WebServices - Axis

1. WebServices - Axis - Caching Early Draft

Draft Proposal of
Caching Mechanism for Axis
as a JAX-RPC runtime system
Early draft 2
25 Aug, 2003 Toshiyuki Kimura
Apache Axis Committer
toshi@apache.org Copyright © 2003 Toshiyuki Kimura - Page 2 - Table of Contents

Copyright © 2003 Toshiyuki Kimura - Page 3 - 1. Introduction

At the present time, SOAP is actually famous as a transport-independent protocol. But in fact, HTTP protocol is the leading binding of SOAP implementations. In addition, a lot of JAX-RPC implementations use the HTTP POST method for invoking Web Services based on SOAP 1.1 specification. The HTTP POST method is specified in HTTP 1.0 specification (RFC 1945) as uncacheable. Because of this, SOAP is also uncacheable in the present situation.

This document proposes a set of APIs for JAX-RPC and Messaging styles to enable the cache control mechanism in various SOAP implementations.

1.1 Design Goals The goals of this document are as follows:
Specify APIs for supporting Caching Mechanism for both Server side and Client side on the Java platform. Caching Mechanism includes 'Enable/Disable Cache', 'Validate Cache', 'Reload Control', and 'Invalidate Cached Data'. Define protocol-bindings independent of Caching Mechanism for SOAP Messages Support both HTTP POST method and HTTP GET method as cacheable protocol bindings for SOAP Messaging Support interoperability across heterogeneous platforms and environments Support conformance and interoperability requirements that are testable for an implementation of this proposal 1.2 Acknowledgments Satoshi Koyama, Takayuki Nagakura, Kenji Suzuki, and Masashi Takeichi (all from NTT DATA corporation and its business partners) have provided precious technical input to this document. 1.3 Status This document is the early draft version of the proposal. 1.4 Notational Conventions Diagrams follow the standard UML notation Code snippets are not shown in complete form. Refer to the Java docs for complete and detailed description. Examples are illustrative (non-prescriptive) Copyright © 2003 Toshiyuki Kimura - Page 4 - 2. Caching Mechanism Usecase This chapter describes use cases for the Cashing model in a non-prescriptive manner. Later chapters of this document specify requirements and APIs in a prescriptive manner. 2.1 Weather Forecast Service The following description uses a weather forecast service example to illustrate Caching Mechanism concepts.
2.1.1 Service Description The WeatherForecastService endpoint defines and implements the following Java interface. Code Example: Interface of WeatherForecastService package com.example;

```
public interface WeatherForecastProvider extends java.rmi.Remote {
int getChanceOfRain ( String areaSymbol ) throws java.rmi.RemoteException;
   // ...
}
```

The WeatherForecastService has the following features to note: It returns a chance of rain for the specified area. It forecasts chance of rain from observation data which is provided every 3 hours. The chance of rain won't be changed until the next update of observation data, even though a client has frequently access to the service. Copyright © 2003 Toshiyuki Kimura - Page 5 - 2.1.2 Service Use A service client uses a JAX-RPC service by invoking remote methods on a service endpoint. The following diagram shows how a service client uses JAX-RPC runtime and Caching Mechanism.

2.2 Caching Mechanisms This section describes an overview of Caching Mechanisms. 2.2.1 Service Client The following use case diagram shows how a client-side JAX-RPC runtime system uses local cached data which is stored by Caching Mechanisms.

Copyright © 2003 Toshiyuki Kimura - Page 6 - 2.2.2 Server Side The diagram shows how a server-side JAX-RPC runtime system informs that a response message is cacheable to the service client.

Copyright © 2003 Toshiyuki Kimura - Page 7 - 3. Requirements This chapter specifies the proposed scope and requirements for the 1.0 version of Caching Mechanisms. These requirements will be described in more depth in the later chapters. R01 Protocol Bindings The goal of this document is to enable support for multiple protocol bindings. SOAP 1.1 specification [1] provides a sample for using SOAP in HTTP protocol [4] with HTTP POST methods. An extended version, SOAP 1.2 [2] adds a scenario by using HTTP GET methods. Note: HTTP 1.0 [3] does not allow caching for HTTP POST methods, however HTTP 1.1 adds some additional rules to enable caching mechanisms for HTTP POST methods. R02 Transport As a minimum this caching mechanism is required to support HTTP 1.1 as the transport for SOAP message. HTTP binding for the SOAP message is based on the SOAP 1.1 specification [1]. 4. Reference [1] W3C Note: SOAP 1.1: http://www.w3c.org./TR/SOAP/ [2] W3C: SOAP 1.2: http://www.w3c.org/TR/soap12/ [3] HTTP 1.0 http://www.w3.org/Protocols/rfc1945/rfc1945 [4] HTTP 1.1: http://www.w3.org/Protocols/rfc2616/rfc2616 Copyright © 2003 Toshiyuki Kimura - Page 8 - 5. Appendix: Prototype of Caching mechanism In order to demonstrate the feasibility of Caching Mechanisms, I made a prototype of a cache controller which provides a local cache on the service client side.

The hatched areas indicate the extended modules for Apache Axis as a typical JAX-RPC implementation. The detail of this implementation (i.e. class diagram, sequence diagram, and Java docs) will be provided after translation and review. 6. Appendix: Sample Application This sample application uses the WeatherForecastService to enable a cacheable scenario. When an end-user sets a location (like as Tokyo, Osaka, or Kyoto) and invokes the service, the system returns chance of rains for the specified area. Additionally, if the request is cached data and the cache is still valid data, the response message will be picked up from the cache repository.

Note: The role of the HTTP Header Handler is to set a HTTP protocol header for cache controls. Copyright © 2003 Toshiyuki Kimura - Page 9 - 7. Appendix: Unresolved issues The following items are to be determined. IDNameDetail (actual state) 1Dynamic cache-control APIThe current version only provides static configurations with server-config.wsdd. 2SwA (SOAP Messages with Attachments)The current version doesn't support caching a response which has attachment parts. 3Protocol-bindings independent Caching MechanismThe current version is a prototype for HTTP bindings. To create a protocol-bindings independent Caching Mechanism, the following needs to be done; Specify a standard SOAP Message header to cache Implement a SOAP Message setter on server side Implement a SOAP Message parser on client side Copyright © 2003 Toshiyuki Kimura - Page 10 - Sun, Sun Microsystems, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. All other product names mentioned herein are trademarks of their respective owners.