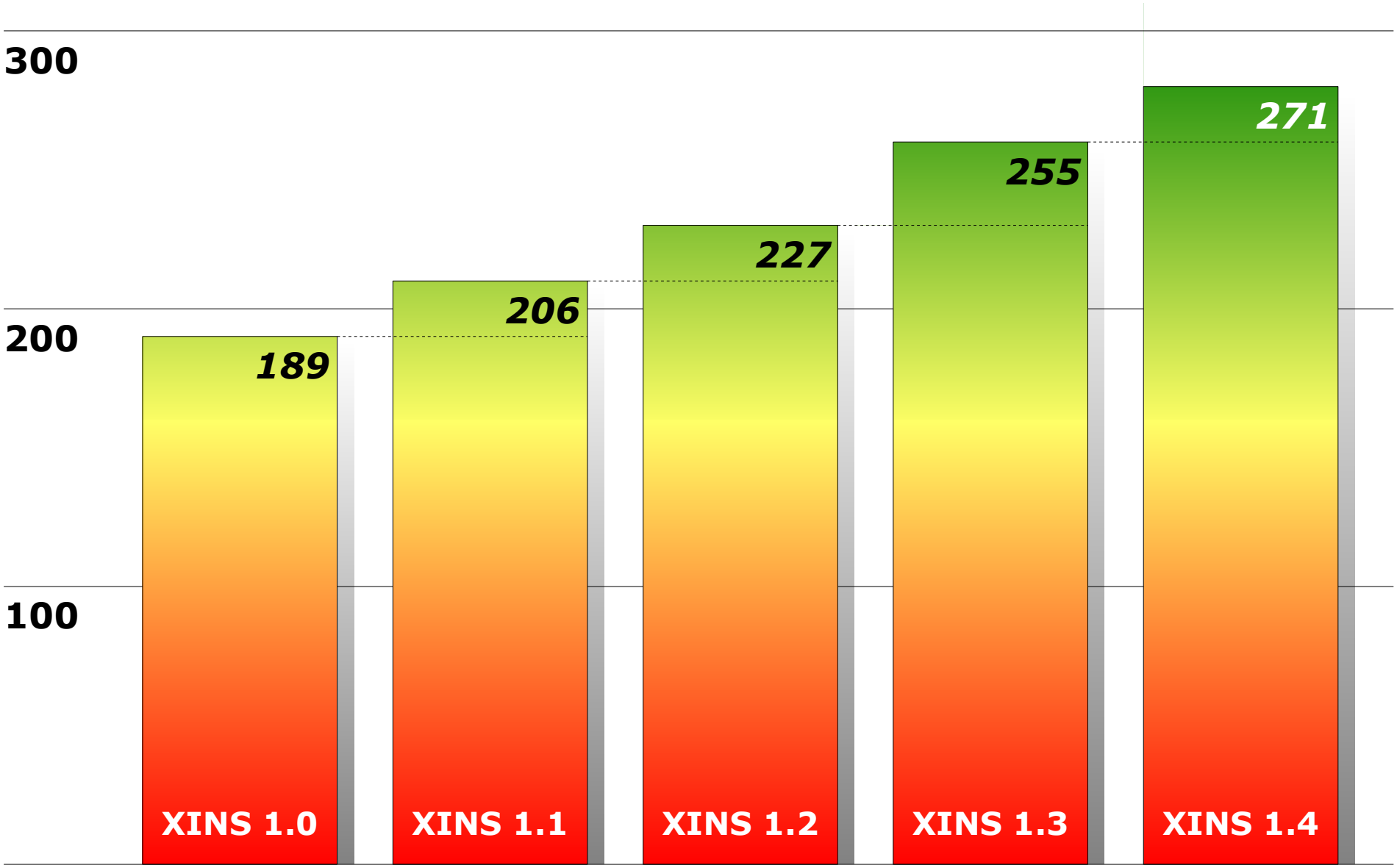
Example: Context identifier



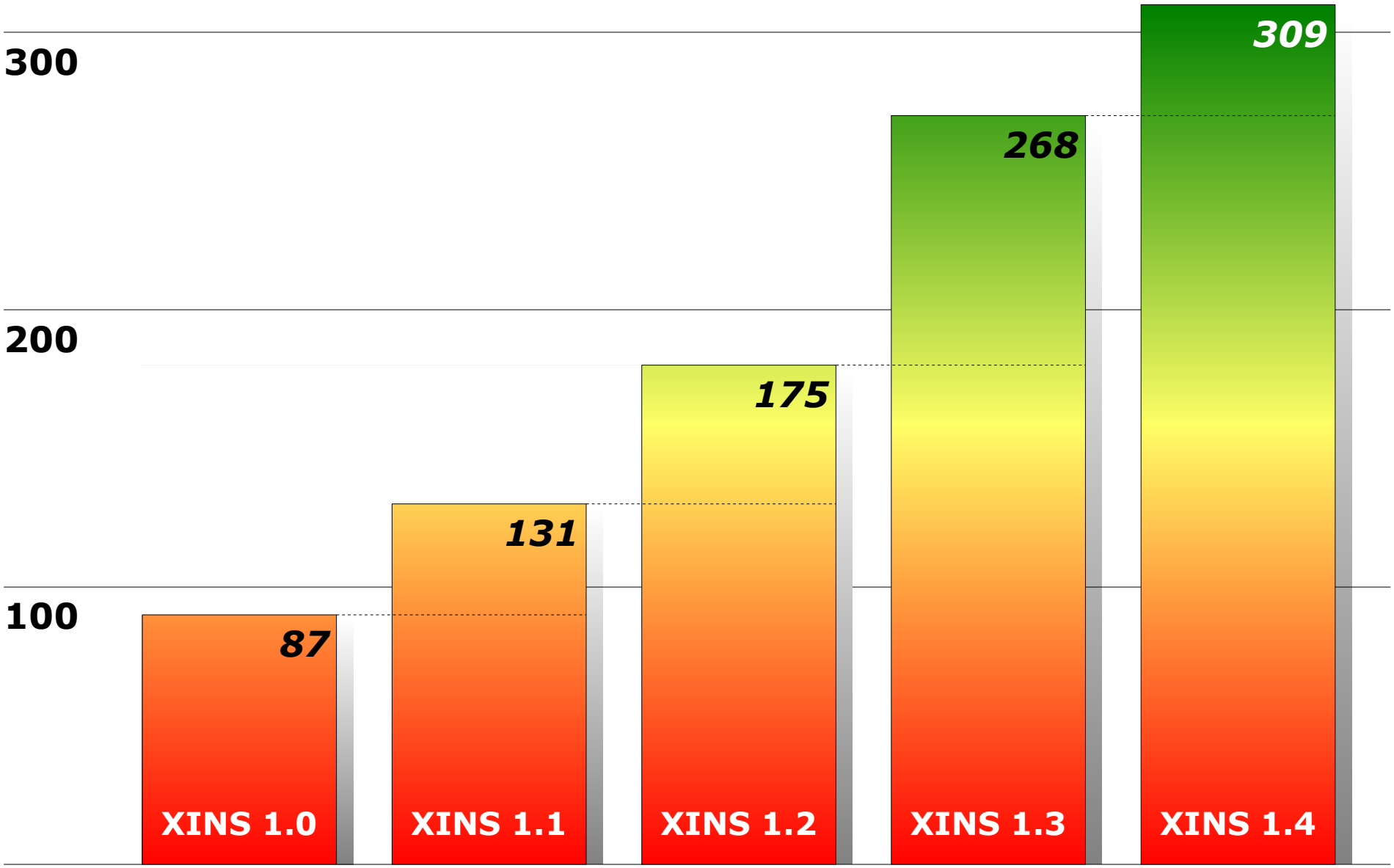
Agenda

- Frameworks
- Fundamentals
- Features
- Qualification

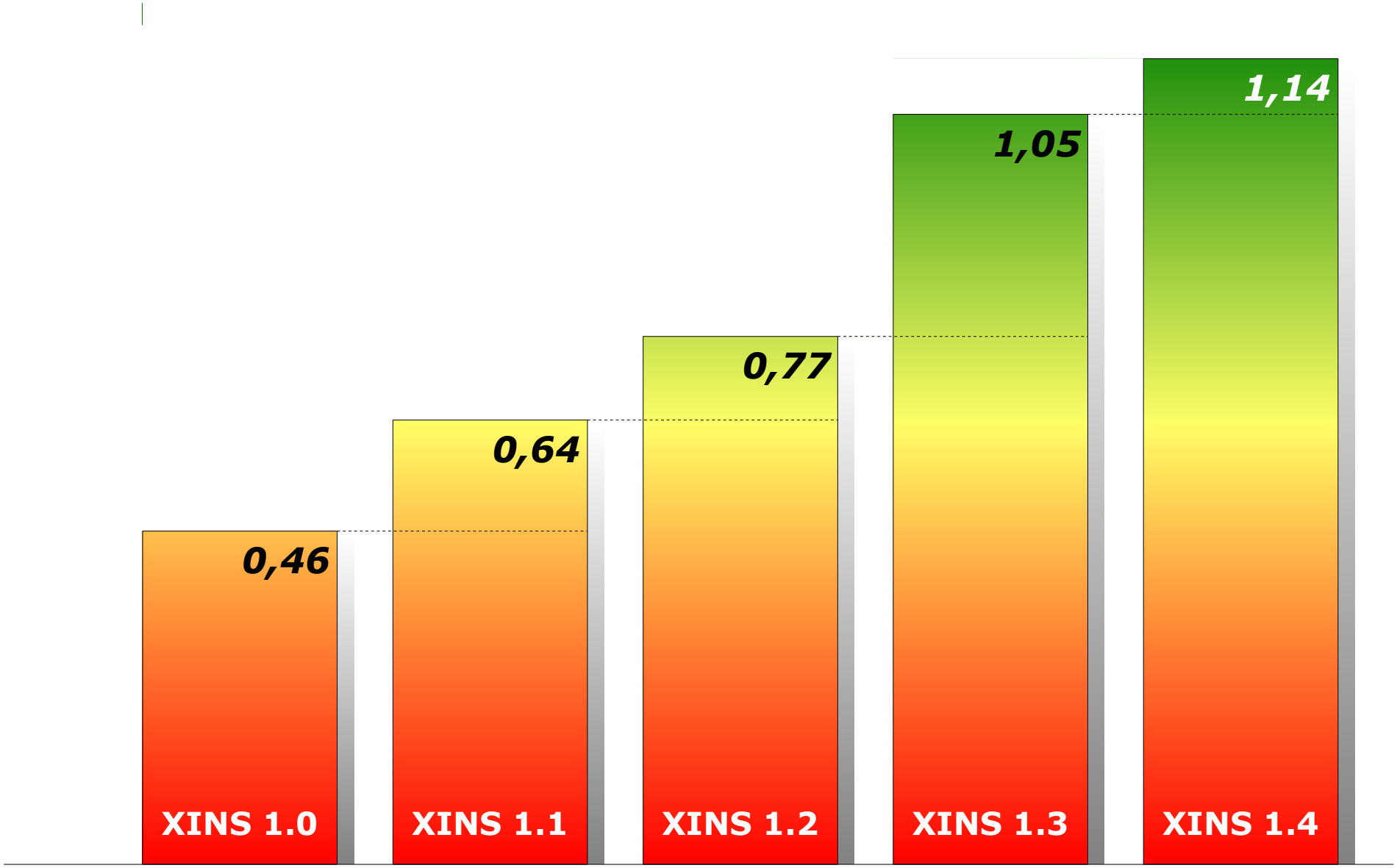
Java and XSLT files



Unit tests


300
200
100
87
131
175
268
309
XINS 1.0
XINS 1.1
XINS 1.2
XINS 1.3
XINS 1.4

Relative: Unit tests per file



Release process

Before a final release (e.g. 1.4.0):

- Alpha releases:
  - Implementation of critical new features
- Beta releases:
  - Testing, documentation, profiling/tuning
- Release candidates:
  - Cool-off period, only bug fixes
  - Testing on various architectures

Testing

- Automatic: 309 unit tests (1.4.0-beta3)
- Various manual tests

- Java: 1.3, 1.4, 1.5, 1.6 EA
- JDK: IBM, Sun
- OS: Solaris, Linux, Win2000, WinXP
- Arch: SPARC, Intel

Conclusion

XINS:

- framework for distributed applications
- high-quality, mature
- easy and feature-rich
  for both Dev and Ops
- actively maintained and supported

XINS

A framework for
distributed applications

Version 4. April 2006
Ernst de Haan