Web Services Globalization Model

February 2003

By Xiao Hui Zhu
IBM SWG Group

Preface

Abstract

Globalization is the proper design and execution of systems, software, services, and procedures so that one instance of software, executing on a single server or end-user machine, can process multilingual data and present culturally correct information (e.g., collation, date, and number formats). Globalization has already become a consideration for Web applications. Web Services, as the open standards to enable dynamic e-business, has stronger global reach cause it runs over networks and involves various participants all over the world.

This paper starts from the base elements of globalization architecture, and then applies them to the Web Services architecture. To strengthen the ideas, some examples are raised in the paper, together with some references for further information.

Target Audience

Since, this paper addresses globalization requirement in the Web Services architecture and introduces the corresponding high-level architecture that is built on the globalization architecture, it is a good starting point for readers who are interested in providing globalization solutions in a Web Services architecture implementation.

Comments

Welcome your comments, and please send to zhuxiaoh@cn.ibm.com. Thanks in advance!

Globalization Architecture

Globalization, as the design and execution of a globalized system, is not a feature. It is an architecture. To succeed in the global marketplace, IBM must define the architecture[1] required to support global e-business, and focus on the enabling technology to provide the key capabilities for globalization.

There are many benefits derived from following the globalization architecture, including:

1. A single server can support applications in multiple languages, thus reducing the cost and time needed to develop, deploy and deliver applications worldwide.
2. The server deployment can be designed based on load levels and resources rather than language support requirements.

[1] A set of architectural elements for global e-business, described in the document titled as "globalization architecture imperatives" by globalization architecture and technology team.

3. The same version and patch level of a product can be used on servers throughout the world, thereby reducing the cost of support, maintenance, and training.
4. Each product will handle the data (e.g. collation, date/time format) in a consistent manner in compliance with established industry standards, so the solution built on those products can assure consistent handling.

The IBM e-business globalization architecture has five interrelated base requirements, as illustrated in Figure 1.

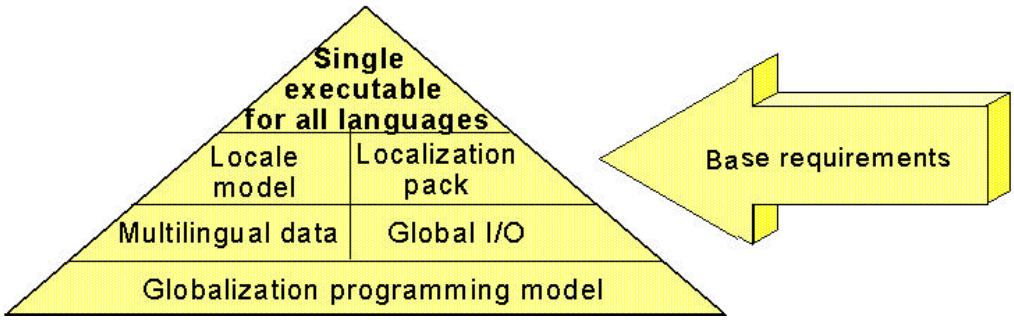

Figure 1 IBM globalization architecture

Single Executable

Single executable means one program code base supports all languages, countries or regions. Software programs must be developed that allows the single executable handles cultural needs of all supported countries/regions. In this way, the cultural and language-independent program code base calls cultural and language-dependent information at run time, thus greatly reduces the cost and efforts invested and simplifies the process as well.

Single executable is the most important requirement of the five. This is the key requirement for ensuring that a global system can be designed, built, and maintained efficiently and correctly.

Locale Model

Since *Single executable* requires one single program code base support all languages, the program code base needs to know which language it shall serve. This information comes the concept of *locale*. In English, the word "*locale*" means a place where something happens or has happened. In globalization, generally speaking, it means language and country/region[2].

[2] For example, en_US stands for English in United States, ja_JP stands for Japanese in Japan, and fr_FR stands for French in France. Sometimes, more specific information is required. For example, in Germany, there are at least two ways of sorting textual data. The de_DE can not tell the differences. This is often done by 1) using a locale variant, e.g. current

Figure 2 shows how *locale model* working with single executable to provide globalized solution through a simple typical scenario:

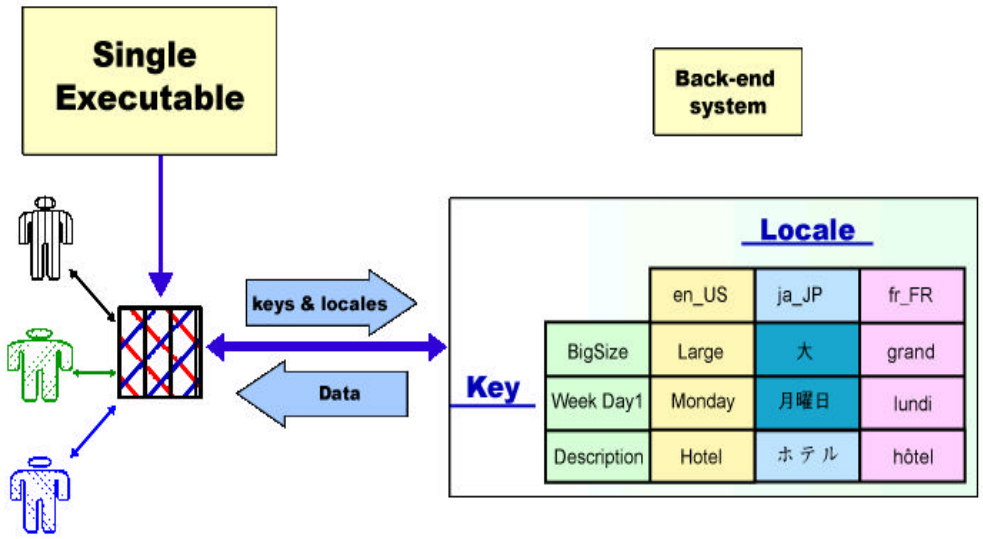

Figure 2 Single executable and locale model

- There are three customers. They're from United States, Japan and France, and their preferred languages are English (in United States), Japanese (in Japan) and French (in France) respectively;
- The application is built on *single executable* base and retrieves back-end data based on customer's cultural preferences;
- The back-end data system is built on *locale model* base, that is a data system categorized by locale information that serves globalized applications;
- The typical data flow is as following:
    - The Application gets the request from the customer along with the *locale* information;
    - The *Single executable* passes the *locale* information to the back-end data system;
    - The back-end data system finds the correct data, and sends back to the *single executable*;
    - The *Single executable* sends the data back to the customer;

  Thus, customer's cultural preferences are well fulfilled.

Multilingual Data

JDK model "de_DE_Traditional 2) using a locale keyword lookup, e.g. OpenI18N locale naming convention de_DE@collation=Traditional.

Because *Single executable* supports all languages, it needs an encoding system[3] to be able to provide a unique number for every character, no matter what the platform, program, or language. Unicode[4] evolved to define this encoding system.
Unicode can represent every character in the world by providing a single consistent "character to number" mapping schema; Unicode enables application to provide multilingual support.

Localization Pack

Since *Single Executable* means to have one and only one executable for multiple locales, there must be a standardized approach (which we call *localization packs*) for working with different sets of locale-specific program data. There are two kinds of *localization packs*, application-dependent[5] and application-independent[6]. The *localization pack manager* is the module that manages the location, loading and accessing of localization pack resources. The Figure 3 shows their relationship.

Figure 3 Localization pack and localization pack manager

A very simple example helps to explain this concept. In an application program, a single key ("msg1") is associated with a single string value ("Hello"). Since we need many language versions, then corresponding localization packs can be created as Figure 4.

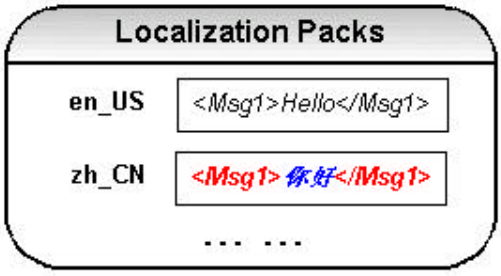

Figure 4 Localization pack for hello message.

Input and Output of Multilingual Data

Globalization also requires the ability to input text in different languages with a keyboard, mouse, or other device and to properly present it in those languages on the

[3] An encoding system is a method of assigning numbers to individual characters so that a computer can process those characters.
[4] Unicode web site: http://www.unicode.org.
[5] Such as menus, dialogs, and other user-interface elements.
[6] Such as collation tables, transliteration rules, and the names of date and time elements.

screen or printer. By using linguistic services, a more human-friendly interface such as speech input can be enabled for the end user. Generally, these functions will be supported by the operating system. Usually, the application does not need to care about this directly.

Web Services Globalization Model

Web Services are the open standard that makes dynamic e-business possible, just as TCP/IP did for Internet and HTTP did for the World Wide Web. Dynamic e-business applications using Web services will be running over open networks and have global reach. Web Services need to be able to support global dynamic e-business. To satisfy this requirement, the base elements of globalization architecture that were introduced in the previous chapter must be applied to Web services.

Web Services is changing rapidly, not only the standards, but also the technologies. This section only takes a brief look at the base architecture.

The globalization in Web Services looks like Figure 5. Universal Description, Discovery, and Integration (UDDI), as a standards-based specification for describing and enabling discovery of Web services, provides a multilingual business description through attribute "xml: lang= …. ". Web Services Description Language (WSDL), as language used to describe service interfaces, is globalization-independent cause it just deals with physical binding for service provider and service requester. The SOAP [7] protocol is used for sending and receiving data in a Web Services system, and it uses XML for carrying data, which is all represented in Unicode. To run globalized business in a Web Services system, both service provider and service requester need to enable globalization following IBM base globalization architecture illustrated in Figure 1 IBM globalization architecture.

[7] Defined by W3C, SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It can be reached at: http://www.w3.org/TR/soap12-part1/

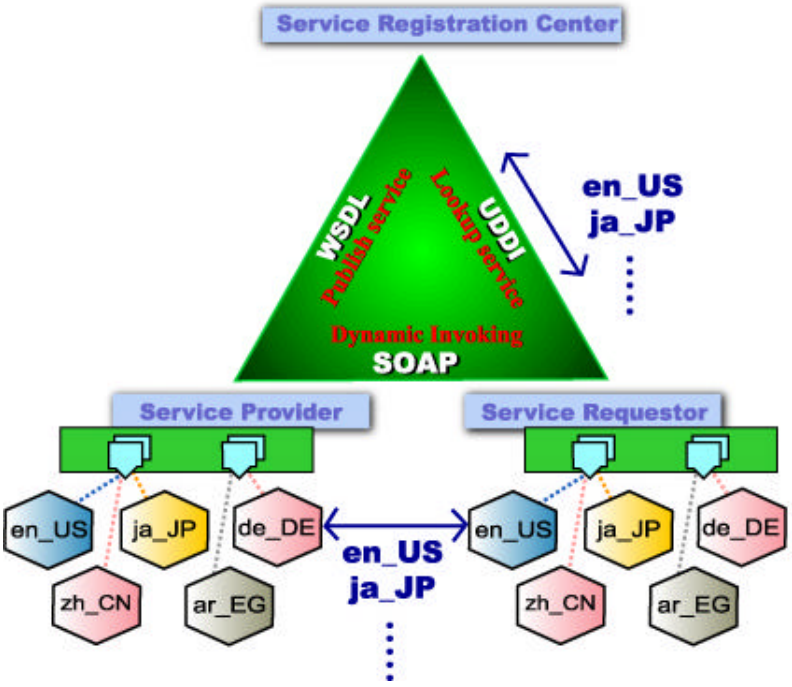

Figure 5 Globalization Web Services Architecture

The base elements for a global Web Services system are the same as those for a global application.

Single Executable

Web Services system contains many programs running on many machines. To have a global Web Services deployment, each program in the system, including the implementation of the service itself, needs to strictly follow single executable policy. Just as single executable is the foundation for a global application, it is also the foundation for a global Web Services system.

Locale Model

In the Globalized Web Services architecture, this requirement needs the most focus. To ensure a workable locale model for a Web Services system, the way of transmitting end-user locale [8] information must be consistent and standardized[9], so it can go through a series of service providers correctly.

---

[8] The definition of locale follows RFC1766, an internet standard which defines the tag for the identification of languages. It follows ISO 3166 for country code and ISO 639 for language code.

[9] Now IBM has started to work towards putting end-user locale definitions on SOAP header.

A scenario maybe this, a travel agent service gets an end-user request, then it contacts the airline service for that end-user, the airline service must return a reservation according to that end-user's cultural preferences, such as date & time representations, currencies representations, measurement representations, etc.

There are different implementation approaches for implementing Web Services in global environment, however, the basic working pattern is similar:

- Publish a multilingual business using UDDI
  - Define a multilingual business
  - Publish a multilingual business
  - Find a multilingual business
- Implement a multilingual business using SOAP
  - Prepare the multilingual environment
  - Implement multilingual Web Services
    - Service provider must provide an interface with locale information in order to support a multilingual business operation [10]
    - Service requester typically:
      - Determines the user's preferred language.
      - Calls a multilingual supported service provider and passes the user's language preference.
      - Returns the result in the specified language to the user.

Multilingual Data

SOAP uses XML for carrying on data, so it is platform-independe nt and programming-language -independent. Therefore, the exchange data in Web services is all represented as XML, and XML is all represented in Unicode that is the key to multilingual data. This means, multilingual support can be well guaranteed by standards.

XML also has tags, such as 'lang', to label certain field(s) to indicate language settings for the linguistically meaningful information, such as descriptions. So, Web Services actually re-enforce a global data representation. Figure 6 Multilingual business description shows an example:

[10] There is no current standard in SOAP for communicating locale/language information. We should switch to the standard as soon as it becomes available.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <businessEntities>
  - <businessEntity name="Airline 1" businessKey="" authorizedName="" Operator="">
    - <discoveryURLs>
        <discoveryURL useType="" text="" />
      </discoveryURLs>
      <name />
      <description lang="en_US" text="Milky Way Co., Ltd is a Hongkong based airline company providing flights to most of the big cities." />
      <description lang="zh_CN" text="银河有限责任公司总部设在香港，作为一家航空公司，它提供的航线可直达多数大城市。" />
      <description lang="zh_TW" text="銀河航空有限公司，總公司設於香港，是一家提供至多數大城市航線的航空公司。" />
      <description lang="ar_EG" text="شركة ميلكي واي، هي شركة طيران مقرها هونج كونج تقدم رحلات طيران لمعظم البلاد الكبيرة." />
      <description lang="de_DE" text="Milky Way Co., Ltd ist eine Fluggesellschaft mit Sitz in Honkong, die Flüge zu den meisten Großstädten anbietet." />
      <description lang="it_IT" text="La Milky Way Co., Ltd è una compagnia aerea con base a Hongkong che fornisce voli verso molte delle grandi città." />
      <description lang="fr_FR" text="Milky Way Co., Ltd est une compagnie aérienne dont le siège social est à Hong Kong et qui dessert la plupart des grandes villes du monde." />
      <description lang="es_ES" text="Milky Way Co., Ldt., es una compañía aérea con sede en Hong Kong que ofrece vuelos a las principales ciudades." />
      <description lang="iw_IL" text="שביל החלב בע"מ היא חברת תעופה שבסיסה בשנגחאי, המספקת טיסות לרוב הערים הגדולות." />
      <description lang="ja_JP" text="Milky Way Co., Ltd は、香港を拠点としてほとんどの大都市へのフライトを提供する航空会社です。" />
      <description lang="ko_KR" text="Milky Way Co., Ltd는 대부분의 대도시에 항공편을 제공하는 홍콩을 기반으로 하는 항공사입니다." />
      <description lang="pt_BR" text="Milky Way Co., Ltd é uma companhia aérea com base em Hong Kong que fornece vôos para a maioria das grandes cidades." />
```

Figure 6 Multilingual business description

Localization Pack

Just as how the *single executable* requirement applies to all the programs that make up the Web Services architecture, the *localization pack* requirement also applies to all the programs. This element is the same for Web services as for normal applications, see Figure 7 for an airline company who wants to provide multilingual services in Web

Services system.

Figure 7 Localization pack in a service provider

Input and Output of Multilingual Data

There is no special requirement for input and output, if inside a Web Services system, because usually this element is taken by operating system or another program and not directly represented to a user, and it is just the same situation here.

Conclusion

The base requirements of a globalization architecture, single executable, locale model, multilingual data, localization pack, and input and output for multilingual data, can be applied well when considering globalization solutions in the Web Services architecture. Web Services is frequently changing and has very rich domain, this paper is only the foundation for globalization in Web Services.

Related Information

1. Globalization Architecture Imperatives

2. Enabling Web Services for Globalization using Websphere Application Server Version 4.0:
   http://www7b.software.ibm.com/wsdd/library/techarticles/0209_zhu/zhu.html

3. SG24-6851-00 e-Business globalization solution design guide: getting started:
   http://www.redbooks.ibm.com/

4. Unicode C onsortium: http://www.unicode.org/