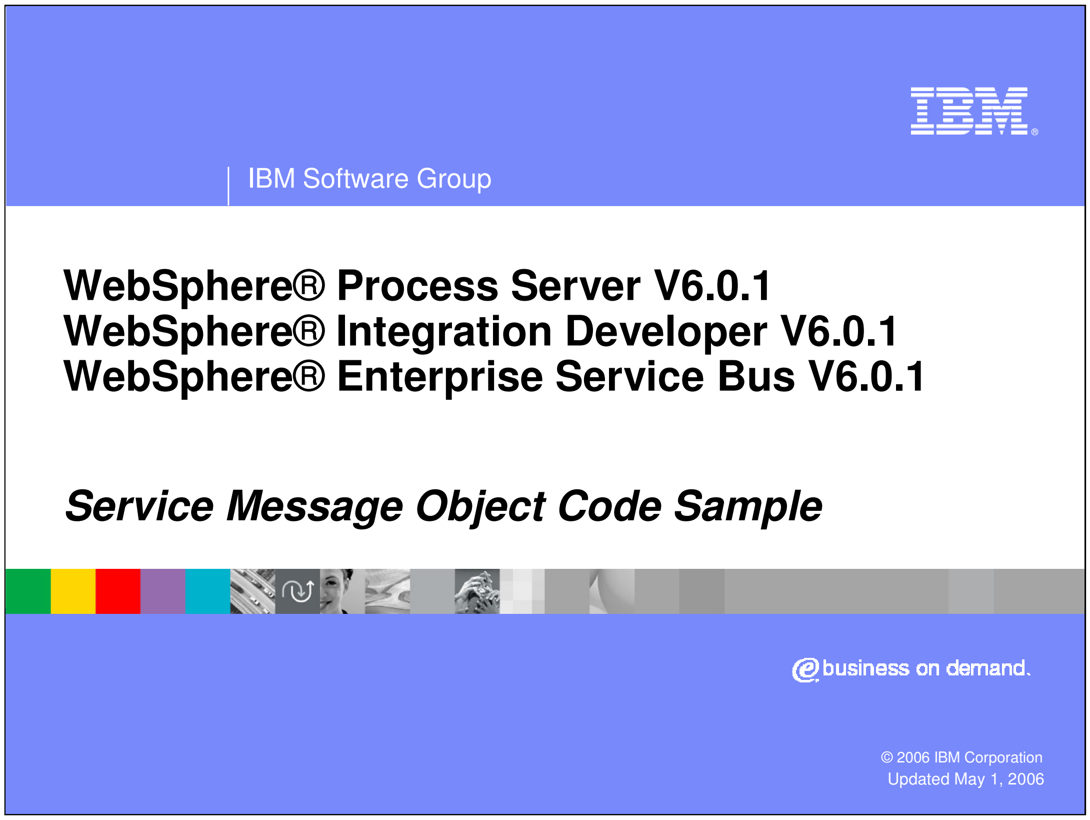


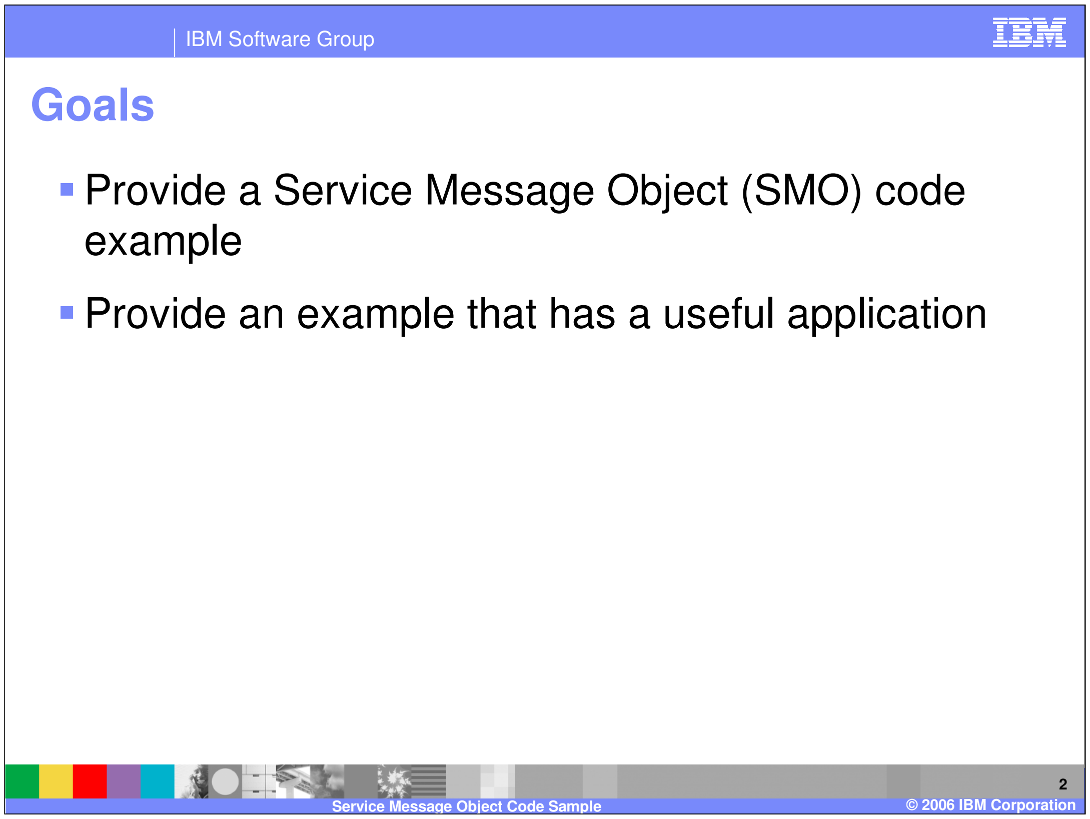
IBM Software Group

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.

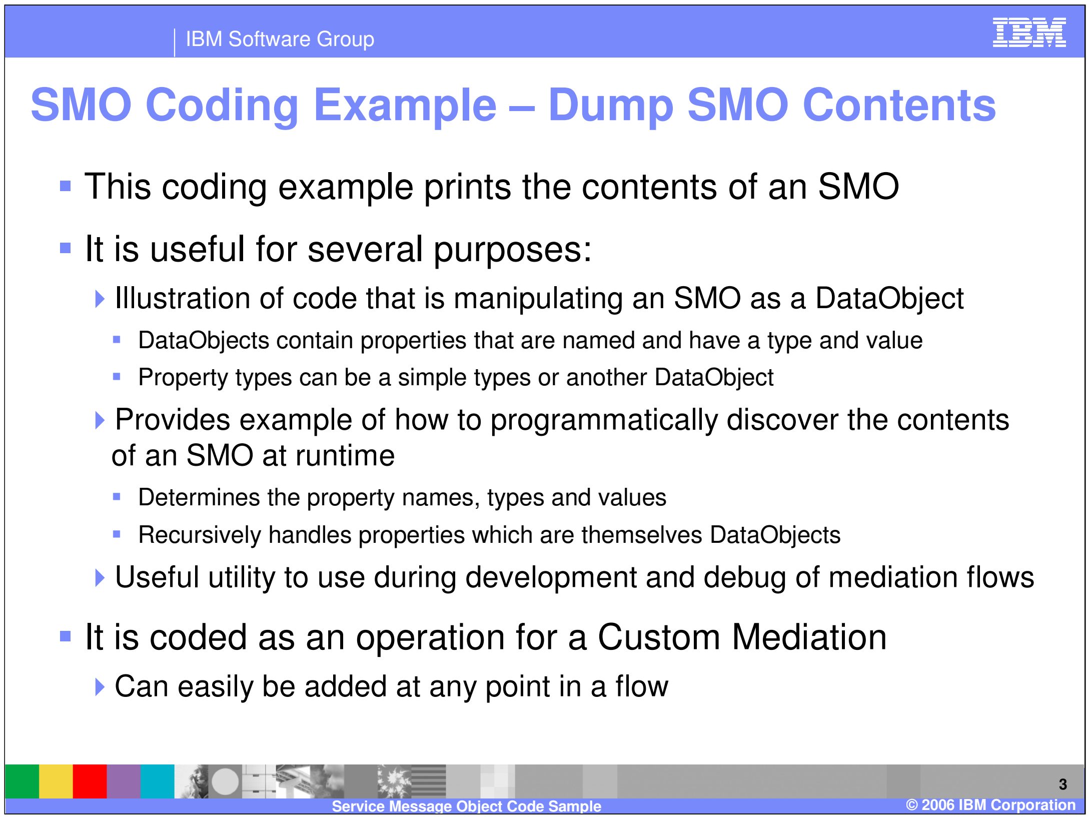
This presentation provides a Service Message Object coding sample.



Goals

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

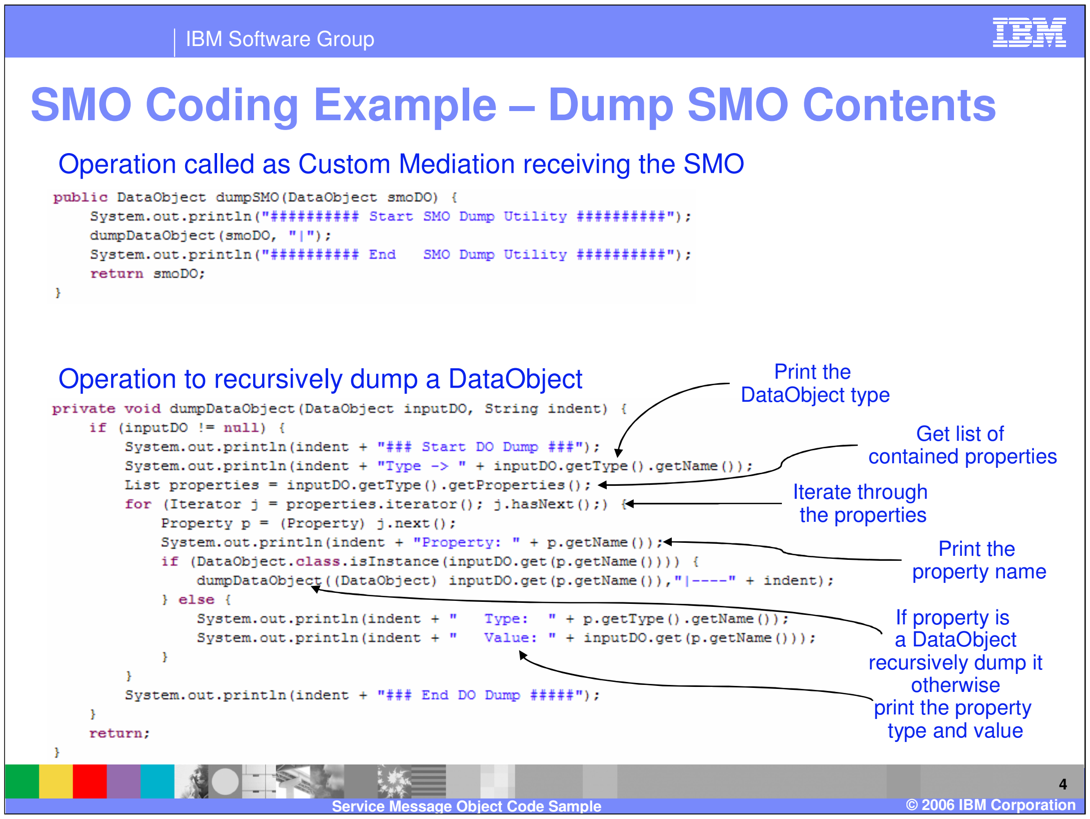
The goal of the presentation is to provide an example of code traversing through a Service Message Object. The intention is to show code using an SMO and provide a sample that will potentially have a useful application.



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

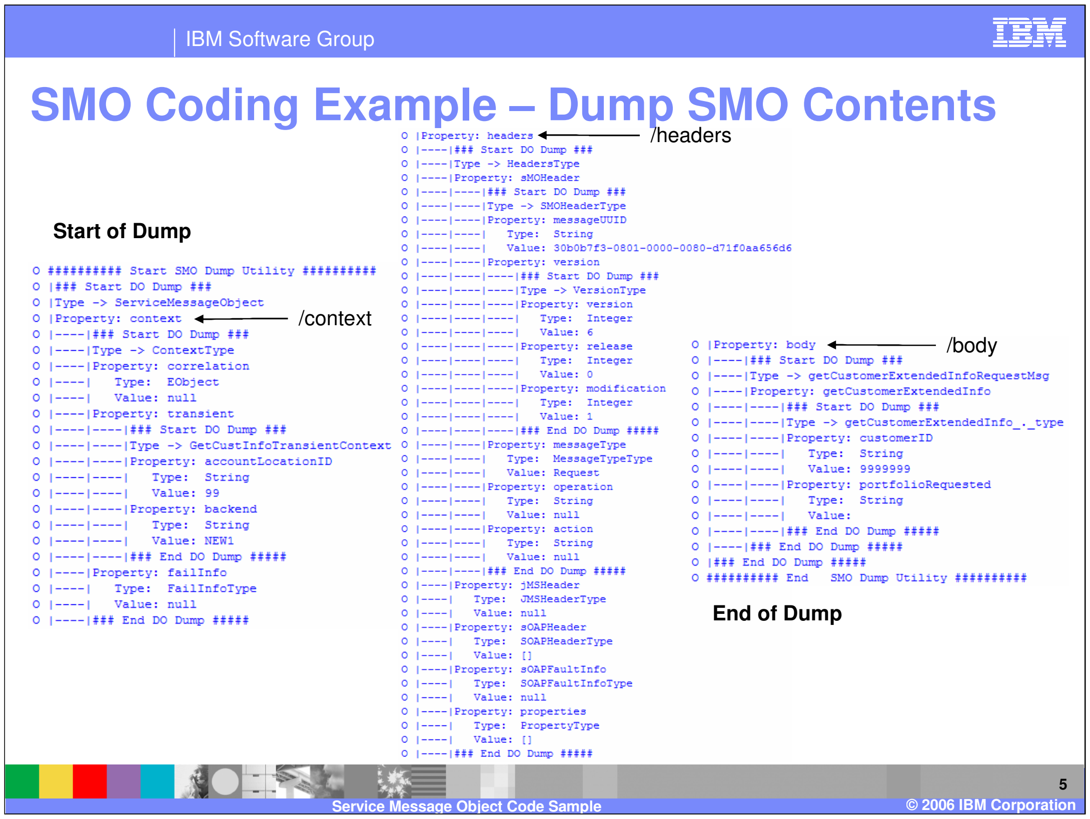
```
public DataObject dumpSMO(DataObject smoDO) {
    System.out.println("########## Start SMO Dump Utility ##########");
    dumpDataObject(smoDO, "|");
    System.out.println("########## End   SMO Dump Utility ##########");
    return smoDO;
}
```

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.



SMO Coding Example – Dump SMO Contents

Start of Dump

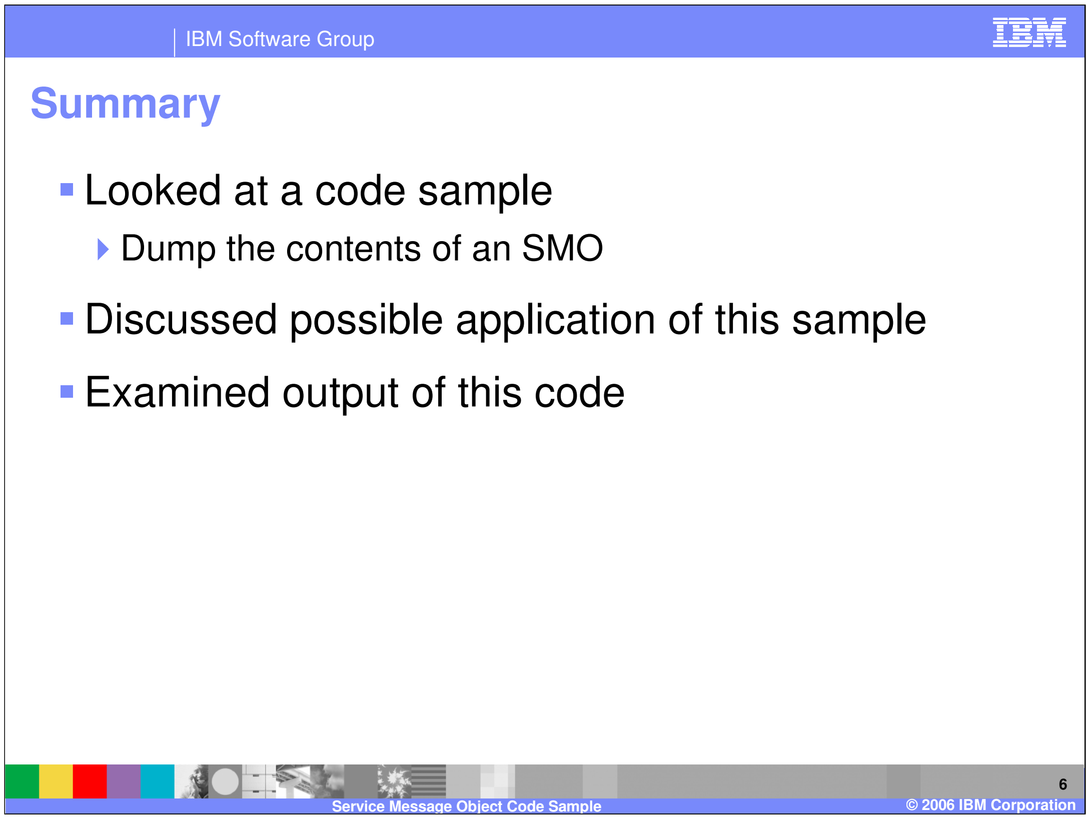
```
O ########## Start SMO Dump Utility ##########
O |### Start DO Dump ###
O |Type -> ServiceMessageObject
O |Property: context                          /context
O |----|### Start DO Dump ###
O |----|Type -> ContextType
O |----|Property: correlation
O |----|   Type:  EObject
O |----|   Value: null
O |----|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: headers                          /headers
O |----|### Start DO Dump ###
O |----|Type -> HeadersType
O |----|Property: sMOHeader
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 |----|----|----|### End DO Dump #####
O |----|----|Property: messageType
O |----|----|   Type:  MessageTypeType
O |----|----|   Value: Request
O |----|----|Property: operation
O |----|----|   Type:  String
O |----|----|   Value: null
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 |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 |----|### End DO Dump #####
O |### End DO Dump #####
O ########## End   SMO Dump Utility ##########
```

End of Dump

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.


IBM Software Group

Summary

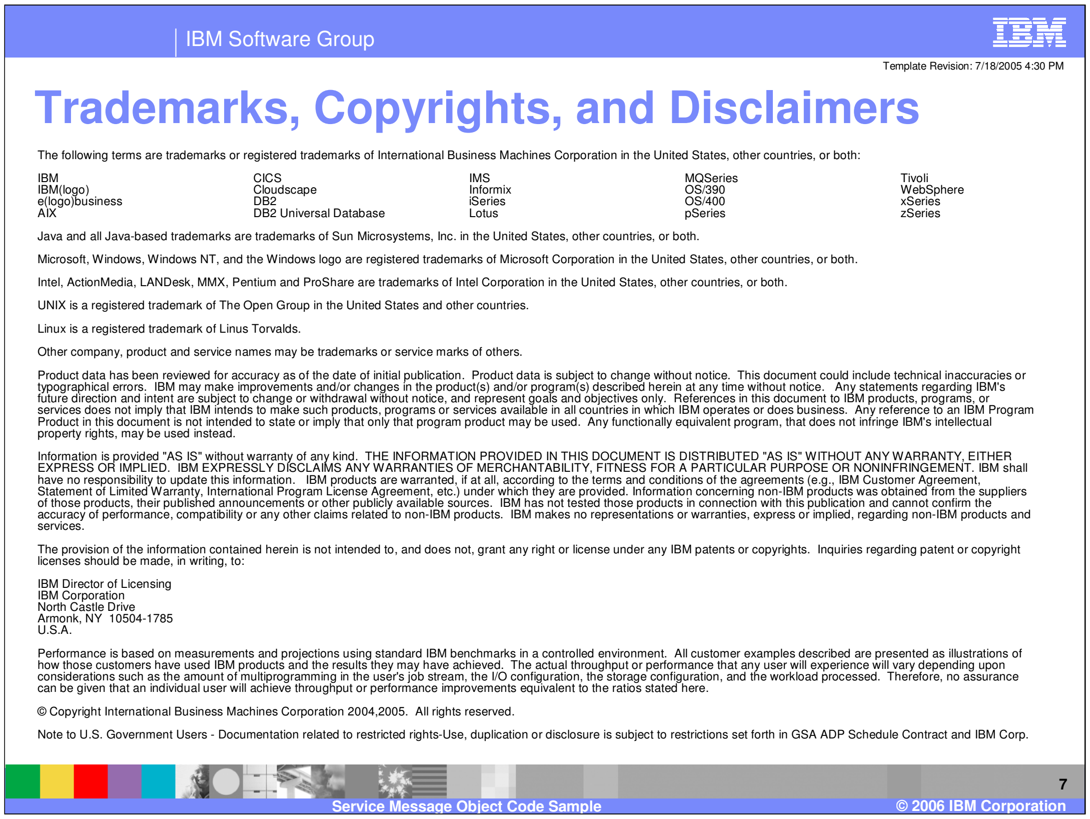
- Looked at a code sample
  - Dump the contents of an SMO
- Discussed possible application of this sample
- Examined output of this code

6

Service Message Object Code Sample

© 2006 IBM Corporation

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.



Template Revision: 7/18/2005 4:30 PM

Trademarks, Copyrights, and Disclaimers