



IBM Software Group

WebSphere® Process Server V6.0.1
WebSphere® Integration Developer V6.0.1
WebSphere® Enterprise Service Bus V6.0.1

Service Message Object Code Sample



@business on demand.

© 2006 IBM Corporation
Updated May 1, 2006

This presentation provides a Service Message Object coding sample.

Goals

- Provide a Service Message Object (SMO) code example
- Provide an example that has a useful application

The goal of the presentation is to provide an example of code traversing through a Service Message Object. The intention is to show code using an SMO and provide a sample that will potentially have a useful application.

SMO Coding Example – Dump SMO Contents

- This coding example prints the contents of an SMO
- It is useful for several purposes:
 - ▶ Illustration of code that is manipulating an SMO as a DataObject
 - DataObjects contain properties that are named and have a type and value
 - Property types can be a simple types or another DataObject
 - ▶ Provides example of how to programmatically discover the contents of an SMO at runtime
 - Determines the property names, types and values
 - Recursively handles properties which are themselves DataObjects
 - ▶ Useful utility to use during development and debug of mediation flows
- It is coded as an operation for a Custom Mediation
 - ▶ Can easily be added at any point in a flow



The sample provided in this presentation dumps the contents of an SMO to the SystemOut.log file, which is useful for several purposes. First, it illustrates code that is manipulating or traversing through an SMO as a DataObject. DataObjects contain properties, which are named values with a defined type. The type of a property can be a simple type or another DataObject. Secondly, the sample shows you how to programmatically discover the contents of an SMO at runtime. It determines the name, type and value of each of the properties contained in the DataObject. When the property itself is another DataObject, it recursively discovers its contents. Thirdly, the sample is a useful utility to use during the development and debug of mediation flows. The sample is coded as an operation that can be called from a Custom Mediation primitive. In this way, it is possible to add this at any point in a flow to dump the SMO contents.

SMO Coding Example – Dump SMO Contents

Operation called as Custom Mediation receiving the SMO

```
public DataObject dumpSMO(DataObject smoDO) {
    System.out.println("##### Start SMO Dump Utility #####");
    dumpDataObject(smoDO, "|");
    System.out.println("##### End SMO Dump Utility #####");
    return smoDO;
}
```

Operation to recursively dump a DataObject

```
private void dumpDataObject(DataObject inputDO, String indent) {
    if (inputDO != null) {
        System.out.println(indent + "### Start DO Dump ###");
        System.out.println(indent + "Type -> " + inputDO.getType().getName());
        List properties = inputDO.getType().getProperties();
        for (Iterator j = properties.iterator(); j.hasNext(); ) {
            Property p = (Property) j.next();
            System.out.println(indent + "Property: " + p.getName());
            if (DataObject.class.isInstance(inputDO.get(p.getName()))) {
                dumpDataObject((DataObject) inputDO.get(p.getName()), "|----" + indent);
            } else {
                System.out.println(indent + " Type: " + p.getType().getName());
                System.out.println(indent + " Value: " + inputDO.get(p.getName()));
            }
        }
        System.out.println(indent + "### End DO Dump ###");
    }
    return;
}
```

Print the DataObject type

Get list of contained properties

Iterate through the properties

Print the property name

If property is a DataObject recursively dump it otherwise print the property type and value

The top portion of the slide shows the **dumpSMO** operation that can be called from a Custom Mediation primitive. It meets the requirements for use from a Custom Mediation primitive in that it takes a single DataObject as the input parameter and also returns a DataObject. This operation simply writes a line indicating the start of the SMO dump, calls an operation that dumps a DataObject passing it the SMO, prints a line indicating the end of the SMO dump and returns the original SMO unchanged. The routine that dumps a DataObject is in the lower portion of the slide. First it ensures that the DataObject passed in is not null, simply returning to the caller if it is and then prints a line indicating that this is the start of the DataObject dump. DataObjects can have specific subtypes, so the next thing that is done is to determine and print out the type of this DataObject. Since a DataObject is composed of properties, a list of the contained properties is then obtained. An iterator is then used to iterate through the list of properties so that each property can be dumped. For each property, the name of the property is printed and the type is determined. If the property is a DataObject, this operation is recursively called to dump it. If it is not a DataObject, the type and value are dumped. After all the properties are dumped, a line is printed indicating the end of the DataObject dump. Since this operation is used to recursively dump DataObjects, it takes an indent string as a parameter, which enables contained DataObjects to be indented in the dump.

IBM Software Group IBM

SMO Coding Example – Dump SMO Contents

```

O |Property: headers ← /headers
O |----|### Start DO Dump ###
O |----|Type -> HeadersType
O |----|Property: sMHeader
O |----|### Start DO Dump ###
O |----|Type -> SMOHeaderType
O |----|Property: messageUUID
O |----|Type: String
O |----|Value: 30b0b7f3-0801-0000-0080-d71f0aa656d6
O |----|Property: version
O |----|### Start DO Dump ###
O |----|Type -> VersionType
O |----|Property: version
O |----|Type: Integer
O |----|Value: 6
O |----|Property: release
O |----|Type: Integer
O |----|Value: 0
O |----|Property: modification
O |----|Type: Integer
O |----|Value: 1
O |----|Property: transient
O |----|### Start DO Dump ###
O |----|Type -> GetCustInfoTransientContext
O |----|Property: accountLocationID
O |----|Type: String
O |----|Value: 99
O |----|Property: backend
O |----|Type: String
O |----|Value: NEW1
O |----|### End DO Dump #####
O |----|Property: failInfo
O |----|Type: FailInfoType
O |----|Value: null
O |----|### End DO Dump #####

O |Property: body ← /body
O |----|### Start DO Dump ###
O |----|Type -> getCustomerExtendedInfoRequestMsg
O |----|Property: getCustomerExtendedInfo
O |----|### Start DO Dump ###
O |----|Type -> getCustomerExtendedInfo._type
O |----|Property: customerID
O |----|Type: String
O |----|Value: 9999999
O |----|Property: portfolioRequested
O |----|Type: String
O |----|Value:
O |----|### End DO Dump #####
O |----|Property: action
O |----|Type: String
O |----|Value: null
O |----|### End DO Dump #####
O |----|Property: jMSHeader
O |----|Type: JMSHeaderType
O |----|Value: null
O |----|Property: sOAPHeader
O |----|Type: SOAPHeaderType
O |----|Value: {}
O |----|Property: sOAPFaultInfo
O |----|Type: SOAPFaultInfoType
O |----|Value: null
O |----|Property: properties
O |----|Type: PropertyType
O |----|Value: {}
O |----|### End DO Dump #####

O ##### End SMO Dump Utility #####

```

Start of Dump (indicated by arrows pointing to the first '### Start DO Dump ###' line)

End of Dump (indicated by an arrow pointing to the final '##### End SMO Dump Utility #####' line)

Service Message Object Code Sample 5

© 2006 IBM Corporation

This slide shows example output from the SMO dump routine. The output would be one contiguous flow, but is broken into three sections here to fit on a single slide. At the beginning of the dump, you can see that the DataObject type that is dumped is **ServiceMessageObject**. The first property it contains is **context**, which is itself a DataObject, whose type is **ContextType**. The properties of **ContextType** are **correlation**, **transient** and **failInfo**. Notice that the **correlation** property and **failInfo** property are both null, but that the **transient** property is another DataObject of type **GetCustInfoTransientContext**. Take some time to examine the rest of the dump to see how the **headers** and **body** properties of the **ServiceMessageObject** are constructed. From this output you can see how this utility would be helpful during development and debug of mediation flows.

Summary

- Looked at a code sample
 - Dump the contents of an SMO
- Discussed possible application of this sample
- Examined output of this code



In summary , this presentation provided example code that dumps the contents of an SMO and examined the output that it produces. This code could be useful during the development and debug of mediation flows.

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM (logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AlX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2004,2005. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

