PRODUCT COMPONENT

DEFINITION	
Name	Common Object Request Broker Architecture (CORBA)
Description	CORBA, short for **C**ommon **O**bject **R**equest **B**roker **A**rchitecture, is an architecture that enables pieces of programs, called objects, to communicate with one another regardless of what programming language they were written in or what operating system they're running on. CORBA was developed by an industry consortium known as the Object Management Group.
Rationale	Agencies have a necessity to communicate across different platforms and operating systems.
Benefits	• Ideally suited to use with legacy systems, and to ensure that applications written now will be accessible in the future which allows Missouri State agencies to preserve current investments in legacy systems • Allows state agencies to collaborate across different platforms • Allows state agencies to reuse existing code.

ASSOCIATED ARCHITECTURE LEVELS	
Specify the Domain Name	Interoperability
Specify the Discipline Name	Application Interoperability
Specify the name of the associated Technology Area	Distributed Object Interoperability Artifact

KEYWORDS	
List Keywords	CORBA, object, brokers, Object Management Group (OMG), CORBA compliance, IDL

VENDOR INFORMATION			
Vendor Name	OMG, Inc.	*Website*	http://www.omg.org/technology/documents/formal/corba_iiop.htm
Contact Information	See website.		

POTENTIAL COMPLIANCE SOURCES			
Name	OMG, Inc.	*Website*	http://www.omg.org/technology/documents/formal/corba_iiop.htm
Contact Information	See website.		
Name		*Website*	
Contact Information			

COMPONENT REVIEW	
List Desirable aspects	• With CORBA objects, as well as with the data and databases they interact with, code can be reused with little effort by many different client applications. • Services can be written in many different languages, executed on many different platforms, and accessed by any language with an interface definition language (IDL) mapping. • With IDL, the interface is clearly separated from implementation, and developers can create different implementations based on the same interface. • Supports primitive data types and a wide range of data structures, as parameters • Ideally suited to use with legacy systems, and to ensure that applications written now will be accessible in the future • An easy way to link objects and systems together • May offer greater performance
List Undesirable aspects	• Describing services require the use of an interface definition language (IDL) which must be learned. Implementing or using services require an IDL mapping to your required language - writing one for a language that isn't supported would take a large amount of work. • IDL to language mapping tools create code stubs based on the interface - some tools may not integrate new changes with existing code. • Does not support the transfer of objects, or code. • Not all classes of applications need real-time performance, and speed may be traded off against ease of use for pure Java systems.

Operating System		*Platform*	Platform independent

ASSOCIATED COMPLIANCE COMPONENTS	
Product	
List the Product-specific Compliance Component Names	

Configuration Links	
List the Configuration-specific Compliance Component Names	

COMPONENT CLASSIFICATION			
Provide the Classification	☐ *Emerging* ☒ *Current* ☐ *Twilight* ☐ *Sunset*		
Sunset Date			

COMPONENT SUB-CLASSIFICATION		
Sub-Classification	Date	Additional Sub-Classification Information
☐ *Technology Watch*		
☐ *Variance*		
☐ *Conditional Use*		

RATIONALE FOR COMPONENT CLASSIFICATION			
Document the Rationale for Component Classification			
IMPACT POSITION STATEMENT			
Document the Migration Strategy			
IMPACT POSITION STATEMENT			
Document the Position Statement on Impact			
AGENCIES			
List the Agencies Currently Utilizing this Product			
CURRENT STATUS			
Provide the Current Status	☒ *In Development* ☐ *Under Review* ☒ *Approved* ☐ *Rejected*		
AUDIT TRAIL			
Creation Date	02/08/05	*Date Approved / Rejected*	10/11/05
Reason for Rejection			
Last Date Reviewed		*Last Date Updated*	
Reason for Update			