



---

# *System Overview*

---

Version 2.1  
August 2002

## **System Overview, version 2.1**

Copyright © 2001–2002 Modulant Solutions, Inc. All rights reserved.

August 2002, Version 2.1

**Ownership of Materials.** This manual, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The contents of this manual are furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Modulant. Modulant assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

This manual is protected by copyright and distributed under licenses restricting its use, copying, translation, distribution, and decompilation. Except as permitted by such licenses, no part of this manual may be reproduced in any form by any means without prior written authorization of Modulant. Except as expressly provided herein, Modulant grants no express or implied rights to anyone under any patents, copyrights, trademarks, trade names, or trade secret information with respect to the contents of the manual.

**Ownership of Trademarks.** The trademarks, service marks, product names, company names or logos and other marks displayed in the manual are the property of Modulant Solutions, Inc. or other third parties. Any use of trademarks, service marks, product names, company names or logos, and other marks, including the reproduction, modification, distribution, or republication of same without the prior written permission of the owner is strictly prohibited.

Modulant, the Modulant logo, and Contextia are trademarks of Modulant Solutions, Inc. Other trademarks, service marks, trade names and company logos referenced are the property of their respective owners.

**Disclaimers.** THIS MANUAL IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. FURTHER MODULANT DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

**Notice to U.S. Government Users.** All Modulant products and publications are commercial in nature. The software and documentation are “commercial items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are licensed to U.S. Government end users (A) only as Commercial Items and (B) with only those rights as are granted to all other end users pursuant to the terms and conditions set forth in the Modulant standard commercial agreement for this software. Unpublished rights reserved under the copyright laws of the United States.

---

# Table of Contents

<b>List of Figures</b> . . . . .	<b>vii</b>
<b>Preface</b> . . . . .	<b>ix</b>
Who Should Read this Guide? . . . . .	ix
Conventions Used in this Guide . . . . .	x
What’s in this Guide? . . . . .	xi
Related Documents . . . . .	xi
<b>1 Introducing Information Interoperability</b> . . . . .	<b>1</b>
Enabling Information Interoperability . . . . .	2
Interoperability is About Communication . . . . .	3
The Role of Context . . . . .	6
<b>2 Introducing the CIIM</b> . . . . .	<b>7</b>
Underlying Principles . . . . .	7
The CIIM Framework . . . . .	8
CIIM Architecture . . . . .	9
Data Elements . . . . .	10
Information Units . . . . .	10
Map Domains . . . . .	10
The CIIM Methods . . . . .	11
Context Discovery . . . . .	11
Context Formalization . . . . .	12
Context Accommodation . . . . .	12
<b>3 The Interoperability Process</b> . . . . .	<b>15</b>
Design-time Process: Applying the CIIM . . . . .	15
Develop an Interoperability Strategy . . . . .	16
About Abstract Conceptual Models . . . . .	17
About Application Transaction Sets (ATs) . . . . .	18
Describe Application Data Formats in Data Maps . . . . .	19

Develop Application Schemas . . . . .	19
Create Data Maps to Describe Physical Structures . . . . .	20
Capture Application Context in Context Maps . . . . .	20
About Context Map Files . . . . .	20
The Mapping Process . . . . .	21
Run-time Process: Achieving Information Interoperability . . . . .	23
Configure Interoperability Run . . . . .	23
Perform Interoperability Run . . . . .	24
Retrieve Target Data in Native Format and Verify Results . . . . .	24
<b>4 Modulant Contextia Tools and Components . . . . .</b>	<b>25</b>
Design-time Tools: The Interoperability Workbench . . . . .	26
The Data Mapper . . . . .	26
The Context Mapper . . . . .	27
FirstSTEP XG and FirstSTEP EXML . . . . .	28
Run-time Tools: The Interoperability Server and its Client Tools . . . . .	29
The Interoperability Run Console . . . . .	29
The Interoperability CL . . . . .	30
The Interoperability Server Administrator . . . . .	31
The Interoperability API . . . . .	33
Inside the Interoperability Server . . . . .	33
Server Components . . . . .	33
Flow of an Interoperability Run . . . . .	36
<b>5 An Interoperability Example . . . . .</b>	<b>39</b>
Introducing the Sample Applications . . . . .	39
Developing a Mapping Strategy . . . . .	40
Creating the Mapping Specifications . . . . .	42
Describing the Data . . . . .	42
Describing the Context . . . . .	45
Performing an Interoperability Run . . . . .	47
<b>6 Architectural Considerations . . . . .</b>	<b>51</b>
Synchronous vs. Asynchronous Interoperability Runs . . . . .	51
Synchronous Operation . . . . .	51
Asynchronous Operation and the Job Queue . . . . .	52
Remote vs. Embedded Servers . . . . .	52
Remote Server Using JMS . . . . .	53
Remote Server Using Web Services and SOAP . . . . .	54
Embedded Server . . . . .	56

<b>Glossary</b> . . . . .	<b>57</b>
<b>Index</b> . . . . .	<b>67</b>



---

# List of Figures

Figure 1: Request for Information Across Related User Communities . . . . .	4
Figure 2: Response to Request for Information. . . . .	5
Figure 3: The CIIM Methods . . . . .	8
Figure 4: Aggregations of Data and Information . . . . .	9
Figure 5: Design-time Process . . . . .	16
Figure 6: Mapping Data Elements to an ACM . . . . .	21
Figure 7: Enabling Information Interoperability . . . . .	23
Figure 8: Data Mapper Main Window . . . . .	27
Figure 9: Context Mapper Main Window . . . . .	28
Figure 10: Interoperability Run Console Main Window . . . . .	30
Figure 11: Interoperability CL Usage Message . . . . .	31
Figure 12: Interoperability Server Administrator Main Page . . . . .	32
Figure 13: The Modulant Contextia Interoperability Server . . . . .	34
Figure 14: Interoperability Flow . . . . .	36
Figure 15: US Tours and Euro Travel . . . . .	40
Figure 16: Mapping Strategy for US Tours and Euro Travel . . . . .	42
Figure 17: Format Conversion to Split start_date into Parts . . . . .	45
Figure 18: The Interoperability API Job Queue . . . . .	52
Figure 19: Using the JMS Conduit . . . . .	54
Figure 20: Using the SOAP Conduit . . . . .	55
Figure 21: Using an Embedded Server . . . . .	56

## List of Figures



---

# Preface

The Modulant Contextia Interoperability Platform is Modulant Solutions, Inc.'s revolutionary toolset for creating information interoperability.

The *System Overview* outlines how the Modulant Contextia platform enables information interoperability, with descriptions of the overall work flow, the individual tools and components that are part of this platform, and examples of how interoperability architects use these tools to make interoperability a reality.

This preface contains the following topics:

- [Who Should Read this Guide?](#)
- [Conventions Used in this Guide](#)
- [What's in this Guide?](#)
- [Related Documents](#)

---

## Who Should Read this Guide?

The audience for the Modulant Contextia Interoperability Platform includes interoperability architects, domain experts, and software developers.

- Interoperability architects describe application data structures in data maps, create context maps from those data structures to an Abstract Conceptual Model, and perform interoperability runs. During this process, interoperability architects retain an overall view of all of the applications in an interoperability environment.
- Domain experts work with interoperability architects to help them understand both the structure and the context of their community's data.
- Java programmers can use the Interoperability API to automate the process of transforming data from one or more data sources to another.

The documentation set is intended to serve as a supplement to the material available in the Modulant training program; for information and a course catalog, visit [www.modulant.com/partners/training.shtml](http://www.modulant.com/partners/training.shtml).

This guide assumes that you are familiar with the following topics:

- Your operating system (Windows or Solaris) and its file system
- Data modeling and XML representation of data

- Your application domain
- Relational database management systems (RDBMSs)

---

## Conventions Used in this Guide

The manuals in the Modulant Contextia Documentation Library use the following typographic conventions:

<b>bold text</b>	File names, XML elements, user interface controls, and language keywords.
<b><i>bold italic text</i></b>	Variable elements; for example, parameters in code syntax.
<i>italic text</i>	New terminology; also emphasized words and book titles.
SMALL CAPS	Names of keys on the keyboard.
monospace text	Examples, such as XML fragments or Java code; or text you type exactly as it appears.

Descriptions of procedures also use the following conventions:

<b>File &gt; Import</b>	A menu path to follow; in this example, from the <b>File</b> menu, select <b>Import</b> .
CTRL+C	Press both keys at the same time.
ESC F I	Press and release each key in succession.

The manuals contain notes, tips, and warnings that provide particular information, as follows:

- *Notes* provide related information that does not fit directly into the flow of the surrounding text.
- *Tips* provide hints containing shortcuts or alternative ways of accomplishing a task.
- *Warnings* contain critical information that could prevent physical damage to equipment, data, or people.

Some of the diagrams in the manuals use EXPRESS-G graphical notation. For an explanation of the symbols in these diagrams, see [Appendix A, “EXPRESS-G Language Notation,”](#) in *Using the Context Mapper*.

Discussions of XML files contain diagrams that show the structure of the associated DTDs (document type definitions). Each of these diagrams contains a legend describing the symbols in the diagram.

**About Sidebars**

As you read the documentation, you will encounter information contained in sidebars like this one. These sidebars provide background material related to the surrounding information.

**Operating System Specifics.** All of the information in this manual applies to both Windows and UNIX systems. However, pictures of dialog boxes show the Windows “look” and path names use a backslash (\), rather than a forward slash (/), to separate directory names.

---

## What's in this Guide?

The *System Overview* contains the following sections:

- [Chapter 1, “Introducing Information Interoperability,”](#) describes what information interoperability is about and how the Modulant Contextia Interoperability Platform solves semantic conflicts among systems that must share information.
- [Chapter 2, “Introducing the CIIM,”](#) describes the Context-based Information Interoperability Methodology, Modulant’s patent-pending methodology.
- [Chapter 3, “The Interoperability Process,”](#) lists the stages in the process of implementing information interoperability from a high-level perspective.
- [Chapter 4, “Modulant Contextia Tools and Components,”](#) describes the design-time tools that are part of the Interoperability Workbench and the run-time tools and components that come with the Interoperability Server.
- [Chapter 5, “An Interoperability Example,”](#) presents a simple example of interoperability between two travel companies.
- [Chapter 6, “Architectural Considerations,”](#) shows some system architectures you might use when implementing information interoperability.
- The [Glossary](#) defines terms specific to the Modulant Contextia platform, as well as industry-standard terms used in the Modulant Contextia Documentation Library.

---

## Related Documents

In addition to this guide, the Modulant Contextia Interoperability Platform Documentation Library contains the following manuals:

- *System Overview* (this manual): Introduces the Modulant Contextia Interoperability Platform and the associated methodology, and describes all of the included tools and components.

- *Modulant Contextia Interoperability Server Guide*, which includes:
  - ◆ *Using the Interoperability Run Console*: Provides detailed explanations of each of the parts of the Contextia Interoperability Run Console, with information on troubleshooting the results of interoperability runs.
  - ◆ *Using the Interoperability Server Administrator*: Describes how to start the Interoperability Server, set database connections, and perform standard server administration tasks.
  - ◆ *Modulant Contextia Developer's Guide*: Introduces the Interoperability API, with examples of how to use it to automate interoperability runs.
  - ◆ *Installing the Modulant Contextia Interoperability Server*: Describes the system requirements for the Modulant Contextia Interoperability Server, and walks you through the installation and configuration process.
- *Modulant Contextia Interoperability Workbench Guide*, which includes:
  - ◆ *Using the Data Mapper*: Describes how to create a description of the physical format of your data and how to connect that to the logical description of your application's data structure—together this information forms a data map.
  - ◆ *Using the Context Mapper*: Describes the mapping process, the elements of the Modulant Abstract Conceptual Model, and how to use the Contextia Context Mapper to create a context map file.
  - ◆ *Installing the Modulant Contextia Interoperability Workbench*: Describes the system requirements for the Contextia Data Mapper and the Context Mapper, and walks you through the installation process.

In addition to the printed documents, your Modulant Contextia installation contains a complete online Documentation Library, in the **doc** subdirectory. You can find the online Documentation Library in both HTML and PDF format.

To access the Modulant Contextia Documentation Library, do one of the following:

- ▶ *From the Windows desktop*, select **Start > Programs > Modulant Contextia Interoperability Server > Documentation Library**.
- ▶ *On Solaris systems*, open the file **doc/wwhelp.htm** in your Interoperability Server installation directory in a browser window.

**Note:** If you cannot open a PDF file, download a copy of the free Acrobat Reader at:

[www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html)

---

# 1

## Introducing Information Interoperability

When there was only one computer program in the world, with only one user, there was no problem making sure all of the necessary information was available. This changed as soon as another computer system with different programs entered the picture. At this point, there was no direct way for one system, whose applications used information in a particular manner, to communicate with any other system, whose applications might use information in another manner altogether. This left each system isolated from the others, unable to share valuable information.

Over time, many methods of enabling disparate systems to communicate have been tried. Modulant has found a way to provide information interoperability—the ability of two or more computer systems to understand each other and use each other’s information in their own native context—while letting each participating system remain autonomous. Information interoperability adds a logical communication layer above the physical connections between systems.

Interoperability solutions complement existing enterprise systems and enable them to work together. Without an interoperability solution, it is difficult for different systems, even in the same conceptual domain, to share information effectively without losing some aspect of the intended meaning of the data.

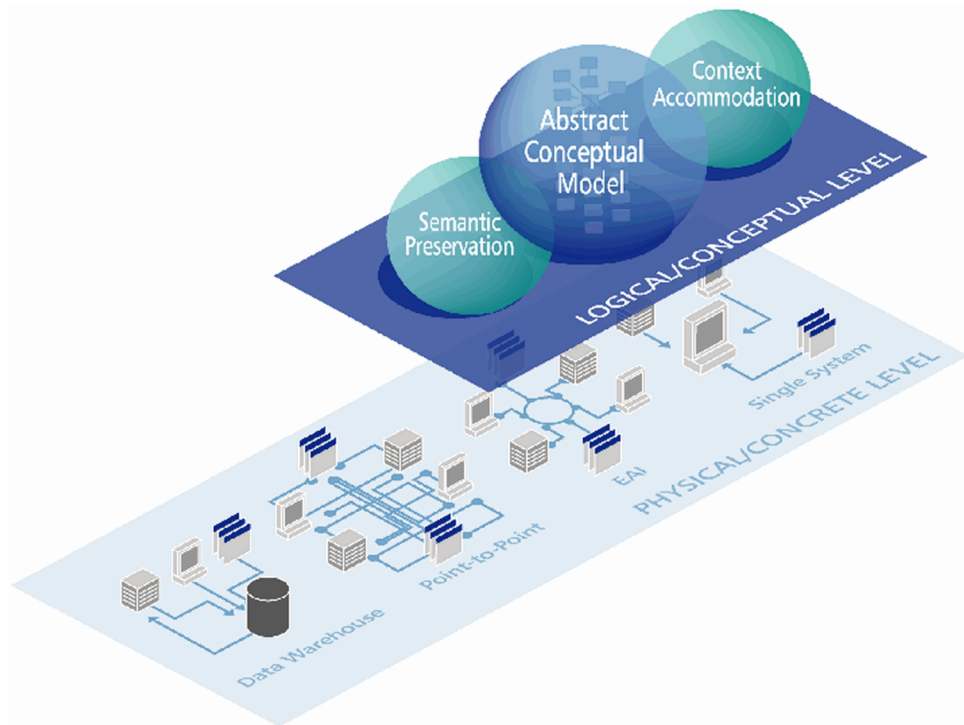
Information interoperability requires an understanding of the context in which data is created and used—the realization that the information conveyed by the data is always based on how people and organizations actually use the data. Therefore, interoperability architects work closely with domain experts to discover both explicit and implicit knowledge about application data.

This discussion covers the following topics:

- [Enabling Information Interoperability](#)
- [Interoperability is About Communication](#)
- [The Role of Context](#)

## Enabling Information Interoperability

Communication is only successful when the intended recipients have understood the message being conveyed. Information interoperability makes it possible to make information available to all members of an interoperability environment who need it to make business decisions. Modulant does this by retaining individual company culture, including specialized terminology, perspective, company folklore—which together form the context of an application. This method recognizes differences and commonalities without compromising the autonomy of each system.



Modulant makes interoperability possible by representing both application data and the context of that data in a computable form, known as a *context map*, using an abstract data model. Using these context maps, the Interoperability Engine enables information interoperability between systems in related conceptual domains (known as an *interoperability environment*), even if they use completely different formats and terminology.

Modulant's interoperability solution enables autonomous systems to exchange semantically-rich information by:

- Operating on a logical as well as a physical level.
- Focusing on the integrity of information exchange, rather than on application connectivity.

- Performing semantic mediation using abstract domain models.
- Accommodating variant application perspectives on data.
- Providing a non-invasive, loosely-coupled solution that remains separate from applications and data sources.

---

## Interoperability is About Communication

Interoperability is the ability of people, organizations, and systems to work together. Information interoperability is the ability of people, organizations, and systems to communicate complete, meaningful information that enables everyone involved to make the necessary decisions to conduct their business.

Consider two companies, one of which refers to the people who work there as “employees” and the other of which refers to the corresponding group of people as “staff members.” It would be burdensome to force either company to change its native terminology in order to be able to work together. Fortunately, that is not necessary. Instead, you can capture the meaning of both terms by moving up a level of abstraction, and refer to “people” in both cases. This enables both user communities to retain their preferred terminology and still communicate successfully.

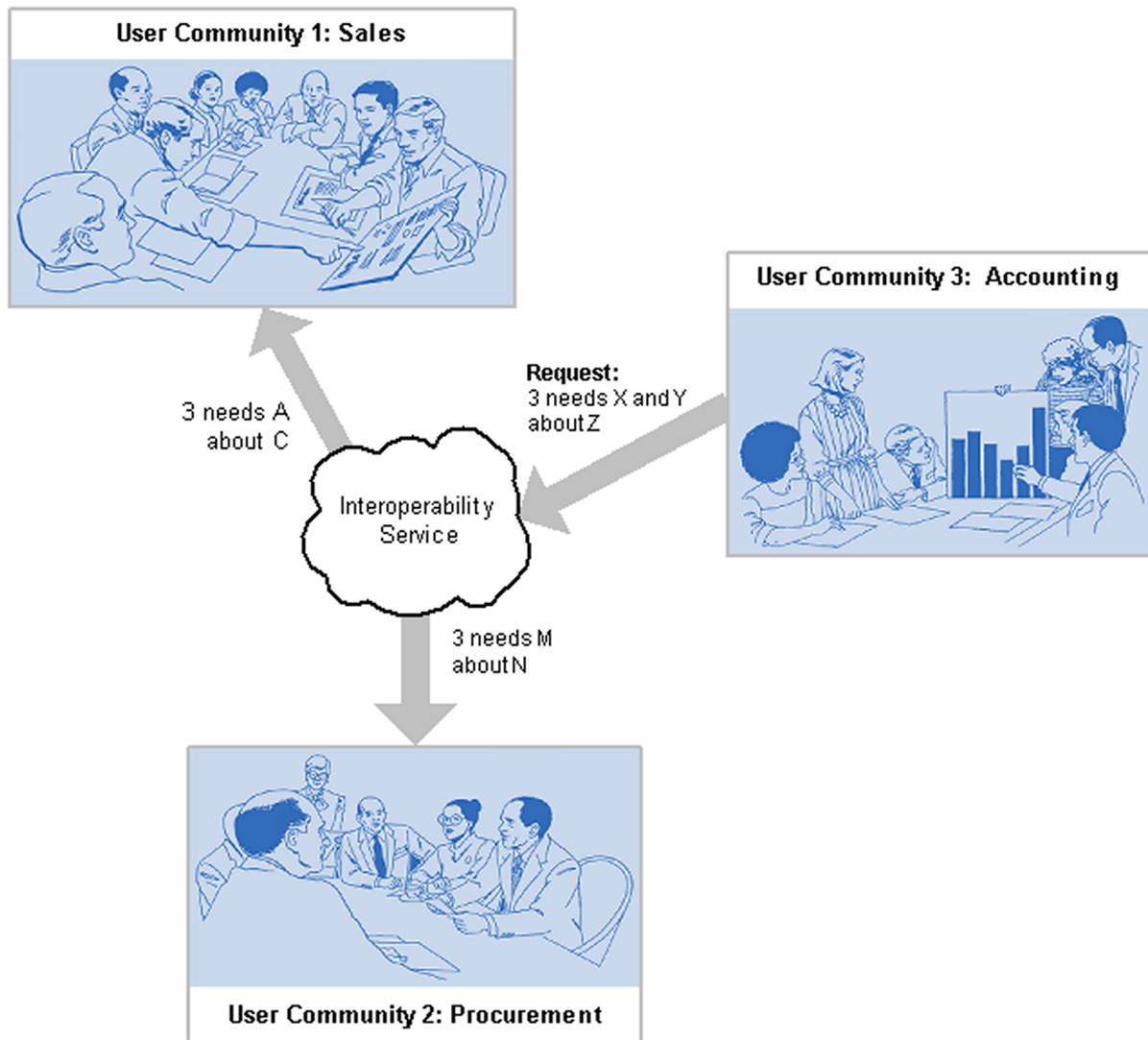
Figure 1 and Figure 2 show how interoperability makes effective communication possible. In Figure 1, members of User Community 3 need two pieces of information. They send a request to an interoperability service, from which additional requests are sent to User Communities 1 and 2 to provide the necessary information. Notice that each request is phrased using terms familiar to the recipients, in their own native context.

Consider the case where User Community 1 represents a Sales organization that stores data about customers and competitors; User Community 2 represents a Procurement organization that stores data about suppliers and subcontractors; and User Community 3 represents an Accounting organization that serves both User Communities 1 and 2.

# 1 Introducing Information Interoperability

Suppose that Accounting needs to know which of Sales's and Procurement's accounts are overdue. The question goes through the interoperability server, asking Sales, "Which of your customers pay late?" and asking Procurement, "Which of your suppliers do you owe money to?"

**Figure 1: Request for Information Across Related User Communities**

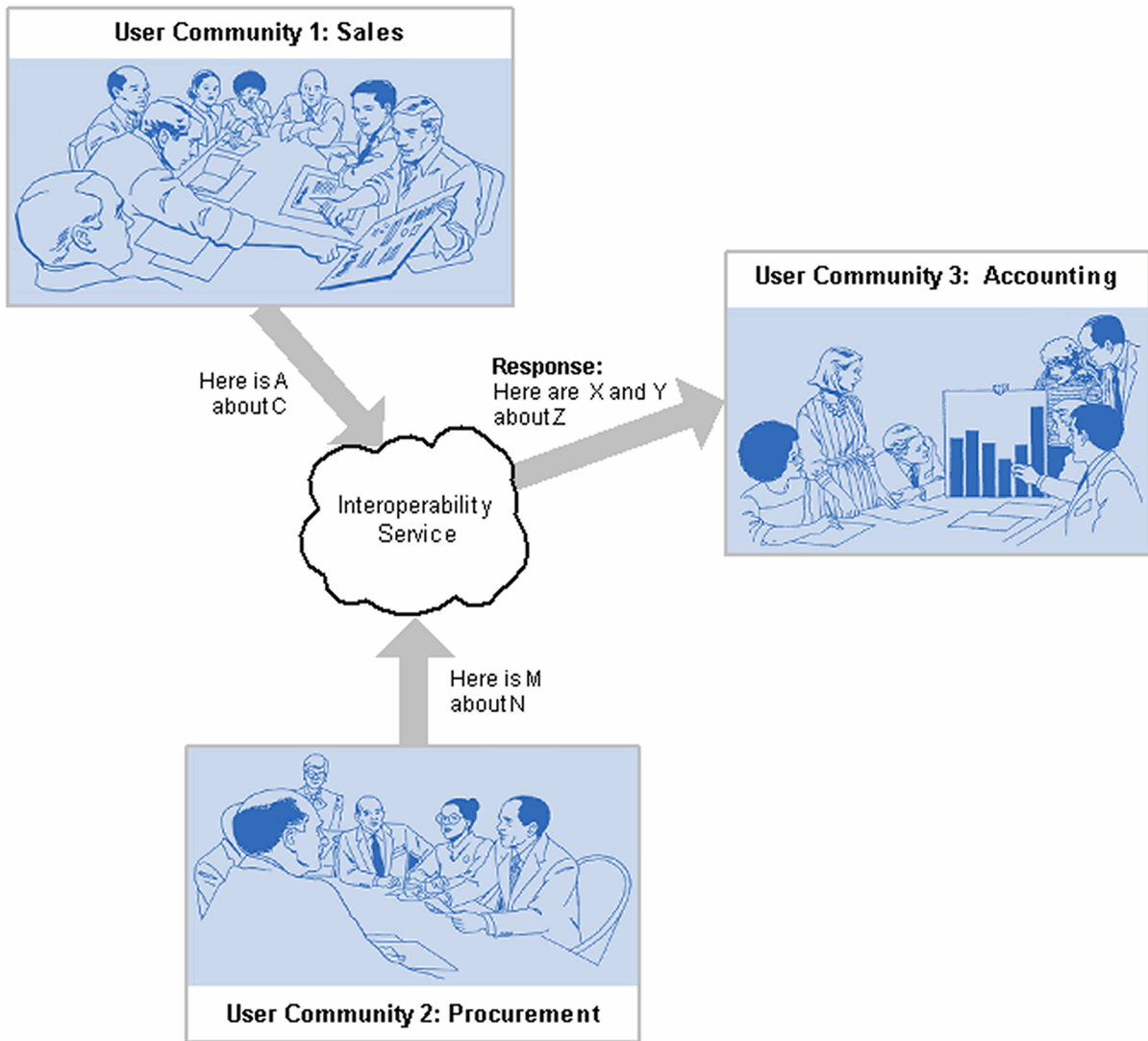




In Figure 2 you can see that each community answers using its own terminology, which gets transformed to a format that the requestors understand:

- Sales returns a list of customers that still have unpaid bills.
- Procurement returns a list of bills they have yet to pay in full.
- The interoperability server transforms this information and returns a list of delinquent accounts to Accounting.

**Figure 2: Response to Request for Information**



---

## The Role of Context

Context is a somewhat elusive concept, because it involves describing what is not always explicitly expressed. When you hear something that seems “out of context,” you know that some fundamental information is missing. The context lies in the details people often leave out because they think everyone knows them. Within your own community, this often works, but when you try to communicate with people outside your community, they might not understand until you clarify what was left out.

Context depends heavily on the audience, and on their shared assumptions. Member communities in an interoperability environment define the context of the data in their own applications, based on how they use that data. Native context includes the meaning and shades of meaning that data might have, as well as any constraints on the values and relationships among data elements. This context provides the foundation for mapping and data transformation for successful interoperability.

Application context is the relevant information about the environment where application data resides, including where the data is defined, created, and used. Data context provides a frame of reference and clarifies the meaning of the data. Without contextual information, data is not fully understandable.

The Modulant methodology relies on semantic abstraction: mapping data elements from a more specific conceptual representation in a narrow context to a more abstract conceptual representation in a broader, more encompassing context. The mapping process focuses on meaning and shades of meaning that data might have as well as constraints on values and allowed relationships.

---

# 2

## Introducing the CIIM

The Context-based Information Interoperability Methodology (CIIM) includes a number of methods that enable you to discover the context of application data elements and capture this information in a formalized representation. The result of following these methods is a data map file that describes the physical format of your data and its logical structure, and a context map file that includes an ATS (Application Transaction Set) schema (a logical representation of the application data structure in terms of entities and attributes) and one or more context maps that you can use as part of an interoperability run.

This section includes the following topics:

- [Underlying Principles](#)
- [The CIIM Framework](#)
- [CIIM Architecture](#)
- [The CIIM Methods](#)

---

### Underlying Principles

Successful implementation of an interoperability solution is impossible without a concrete and consistent logical framework for understanding digital information. Such a theory must consider how an interoperability solution views, understands, represents, and manipulates the data that must be shared among applications in an interoperability environment.

For example, without a concept of information apart from awareness of the data between the tags in an XML document or the data in a database, no tool can provide effective interoperability capabilities. It is necessary to account for all of the meta-information that makes raw data useful to actual people in order for the software to do anything significant to the information structures themselves.

The Context-based Information Interoperability Methodology is based on the following principles:

- 1 Effective information interoperability depends on knowledge of context.

- 2 Context can be described by characterizing data definition, data aggregation, data usage, and data constraints.
- 3 Context can be discovered from data values, data patterns, data descriptions, activities that create or use data, and functions that process or present data.
- 4 Context can be represented in a computable form.
- 5 Application information, including context, can be represented using an Abstract Conceptual Model (ACM).
- 6 Interrelated application contexts can be accommodated using common abstractions.
- 7 Declarative representation of instructions for information-preserving transformations between application data and an ACM enable ACM-mediated interoperability among applications.

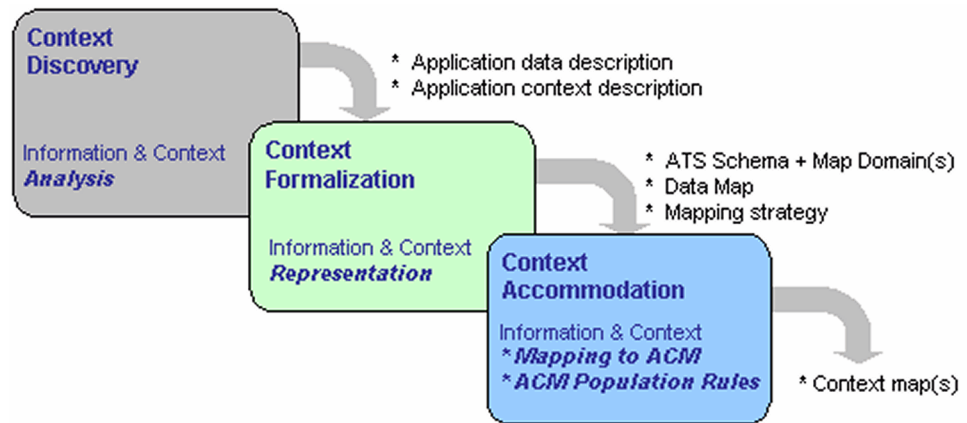
---

## The CIIM Framework

As shown in Figure 3 on page 8, the Context-based Information Interoperability Methodology includes a number of methods. The result of applying these methods is a data map and one or more context maps.

Figure 3 shows the methods that form the core of the Context-based Information Interoperability Methodology. The result of each method forms the input for the next method in the cascade.

**Figure 3: The CIIM Methods**



The major methods in the CIIM all address information and its context, each with a different focus:

- Context discovery focuses on analysis, of both data and its usage in context. This phase also includes learning about the physical data structures.

- Context formalization focuses on representation of that data, and results in an ATS schema—a list of data elements and their relationships—and a data map—a description of the physical format of the application data and how the data fields correspond to the logical description in the ATS schema.
- Context accommodation focuses on mapping data elements to an Abstract Conceptual Model and defining the rules the Interoperability Server uses for population and extraction operations on the ACM.

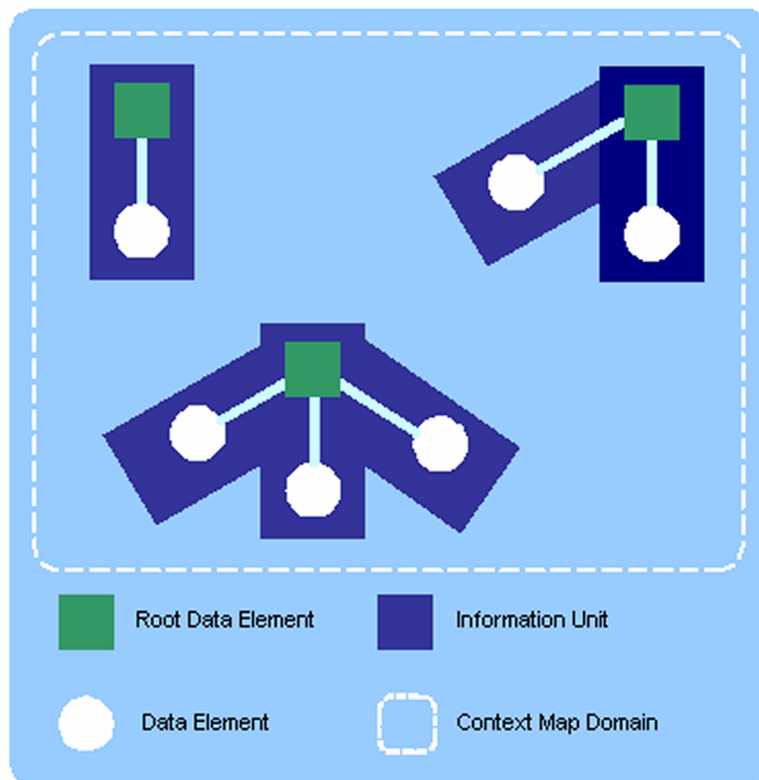
## CIIM Architecture

The Context-based Information Interoperability Methodology relies on the following key concepts to describe application data structures and relationships:

- Data Elements
- Information Units
- Map Domains

Figure 4 shows how data elements combine to form information units.

**Figure 4: Aggregations of Data and Information**



The following sections explain these terms.

### Data Elements

The CIIM recognizes two types of data elements in an application:

- *Data elements* are logical representations of data fields in an application. Each data element is directly related to a root data element.
- *Root data elements* are specialized data elements that identify a central concept or subject in a data set. They serve as anchor points for mappings of related data elements.

Root data elements are often the primary keys of database tables, or similar identifiers on which other data elements depend for them to make sense. For example, a product identifier is related to other characteristics of that product's appearance, and serves as the root data element for data elements related to products.

### Information Units

An *information unit* is the combination of a data element and the related root data element. This combination describes one fact about an identified "thing," where the root data element is the identifier. The smallest information unit possible contains only the root data element (and its relationship to itself).

For example, in a personnel application, if the employee ID is the root data element, the combination of the employee ID with the employee's name is an information unit.

The collection of information units that share the same root data element is an *information aggregation*. An information aggregation contains "information about the same thing" as it occurs in an application. Information aggregations often represent structures such as database tables, XML elements and their subelements, and entities in a data model.

**Note:** When you create structure mappings using the Context Mapper, each root structure in a map structure represents the context for an information aggregation. For more information, see ["Completing Structure Mappings" on page 202](#).

### Map Domains

The domain of a context map is the set of data elements required for an identified business process. For example, in an application that stores information about employees, you might learn during the context discovery phase that the process that prints timesheets requires only an employee's identification number and the person's last name. To enable information interoperability, you could create a context map that included only these two data elements. For an example, see [Chapter 2, "The Mapping Process."](#)

Depending on the complexity of the application and the number of business processes it supports, you might need to define more than one context map. Your analysis of the data and the mapping strategy you choose will help you decide which context maps are needed.

---

## The CIIM Methods

The methods of the CIIM are concerned with understanding, analyzing, and formally describing the semantics and context of application data.

This section discusses the following methods:

- [Context Discovery](#)
- [Context Formalization](#)
- [Context Accommodation](#)

### Context Discovery

Interoperability architects work with domain experts to apply the context discovery methods. Using these methods, they identify both explicit and implicit information about application data. Context discovery includes these stages:

- **Application Information Analysis**

First, you must learn what the application data represents and how the data elements relate to each other. To do this, you begin by examining instances of the data itself, along with any written documentation that describes the data structures. Knowing the scope of the data that will participate in the interoperability process helps you identify the data elements involved.

- **Business Process Context Analysis**

Once you have learned about the data, you proceed to discovering specific ways in which people and organizations use the data. For each of these usage scenarios, you collect information about the data in context. As you identify the data used in each business process, you can use the information about the context to plan an effective mapping strategy.

The context discovery phase produces a description of the application context, including details that were originally both explicit and implicit. At the end of this phase, you should have a description of the application data, either in the form of a physical application schema (for example, in an EXPRESS file or a DTD) or a written description of the data structures.

### Context Formalization

*Context Formalization* results in a definition of your application data structure, in the form of a data map (stored in XML format in a file with a **.dtm** extension). In this stage, you consolidate what you learned from the context discovery phase. Using this method, you codify all of the explicit and implicit knowledge that you gathered during your examination of the application data and design documents and your interviews with domain experts.

As part of this process, you create a description of the logical structure of the data, in terms of entities and attributes. This description is known as the *ATS schema*. The ATS schema includes a list of the data elements in the application, including the root data elements. At this point, you also identify the data elements in the application and their related root data elements, along with any requirements for format or data conversions.

Next, you develop a plan for how you will map the application's data elements and the relationships among them to your Abstract Conceptual Model. The mapping strategy includes a list of the Abstract Conceptual Model attributes to use as mapping targets for the data elements in the ATS schema and the relationships among ACM entities that represent the context of the application data.

### Context Accommodation

Having created a formal definition of an application's data set and identified an Abstract Conceptual Model that can represent the concepts in your application, you apply the *Context Accommodation* method. Using this method, you specify mapping targets for the application's data elements in your Abstract Conceptual Model based on your mapping strategy, and add contextual information that describes the relationships among related data elements. This is the heart of the mapping process.

Another part of the context accommodation process uses the formal description of the application data's semantics and context to create computable instructions for transforming data between different applications, while preserving semantics and accommodating context. This includes:

- Conversion Definition

Format conversions and data conversions provide additional information for the Interoperability Server to use as it processes application data.

- ◆ Using a *format conversion* you could specify how different applications represent the same value. For example, you can specify data conversions that separate the month, day, and year components from a date. This lets each application specify how it represents dates, independently of the abstract representation of each individual component. You can also use data conversions to define derived fields, so that you can map data elements that do not appear in the original data structures.



- ◆ Using *data conversions*, you can specify formulas for translating data from one application so that other applications can understand it. Data conversions let you create “derived” fields to enable you to map data elements that do not explicitly appear in the application data.
- Population Rules

Population rules enable you to connect application data elements to attributes in your Abstract Conceptual Model to supplement the connections already in the context map. These connections let you refine how the Interoperability Server populates the Abstract Conceptual Model with application data during an interoperability run. At this point, you also specify default values to use when populating Abstract Conceptual Model attributes for which there are no corresponding application data values.

The results of the context definition methods form the core of the context map, in the form of computer-processible declarative statements.

At the end of this stage, you have a context map file, with an ATS schema and one or more context maps that capture the context of specific business processes.

## 2 Introducing the CIIM

---

# 3

## The Interoperability Process

The interoperability process begins when you identify two or more applications that require information from each other. These applications together form an interoperability environment. To begin with, interoperability architects meet with domain experts to develop an interoperability strategy. For more information about developing this strategy, identifying an abstract representation to use, and producing context maps, see *Using the Context Mapper*. For information about creating data maps that describe the physical format of application data, see *Using the Data Mapper*.

The overall interoperability process follows these stages:

- [Design-time Process: Applying the CIIM](#)
- [Run-time Process: Achieving Information Interoperability](#)

Some of the stages in the interoperability process are supported directly by the Modulant Contextia Interoperability Platform. Other steps depend on your own system architecture and tools.

---

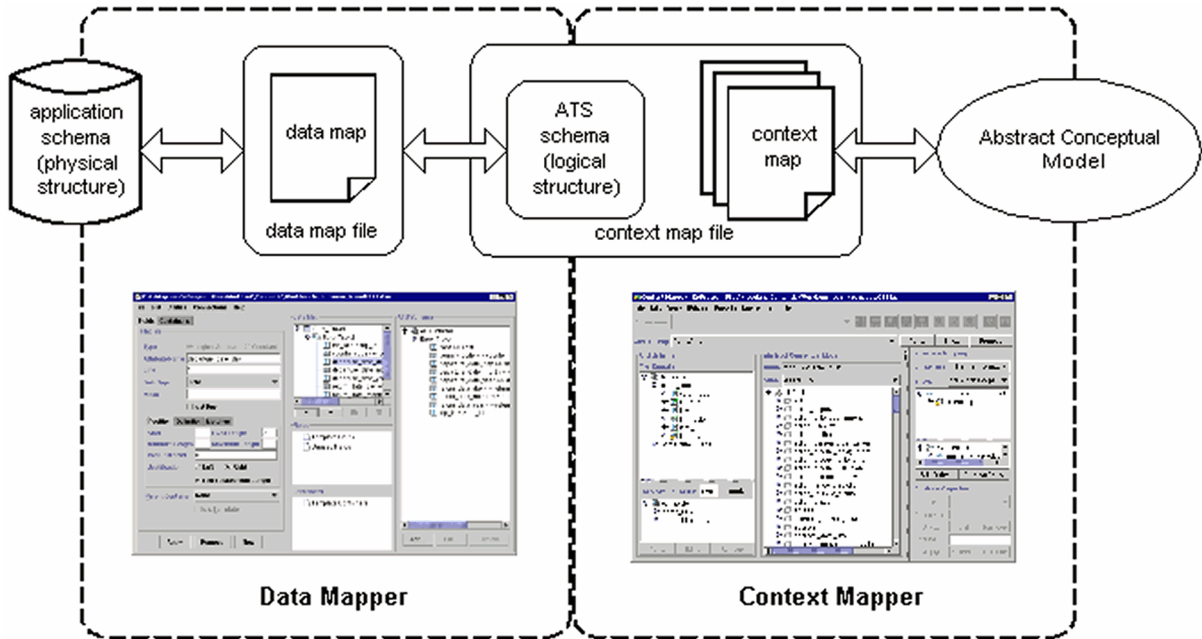
### Design-time Process: Applying the CIIM

The design-time process includes the following stages:

- [Develop an Interoperability Strategy](#)
- [Describe Application Data Formats in Data Maps](#)
- [Capture Application Context in Context Maps](#)

Figure 5 shows what you create during the design-time process and the Modulant Contextia tools that support each part of the process. For each application in your interoperability environment, you start with a physical description of the application data. Using this description, you create both a data map, which mirrors the structure of the physical data fields in the application and an ATS schema, which represents the logical structure of the data in terms of entities and attributes. The ATS schema becomes the foundation for one or more context maps, which describe the context of the data elements in the application and their relationships in terms of an Abstract Conceptual Model.

Figure 5: Design-time Process



The following sections describe the high-level process that supports the creation of the mapping strategy and the resulting mapping specifications.

## Develop an Interoperability Strategy

The first step in implementing information interoperability is identifying the member communities in an interoperability environment, and the applications that must share information. With this information, you can begin to develop an interoperability strategy.

A significant part of any interoperability strategy is the abstract representation of the application domain, or *ontology*, that will be the starting point for your context maps. The Modulant Contextia Interoperability Platform refers to this representation as an *Abstract Conceptual Model (ACM)*. The default installation includes an ACM schema defined as an EXPRESS model. You can modify the Modulant ACM to fit the needs of your application domain. For more information about the contents of this model, see [Chapter 4, “Inside the Abstract Conceptual Model,”](#) in *Using the Context Mapper*.

When you design interoperability strategies for the member applications in an interoperability environment, it is essential that you plan all your context maps using the same ACM—one that contains enough detail to represent the concepts in all of the applications that must be able to work together.

An effective interoperability strategy requires collaboration between interoperability architects and domain experts:

- Interoperability architects bring their knowledge of the application data, including its context and knowledge of the Abstract Conceptual Model to which the application data is to be mapped.
- Domain experts bring their familiarity with the specifics of each application, including the context of the data and how it is used.

Together, they identify the following information:

- The major concepts, ideas, or things that each application deals with and holds information about.
- The major concepts, ideas, or things that are common across applications that need to interoperate.

Using this information, they can determine which concepts from the Abstract Conceptual Model to use to represent each of the application concepts and their relationships.

At the end of this stage, you will have an EXPRESS model containing an Abstract Conceptual Model schema that you can use to define context maps and a basic agreement about which concepts in this model correspond to the major areas of application functionality in the member applications.

## About Abstract Conceptual Models

An Abstract Conceptual Model (ACM) represents a collective understanding of the semantics of information commonly used by business applications. The entities in this model are abstract enough to apply to a variety of situations. This abstraction allows you to reuse the same entities as mapping targets in different contexts within an interoperability environment.

The process of mapping involves associating data elements from an ATS schema with attributes of entities in an Abstract Conceptual Model.

To create effective mappings, you must understand not only the entities in your ACM and their relationships, but also the context and relationships of the data elements in the applications that will share this data.

In order to work with the Modulant Contextia Interoperability Platform, an Abstract Conceptual Model must be written in the EXPRESS language. The Modulant ACM is extensible—you can add to it if it does not contain entities that apply closely enough to a particular situation. You can even expand this EXPRESS model to encompass new application domains.

## The EXPRESS Modeling Language

The EXPRESS data specification language is part of STEP (Standard for the Exchange of Product model data), defined in International Standard ISO 10303, and is used in the STEP methodology.

The STEP team discovered that existing modeling languages were not adequate to convey the richness of the semantics required to represent abstract models containing things, associations among those things, constraints, and inheritance (represented by supertypes and subtypes).

The EXPRESS language can formally describe the structure and correctness conditions of any information that needs to be exchanged. The EXPRESS language describes constraints as well as data structure. Formal correctness rules will prevent conflicting interpretations.

## About Application Transaction Sets (ATSs)

An *application transaction set* (ATS) is information that collectively represents the data and context of an application for the purpose of interoperability. An ATS serves as the “footprint” of an application in the Modulant toolset. An ATS contains the following components:

- Context map files, which are also known as CXM files. A CXM file has two parts:
  - ◆ The *ATS schema*, which represents the logical structure of the application data—the data elements and their properties.  
The physical representation of the structure is known as the *application schema*. You use the application schema to create the ATS schema.
  - ◆ One or more *context maps*, which contain a set of mapping statements and the associated structure mappings. A context map associates the ATS schema with an Abstract Conceptual Model (ACM) schema.

You use the Context Mapper, part of the Modulant Contextia Interoperability Workbench, to create CXM files.

- Data map files, which describe the physical format of each source of application data, and relate the physical data fields to logical data elements in the ATS schema.

You use the Contextia Data Mapper, another part of the Modulant Contextia Interoperability Workbench, to create data map files.

The context map files and data map files are created in the mapping process and is used in the interoperability run process.

Associated with each ATS is application data, which becomes known as *ATS data* after it has been imported into the Interoperability Server. The application data can come from:

- the tables of a relational database
- forms or reports generated by an application
- XML data files
- free-form or delimited flat files

## Describe Application Data Formats in Data Maps

As shown in [Figure 5 on page 16](#), the design-time process starts with a physical description of an application's data structure. From there you build a logical representation of that data structure, which you use to define context maps that capture the abstract concepts conveyed by the data and the context in which it is used.

This stage includes the following tasks:

- [Develop Application Schemas](#)
- [Create Data Maps to Describe Physical Structures](#)

## Develop Application Schemas

For each application in your interoperability environment—whether it will serve as a source or a target of an interoperability run—you must describe the physical structure of the data in a format that the tools in the Modulant Contextia platform can understand.

To begin this process, you can create an application schema. You can save the application schema as an XML DTD, a delimited flat file (such as CSV, comma-separated values), or an EXPRESS model, or you can read the structure directly from a relational database.

Using the application schema, you create a logical representation of the data structure in terms of entities and attributes, which you save as an *ATS schema* in a context map file. Each data map file you define will describe the data fields in the application schema and how they correspond to the logical data elements in the ATS schema.

At the end of this stage, you will have an application schema that describes the physical data structures used by your application. You will use the application schema to create the ATS schema section of your context map file.

### Create Data Maps to Describe Physical Structures

For each physical source of application data, that is, for each file format your application data uses, you create a *data map file* that delineates the actual data formats. The data map file serves two purposes:

- To enable the Data Importer and Exporter components of the Interoperability Server to parse application data in its native format.
- To define the relationship between the data elements in the ATS schema portion of the context map file and the actual application data.

Your application data can take a variety of formats, including XML, delimited or free form flat-files, or the tables of a relational database. The corresponding ATS schema, however, must describe the data in a logical format using entities and their attributes. Using the data map file, you specify how each of the fields in the physical data file corresponds to an attribute of one of the entities in the logical representation of the data structure. For more information, see *Using the Data Mapper*.

At the end of this stage, you will have data map files that define the format of source and target application data. The Modulant Contextia Interoperability Server will use these files to parse source data as the input to an interoperability run and to export target application data in its native format.

### Capture Application Context in Context Maps

For each application in your interoperability environment, you must create one or more context maps that define the relationships between the application schema and attributes of entities in your Abstract Conceptual Model.

#### About Context Map Files

A context map file (also known as *CXM file*) is a representation of one or more context maps in XML format. The context map includes both the ATS schema and mapping information, including mapping statements and structure mappings. The ATS schema portion of the CXM file is a list of the data elements in your application schema and their properties, which forms a logical representation of the structure of your data, using entities and their attributes.

Mapping involves discussions between interoperability architects (specialists in the Context-based Information Interoperability Methodology, including the Modulant Abstract Conceptual Model) and domain experts (specialists in the source or target applications). Jointly, they confer about:

- data meaning and nuance
- isolating individual semantics so they can be separately mapped (think of this as semantic normalization)
- how to represent concepts in the Abstract Conceptual Model



The end result of the mapping process is one or more context maps that associate data elements in a source or target application with attributes in the Abstract Conceptual Model.

For details about the structure and contents of CXM files, see *Using the Context Mapper*.

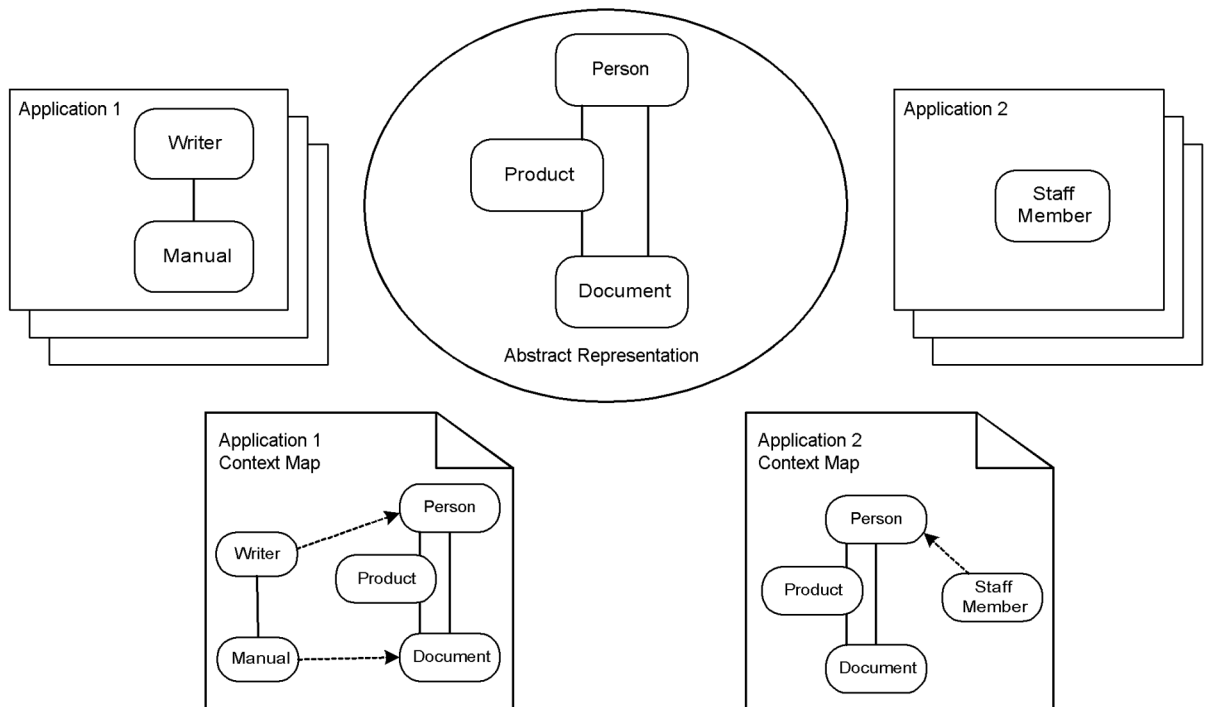
## The Mapping Process

Figure 6 shows a small portion of the structure of two applications and how each one can be mapped to an Abstract Conceptual Model (an abstract representation):

- **Manual** and **Writer** in the Application 1 are mapped to **Document** and **Person** in the Abstract Conceptual Model.
- **Staff Member** in Application 2 is mapped to **Person** in the Abstract Conceptual Model.

In an interoperability run, **Writer** in the source application (Application 1) would be transformed into **Staff Member** in the target application (Application 2).

**Figure 6: Mapping Data Elements to an ACM**



The mapping process includes the following steps:

- Identify Map Domains
- Map Data Elements to ACM
- Specify Required Conversion Definitions

At the end of this stage, you will have CXM files for the source and target applications in your interoperability environment. You will be able to use these CXM files when you perform interoperability runs.

For more information about context maps and the process for creating them, see *Using the Context Mapper*.

### Identify Map Domains

Before you can define a context map, you must determine which data elements in the logical representation of the application data structure—the *ATS schema*—will participate in each of your context maps. You can create the ATS schema using either the Data Mapper or the Context Mapper.

The ATS schema lists the data elements in the application, along with their properties, and defines each data element as an attribute of a logical entity that represents a particular “thing” in your application.

Each context map can contain a subset of the data elements in the ATS schema, or all of the data elements. The data elements in a context map are known as the *map domain*. The domain of a context map contains all of the data elements necessary to represent a known business process, whose context the map will capture.

### Map Data Elements to ACM

After defining the domain of a context map, you specify mapping targets for the data elements from the attributes of entities in your Abstract Conceptual Model. Then you complete the structure mapping for the context map. The structure mapping connects the mapping target for each data element and the mapping target for the associated root data element.

### Specify Required Conversion Definitions

In some cases, the data elements in your ATS schema do not correspond directly to attributes of entities in your ACM. A *conversion definition* enables you to specify how to handle a data element that does not have a direct mapping target in the ACM.

Specifying the rules for separating or combining data elements, or for deriving entirely new data elements from existing ones is part of the conversion definition process.

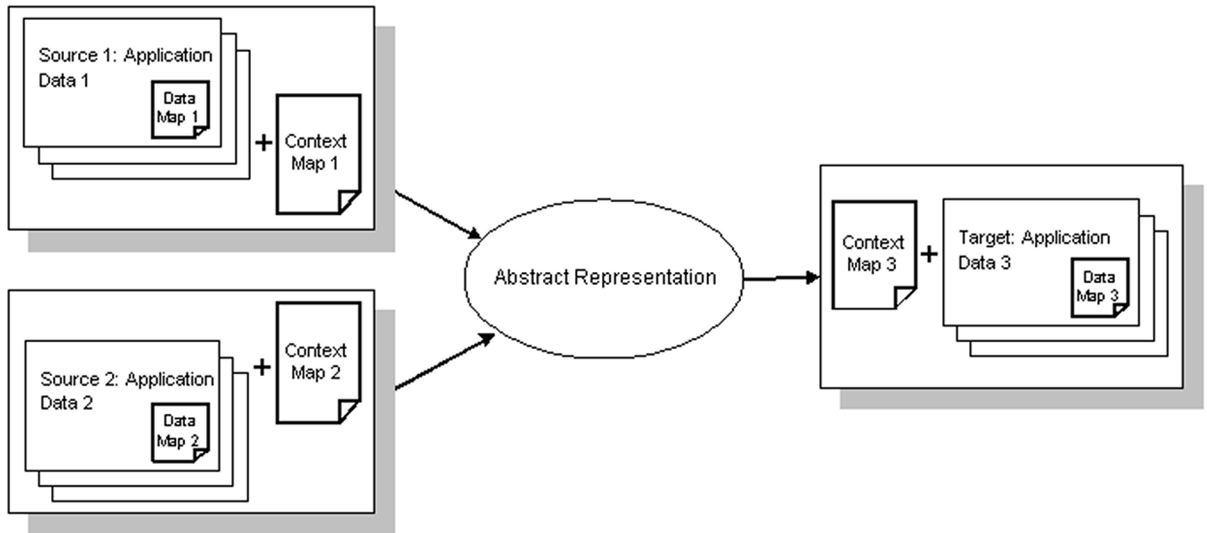
## Run-time Process: Achieving Information Interoperability

The run-time process includes the following stages:

- Configure Interoperability Run
- Perform Interoperability Run
- Retrieve Target Data in Native Format and Verify Results

Figure 7 shows how the run-time process uses the mapping specifications you created at design-time to produce information interoperability.

**Figure 7: Enabling Information Interoperability**



### Configure Interoperability Run

After creating data maps and context maps for each of the applications that will participate in an interoperability run, you are ready to define the parameters of the run itself.

For each source and target application that will participate in an interoperability run, you must specify the locations of one or more files of the following types:

- *CXM file*, which contains the ATS schema (a logical description, in terms of entities and attributes, of the application's data structure) and one or more context maps.

For information about creating context maps and the XML format in which they are stored, see *Using the Context Mapper*.

- *Application data*, which can be in a file or a relational database or inline with the run configuration parameters.
- *Data map file*, which describes how to parse and format the physical data, and how the data fields in the application correspond to the logical entities and attributes in the ATS schema section of the CXM file.

For information about creating data maps and the XML formats in which they can be stored, see *Using the Data Mapper*.

You can cache frequently used files, such as your Abstract Conceptual Model, on the Interoperability Server before you initiate interoperability runs that use those files. You can do this in the following ways:

- using the Interoperability CL
- using SOAP messages you send to the Interoperability Server using HTTP
- using the Interoperability API

This stage produces a set of run configuration parameters that you pass to the Interoperability Server. If you use the Interoperability Run Console to define the interoperability run, you can save the run configuration parameters in an XML file. You can use this file to perform interoperability runs using the Interoperability CL, SOAP messages, or Java client programs you write using the Interoperability API.

### Perform Interoperability Run

Using the configuration parameters you provide, the Interoperability Server performs the interoperability run. At the end of the run, the Interoperability Server extracts target data from the ACM as ATS data in the internal work space. From there, the data is exported to target application data files. The Interoperability Server uses the information in the target data map files to format the exported data in the target application files you specified.

For more information, see [“Flow of an Interoperability Run” on page 36](#).

At the end of this stage, you will have target application data. At this point, the target data is formatted using the context and format familiar to users of the target applications, and is ready to transfer to the system that needs it.

### Retrieve Target Data in Native Format and Verify Results

After a successful interoperability run, you retrieve the target data from the Interoperability Server. At this point you can review the results and verify that the target data matches your expectations.

At the end of this stage, your target application data will be available for use by the applications that need it.

---

# 4

## Modulant Contextia Tools and Components

The Modulant Contextia Interoperability Platform is a collection of tools and components that enable interoperability among heterogeneous applications running on different machines and platforms. It fosters interoperability by sharing information—including the full semantics and the context of the data usage. The Modulant Contextia platform resolves conflicts between incompatible systems by preserving the semantics and context of data.

To create true interoperability, the Modulant Contextia platform transforms data from one source to another by mapping the logical schema of each application to an abstract representation—known as an *Abstract Conceptual Model* (or, for short, an ACM). The Modulant Contextia Interoperability Server reads context and data maps for each participating application. The schema and map files that describe the structure and context of an application’s data are known collectively as *Application Transaction Sets*, or ATSS.

The tools that come with both the Interoperability Workbench and the Interoperability Server run on both Windows and Solaris systems. The Interoperability Server uses a database to manage the internal details of interoperability runs. The number of database connections you define determines the multi-threaded behavior of the server. Both Oracle and SQL Server databases are supported.

This chapter addresses the following topics:

- Design-time Tools: The Interoperability Workbench
- Run-time Tools: The Interoperability Server and its Client Tools
- Inside the Interoperability Server

---

### Design-time Tools: The Interoperability Workbench

The Modulant Contextia Interoperability Workbench contains tools that support the design-time stages described in “[Design-time Process: Applying the CIIM](#)” on page 15.

The Interoperability Workbench contains the following tools:

- [The Data Mapper](#)
- [The Context Mapper](#)

Two additional Windows-based data modeling tools, [FirstSTEP XG](#) and [FirstSTEP EXML](#), support the Interoperability Workbench. Both of these tools are available on the Workbench CD.

### The Data Mapper

The Data Mapper enables you to describe the physical structure and format of application data. The information you provide specifies how to parse the data to identify the values for each data element at runtime. In addition to the physical description, you specify how the physical constructs in the application data correspond to the logical entities and attributes in an ATS schema.

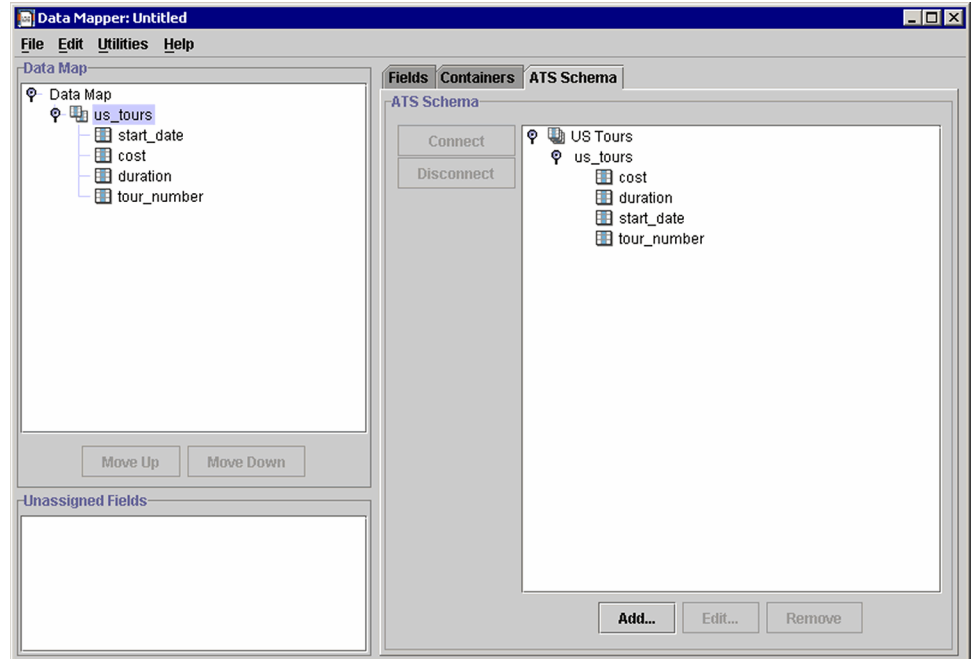
Data map files can describe data in the following formats:

- XML data
- flat-file data, with or without specific delimiters between fields
- relational database tables in Oracle and SQL Server databases

Figure 8 shows the main window of the Data Mapper:

- The **Fields and Containers** region on the left shows you the properties of the selected element in the data map.
- The **Data Map** region in the middle shows the structure of the data map as you build it.
- The **ATS Schema** region on the right shows the logical structure of the data, in terms of entities and their attributes, and how this logical structure corresponds to the physical data fields in the application data.

Figure 8: Data Mapper Main Window



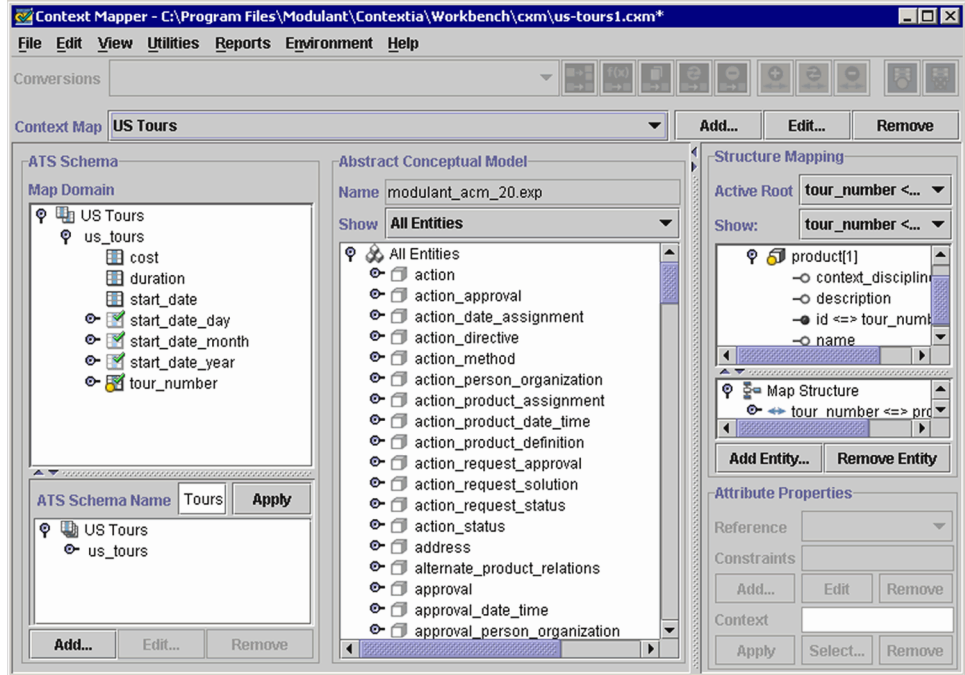
## The Context Mapper

The Contextia Context Mapper enables you to define one or more context maps for an application schema. The main window shows you the data elements in your application along with the parts of the context map as you develop it. In addition, you can view all or part of the structure of the Abstract Conceptual Model that contains the mapping targets for your data elements.

Figure 9 shows the main window of the Context Mapper:

- The **ATS Schema** region lists the data elements in your ATS schema and the domain of the current context map. In this region, you can view and change the properties of data elements and add data elements to the domain of a context map.
- The **Abstract Conceptual Model** region lists the entities in the Abstract Conceptual Model that contains the mapping targets for the data elements in your ATS schema.
- The **Structure Mapping** region displays the structure mapping for a context map. The structure mapping is a set of instances of Abstract Conceptual Model entities and their relationships that together provide the context for that data elements that appear in the context map.

**Figure 9: Context Mapper Main Window**



## FirstSTEP XG and FirstSTEP EXML

The Modulant Contextia methodology takes advantage of two Windows-based data modeling tools, FirstSTEP XG and FirstSTEP EXML. Both of these tools were developed by PDIT (Product Data Integration Technologies), which is a wholly-owned subsidiary of Modulant Solutions, Inc.

FirstSTEP XG lets you create a graphical representation of a data model using the EXPRESS modeling language. You can export the graphical representation to a text file in EXPRESS format.

FirstSTEP EXML takes a text-based EXPRESS schema and exports an XML DTD in a format that the Modulant Contextia tools recognize and can use.



## Run-time Tools: The Interoperability Server and its Client Tools

The Modulant Contextia Interoperability Server comes with the following client tools, which provide a variety of ways you can configure and initiate interoperability runs:

- The Interoperability Run Console
- The Interoperability CL
- The Interoperability Server Administrator
- The Interoperability API

For details about the components of the Interoperability Server itself, see “[Inside the Interoperability Server](#)” on page 33.

### The Interoperability Run Console

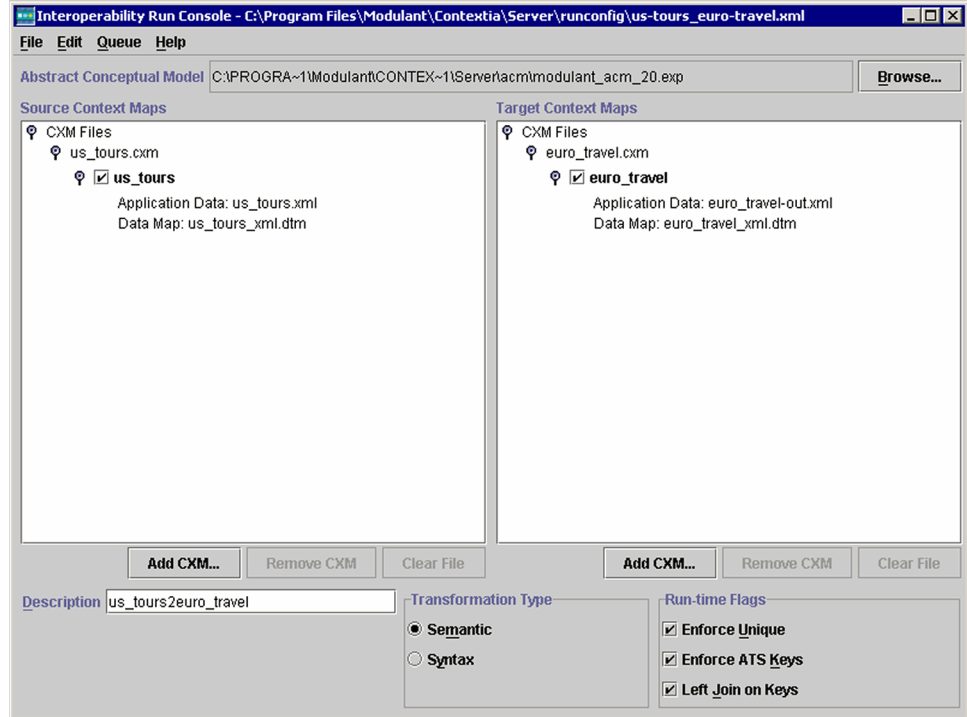
The Interoperability Run Console enables you to perform asynchronous interoperability runs, managed in a job queue. Figure 10 shows the main window of the Interoperability Run Console.

The region on the left contains information about source applications that can participate in interoperability runs. The region on the right contains parallel information about target applications that will contain the results of interoperability runs.

Below the **Source Context Maps** and **Target Context Maps** regions is a set of global parameters that apply to this interoperability run.

- *Transformation Type*: Specifies how to process this interoperability run.
  - ◆ **Semantic**: Perform a complete interoperability run, using the full capabilities of the Modulant Contextia Interoperability Server.
  - ◆ **Syntax**: Bypass the Populator and the Extractor and only perform conversion definition operations.
- *Run-time Flags*: A group of settings that let you control the behavior of the Interoperability Server during the population and extraction phases of an interoperability run.
  - ◆ **Enforce Unique**: If you set this option, the Interoperability Server enforces the uniqueness rules (that is, the key attribute constraints) defined in the Abstract Conceptual Model while populating source application data.
  - ◆ **Enforce ATS Keys**: If you set this option, the Interoperability Server enforces primary keys to avoid duplicate rows when extracting target application data.
  - ◆ **Left Join on Keys**: If you set this option, the Extractor creates a key query that uses the intersection of database table keys, plus the records that match the keys. Setting this option guarantees that you get a record back for every combination of the keys for which you have data.

**Figure 10: Interoperability Run Console Main Window**



## The Interoperability CL

As a companion to the Interoperability Run Console, Modulant provides a command-line tool that lets you manage the queue of interoperability runs. Using the Interoperability CL (**icmd**), you can add interoperability run jobs to the queue, obtain the status of a queued job, cancel a job, or remove jobs from the queue.

Using the Interoperability CL, you can perform interoperability runs either synchronously or asynchronously. Synchronous runs are executed sequentially from submittal to the Interoperability Server through processing to the return of results to the requestor in one operation. Asynchronous runs, on the other hand, are placed on a queue of interoperability jobs, and processed from there. At the end of a synchronous run, you must explicitly request the Interoperability Server to return the target data to you.

Figure 11 shows the usage message of the Interoperability CL, listing the commands and options you can specify when you use this tool.

Figure 11: Interoperability CL Usage Message

```

C:\>cd program files\modulant\contextia\server\bin
C:\Program Files\Modulant\Contextia\Server\bin>icmd
Must specify an action.
Usage: java -Dmodulant.root=<dir> modulant.TransformCtrl [options] run <run conf
ig file>... ! runlist <list file>... ! submit <run config>... ! submitlist <lis
t file> ! list [-<status>] ! info <id> ! cancel <id> ! remove <id> ! result <id>
! upload <type> <file> ! server ! version
Options:
-t, --times
    output timing information
-S, --single
    single threaded engine (only useful with 'server' action)
-s, --silent
    disable feedback messages
-c, --configurator type <argument>
    choose a configurator type [embedded|jms|soap]
-h, --host name with port number <argument>
    SOAP server host name with port number [hostname:port] (for
    soap only)
-d, --debug
    enable additional debug information
C:\Program Files\Modulant\Contextia\Server\bin>

```

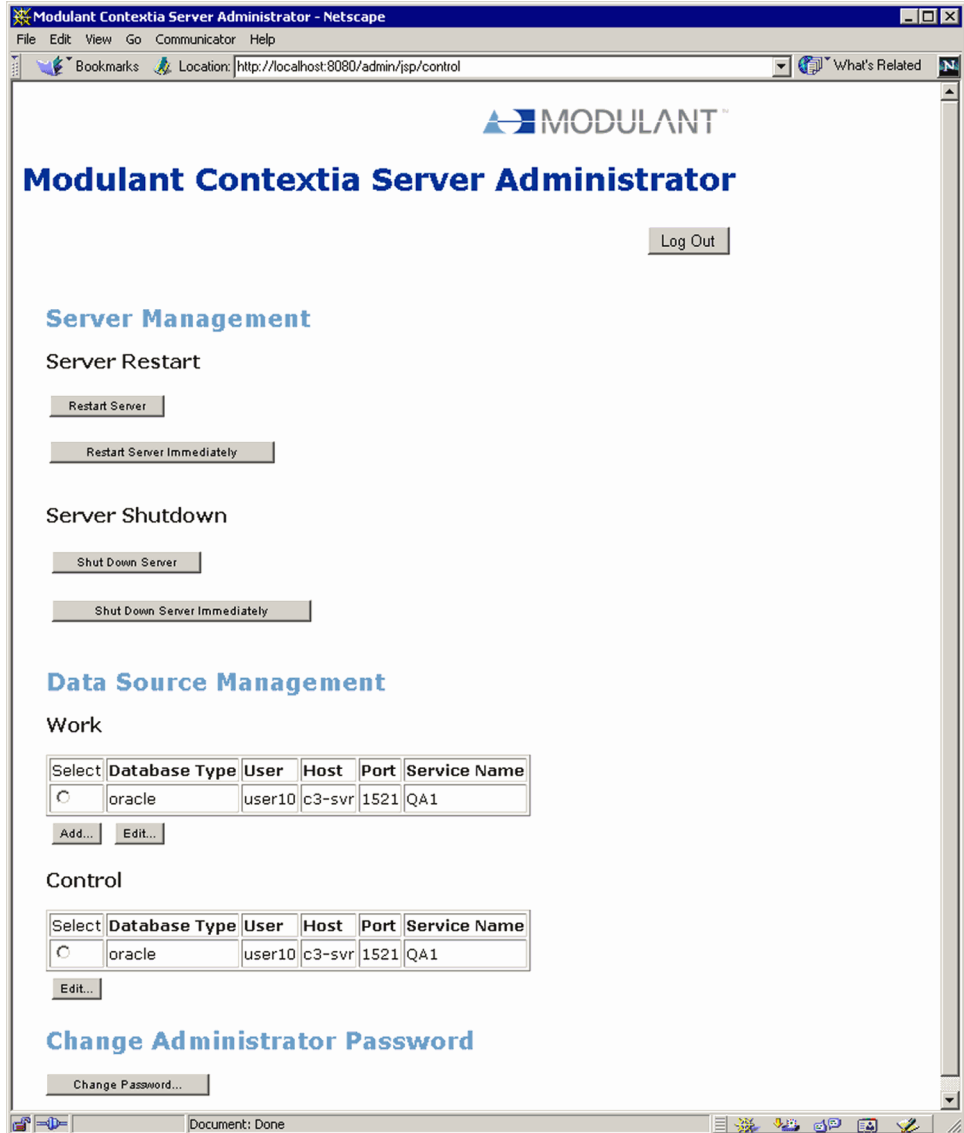
## The Interoperability Server Administrator

The Interoperability Server Administrator provides a Web-based interface for remote management of basic operation of the Interoperability Server. The Server Administrator is only available when the Interoperability Server is already running. You must start the Interoperability Server locally using a command-line interface, on the host machine where it is installed.

After the Interoperability Server is running, you can use the Server Administrator to restart the server, shut down the server, and modify database connection information.

Figure 12 shows the main window of the Interoperability Server Administrator. This page has three main regions: one for server management, one for data source management, and one for changing the administrator password.

Figure 12: Interoperability Server Administrator Main Page



## The Interoperability API

The Interoperability API is a set of Java classes and interfaces that you can use to create custom client programs to define, initiate, and monitor interoperability runs.

Using the Interoperability API, you can specify the parameters for an interoperability run, including the source and target applications and the run-time options. You can submit this information directly to the Interoperability Server or you can save it in an XML file.

You can perform synchronous or asynchronous interoperability runs, using either JMS or SOAP to communicate from a client program to a remote server.

---

## Inside the Interoperability Server

The Interoperability Server does the actual work of performing interoperability runs, using the parameters you specify.

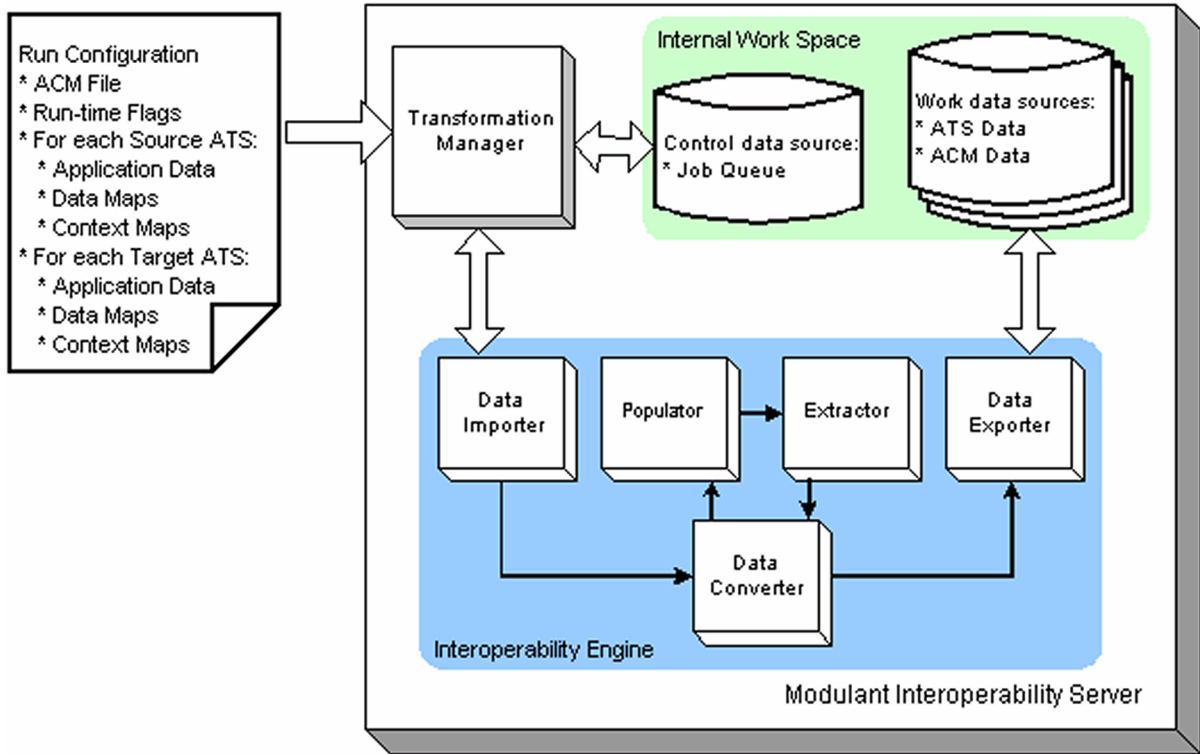
This section contains the following topics:

- [Server Components](#)
- [Flow of an Interoperability Run](#)

## Server Components

Each interoperability run is defined by the parameters in a single run configuration file. This file contains overall configuration information including the location of the Abstract Conceptual Model and server-related settings, and a list of files associated with each source and target application.

Figure 13: The Modulant Contextia Interoperability Server



The Interoperability Server stores information in an internal work space during each interoperability run. The internal work space contains two types of data sources:

- The **control** data source stores a queue of asynchronous interoperability runs.
- Each **work** data source stores temporary data used during a single interoperability run. The number of work connections determines how many threads the Interoperability Server can use in multi-threaded mode—one thread per connection.

The Interoperability Server contains the following components, shown in Figure 13, which work together to perform interoperability runs:

- *Transformation Manager*: The Transformation Manager uses the **control** data source to manage a queue of interoperability jobs and initiates jobs when server threads are available to run them.
- *Data Importer*: At the beginning of a run, the Data Importer parses source application data based on the corresponding data map file and imports that data into source ATS data in the **work** data source.

- *Populator/Extractor*: This part of the Interoperability Server first takes imported source data from ATS data in the **work** data source and, following the rules in the associated context maps, populates ACM tables with that data. Then it uses the rules in the target context maps to extract data from those populated ACM tables into target ATS data so that it can be exported.
- *Data Converter*: This part of the Interoperability Server comes into play twice during an interoperability run, to execute any conversion definitions you have specified in source or target context maps:
  - ◆ After the Data Importer has imported source application data to ATS data, but before the imported data is populated to ACM tables by the Populator.
  - ◆ After the data populated into the ACM tables has been extracted as ATS data by the Extractor, but before the Data Exporter exports the extracted data to a target application data file.
- *Data Exporter*: At the end of a run, the Data Exporter reads data from target ATS data in the **work** data source and exports it to target application data using the corresponding data map file.

The Interoperability Server uses the following additional components to communicate with client programs:

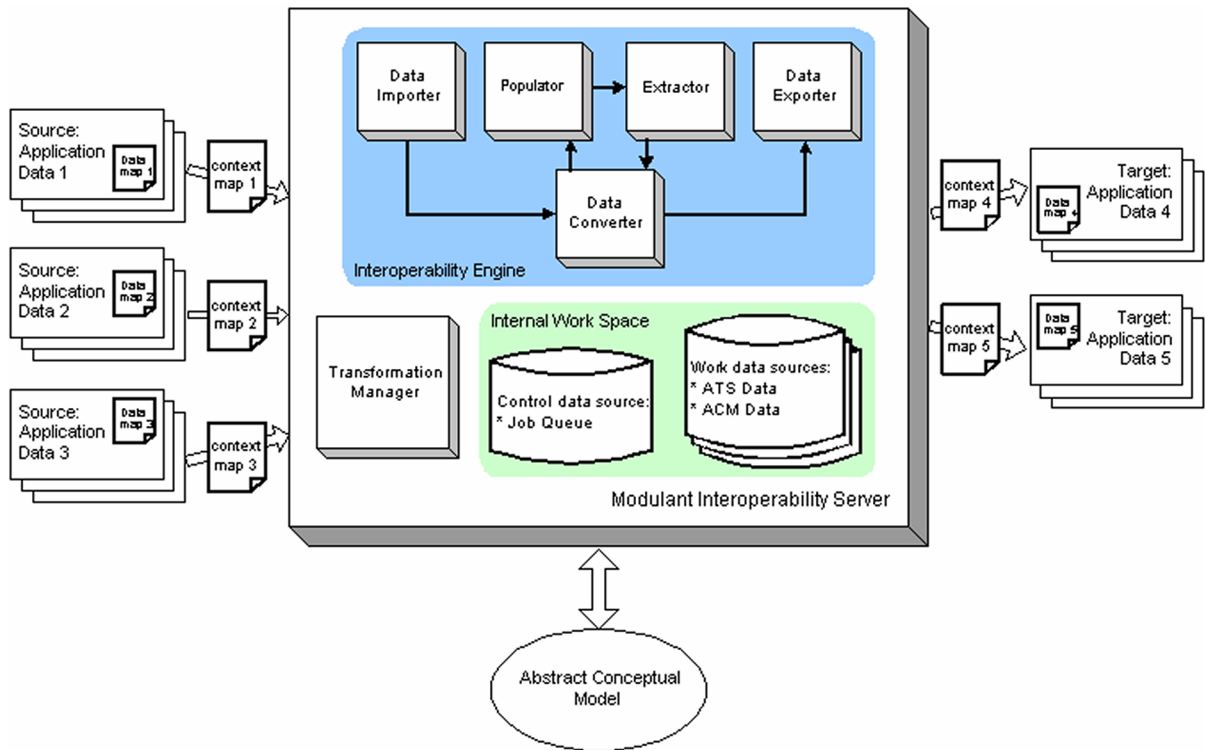
- *JMS Conduit*: The JMS Conduit enables you to send files and commands to a remote server using JMS messages.
- *SOAP Conduit*: The SOAP Conduit enables you to use Web Services to communicate directly with the Interoperability Server, without the need for any API calls.

## Flow of an Interoperability Run

During interoperability runs, the Interoperability Server transfers data from source ATs to the Abstract Conceptual Model and, from there, to the target ATs. Because data is transferred to and from the Abstract Conceptual Model (an abstract schema), meaning is preserved throughout the interoperability run.

Figure 14 shows the basic flow of an interoperability run.

Figure 14: Interoperability Flow



The following steps describe the process of an interoperability run from the perspective of the data files and their interactions with the components of the Interoperability Server and the internal work space:

- 1 The server locates the **control** data source and the **work** data sources.
- 2 The server waits for a request to perform an interoperability run:
  - a If a synchronous run is requested, the server reads the associated run configuration file and begins the run, as described in step 3.
  - b If an asynchronous run is requested, the server sends the run configuration parameters to the Transformation Manager, which adds the job to the queue in the **control** data source. The Transformation Manager initiates the run when the server is ready.



- 3 The Interoperability Server reads the run configuration parameters and identifies the application files required for this interoperability run:
  - ◆ One or more application data files for each source and target application
  - ◆ One or more data map files for each source or target application data file
  - ◆ One or more context map files for each source and target application
  - ◆ One Abstract Conceptual Model for the entire interoperability run
- 4 The Interoperability Server waits until all of these files are available and stores them together for this interoperability run.
- 5 The Data Importer reads each source data map file and uses that information to parse the corresponding application data file and save it in the **work** data source as ATS data.
- 6 For each source of ATS data, if the corresponding context map contains data conversion definitions, the Data Converter performs any necessary preprocessing (input) conversions on the data in that ATS data.
- 7 If the transformation type is **Semantic**, the Populator uses the information in the source context maps to populate the data from each source of ATS data into ACM data in the **work** data source. Otherwise, if the transformation type is **Syntax**, skip to step 9.
  - ◆ **Semantic** transformation performs a full interoperability run, using the full capabilities of the Modulant Contextia Interoperability Server.
  - ◆ **Syntax** transformation includes format conversion only, using the Data Converter, but bypassing the Populator and the Extractor.
- 8 The Extractor uses the information in the target context maps to extract data from the populated ACM data to target ATS data in the **work** data source.
- 9 For each set of target ATS data, if the corresponding context map contains data conversion definitions, the Data Converter performs any necessary postprocessing (output) conversions on the data in that ATS data.
- 10 The Data Exporter reads each target data map file and uses that information to export extracted data from target ATS data into target application data files.

## 4 Moduland Contextia Tools and Components

---

# 5

## An Interoperability Example

Now it's time to look at a real example. This chapter describes two sample travel applications that want to share information with each other. One company is based in the U.S. and the other has offices throughout Europe. Both companies want to advertise the group tours available from each company.

This chapter addresses the following topics:

- Introducing the Sample Applications
- Developing a Mapping Strategy
- Creating the Mapping Specifications
- Performing an Interoperability Run

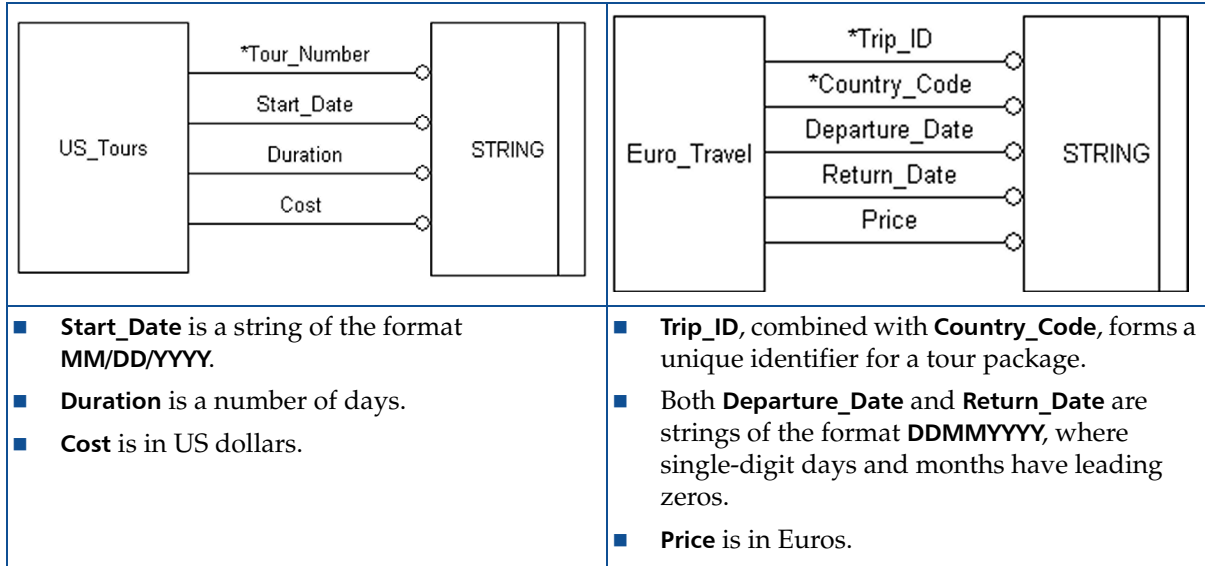
---

### Introducing the Sample Applications

Consider two applications in a sample interoperability environment, US Tours and Euro Travel. Both store information about tour packages, including the length of each trip and its cost. Each company wants to provide information to their customers about tours available from the other company.

Figure 15 shows EXPRESS models that describe the data structures used by each of these travel applications. For information about reading EXPRESS diagrams, see [Appendix A, "EXPRESS-G Language Notation,"](#) in *Using the Context Mapper*.

**Figure 15: US Tours and Euro Travel**



Notice the following things about these applications:

- The US Tours application has a starting date and the duration of a tour package, but does not explicitly store the ending date.
- The Euro Travel application, on the other hand, has a starting date and an ending date, and leaves the duration implicit.
- Both applications list the price of a tour, but use different currencies.
- Both applications list dates, but display them in different formats.
- The Euro Travel application uses a composite key—**Trip\_ID** plus **Country\_Code**—to uniquely identify tour packages.

## Developing a Mapping Strategy

Based on the information in “Introducing the Sample Applications,” a valid mapping strategy for this interoperability environment includes the following considerations:

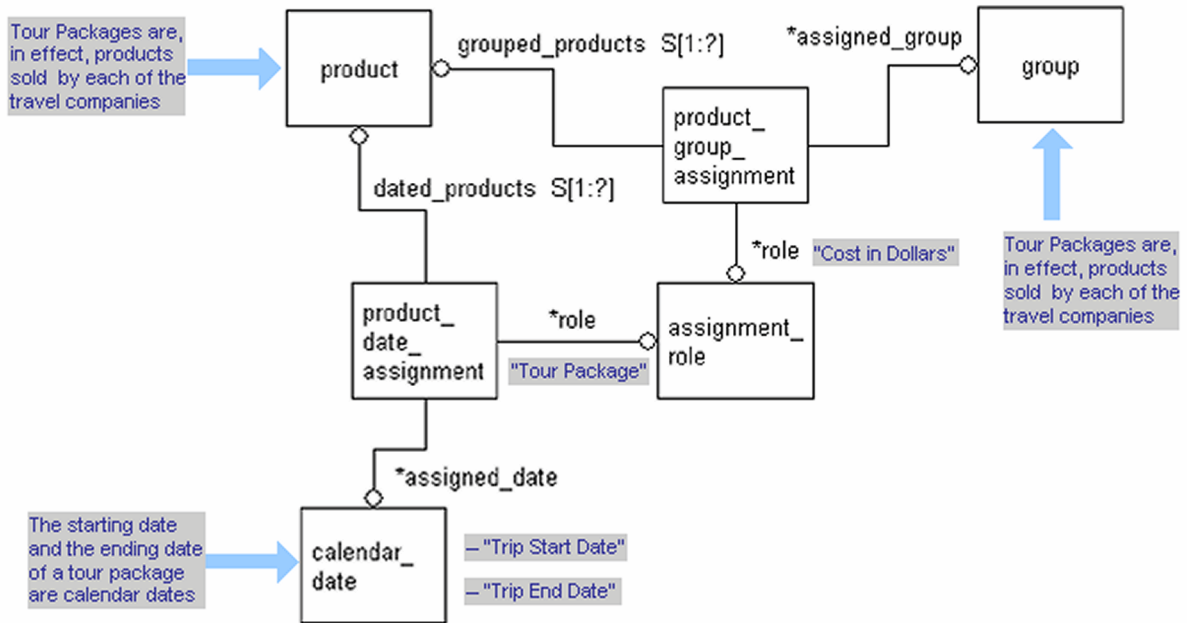
- 1 Identify the major concepts represented by both applications.
  - ◆ *Tour packages*, which can be considered products sold by each company. Using the Modulant Abstract Conceptual Model, you can map the unique identifier for each tour package to the **id** attribute of the abstract **product** entity.

**Note:** The Context Mapper uses the notation *entity.attribute* to identify objects in the ACM; therefore, the mapping target for the identifier of a tour package becomes **product.id**.

- ◆ The *dates* of each tour package you can purchase.  
Using the Modulant ACM, you can map the individual parts of each date to the **day\_component**, **month\_component**, and **year\_component** attributes of the abstract **calendar\_date** entity.
  - ◆ The *cost* of each tour package.  
Using the Modulant ACM, you can view the cost of a tour as a way of classifying the tour package, which enables you to map it to the **name** attribute of the abstract **group** entity.
- 2 Choose common representations for data that must be shared across applications, and convert to and from those representations.
- ◆ Both applications store dates, but each one uses different date formats and each stores different date values. Therefore, you can decide to map only starting and ending dates and calculate the duration when necessary.
  - ◆ Both applications include the price of a tour, using different currencies. Therefore, you can decide to map all prices in US dollars and convert application data values that use other currencies to and from dollars when necessary.
- 3 Identify the conversion definitions necessary to support the common representations you chose.
- ◆ To handle dates, both applications require format conversions to split the date fields into component parts. In addition, the US Tours application requires two calculations.  
On input, you must derive the ending date from the imported starting date and duration values. On output, you must calculate the duration using extracted values of starting and ending dates.  
**Note:** Because Euro Travel already stores both starting and ending dates, you do not need to do any additional calculations for date values in the Euro Travel context map.
  - ◆ To handle currency values, you must convert the price from the native currency to dollars on input. For Euro Travel, this means converting Euros to dollars on input and back to Euros on output.  
**Note:** Because US Tours already stores prices in dollars, you do not need to do anything special to handle currency values in the US Tours context map.

Figure 16 shows the subset of the Abstract Conceptual Model that contains the mapping targets, with a description of the mapping strategy outlined here.

Figure 16: Mapping Strategy for US Tours and Euro Travel



## Creating the Mapping Specifications

In order to be able to share information between the US Tours and Euro Travel applications, you must create both data maps and context maps for each application. Based on the mapping strategy described in the preceding section, you have enough information to accomplish this.

## Describing the Data

From discussions with domain experts, you learn that US Tours stores its data in XML format, and Euro Travel prefers to use delimited flat files for its data.

For example, sample data from US Tours looks like:

```

<ustourdata>
  <ustours>
    <tour_number>A00100</tour_number>
    <start_date>11/01/2001</start_date>
    <duration>10</duration>
    <cost>1</cost>
  </ustours>
</ustourdata>

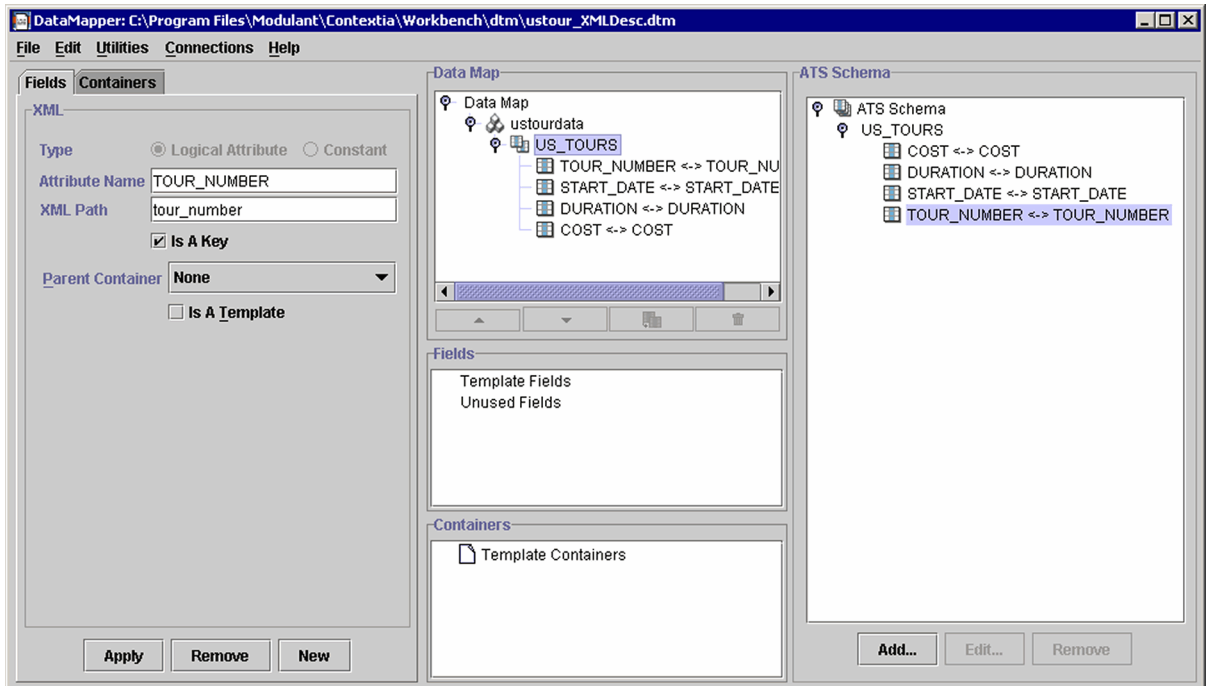
```

```

<ustours>
  <tour_number>A00101</tour_number>
  <start_date>01/01/2002</start_date>
  <duration>20</duration>
  <cost>2</cost>
</ustours>
.
.
.
</ustourdata>

```

The data map corresponding to this data looks like:



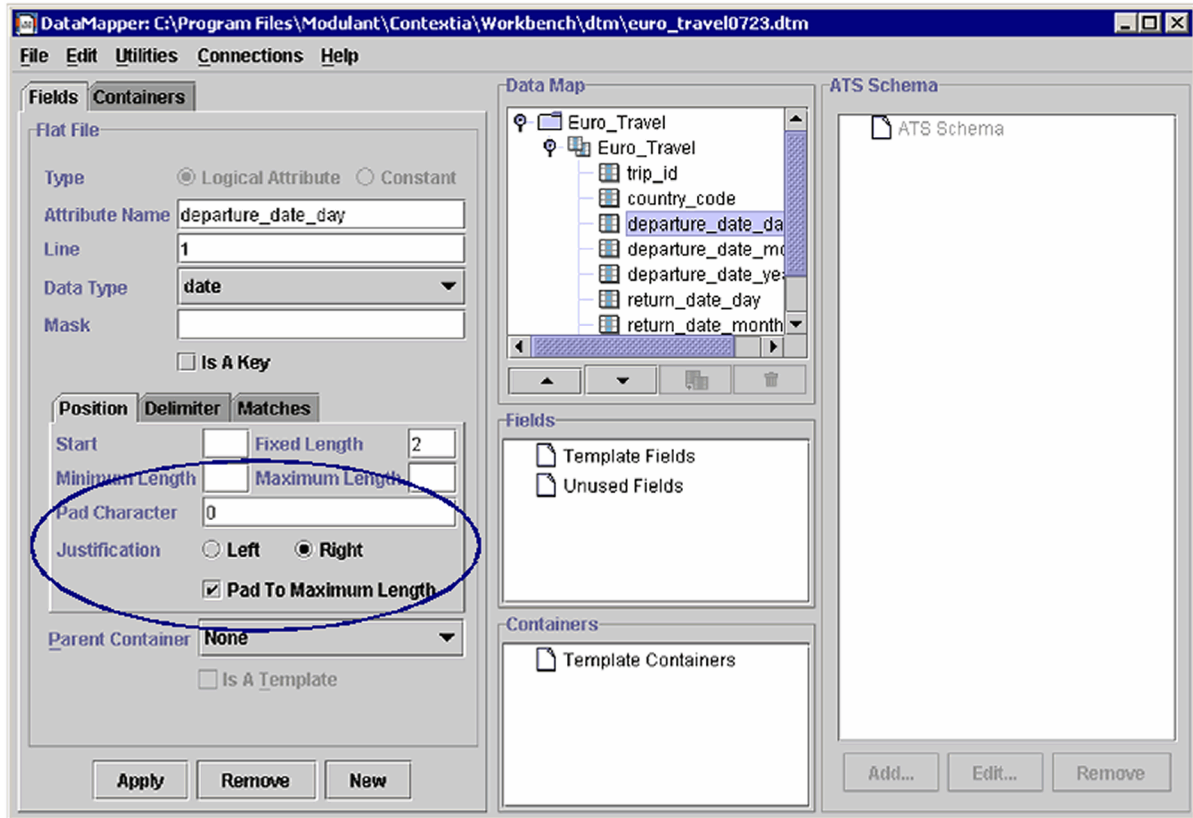
In the XML data structure, the starting date is contained in a single XML element. Therefore, you can use a conversion definition in the context map to split the date into separate month, day, and year parts for mapping to the ACM.

## 5 An Interoperability Example

Sample data from Euro Travel looks like:

```
13412, FR, 21062002, 30062002, 2061.25  
13413, UK, 21062002, 25062002, 1212.50  
13414, CH, 24062002, 26062002, 824.50
```

The data map corresponding to this data looks like:

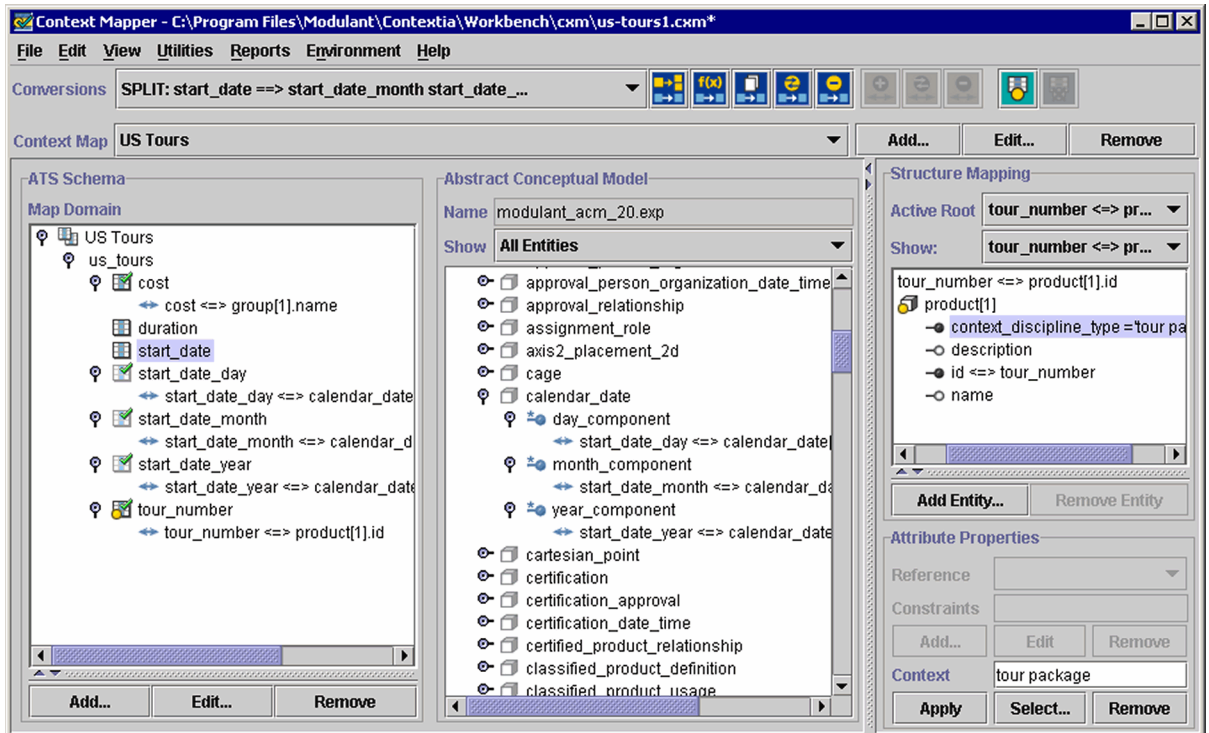


Notice that in the flat-file data structure, the dates appear in individual fields, surrounded by commas. In a flat-file data map, you can specify how to split each date into three parts based on its position in the date field. Using the **Right Justified** and **Pad Character** settings, you can specify that both the day and the month parts have a fixed length of two characters, and a leading zero (the *pad character*) when the value is less than two characters long.



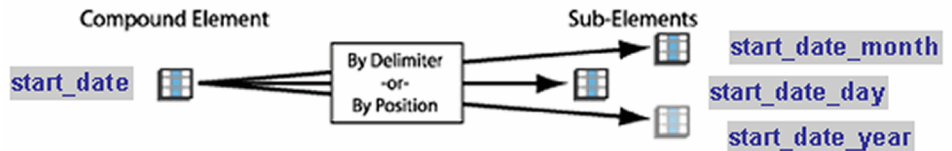
## Describing the Context

According to the mapping strategy, tour packages are considered as products. The following context map shows the mapping statements for the data elements in the ATS schema for US Tours. On the right you can also see the structure mapping for the root data element, **tour\_number**, which is mapped to **product.id**, and includes the context string **tour package** (below the structure mapping).



Notice also at the top left, you can see the conversion definition that separates the single data element **start\_date** into three component parts. Figure 17 shows this format conversion graphically.

Figure 17: Format Conversion to Split start\_date into Parts



The **Abstract Conceptual Model** region in the center of the Context Mapper main window shows how the temporary data elements created in the conversion definition have been mapped to attributes of the abstract entity **calendar\_date**.

## 5 An Interoperability Example

As you define your context maps, you can take advantage of the fact that related context maps are part of an interoperability environment. The Context Mapper shows you the context strings that have been defined in related context maps so that you can reuse existing values to ensure consistency in your definitions of context.

The screenshot shows the 'Select Context' dialog box. The 'Structure Mapping' section on the left displays a tree view of context maps. The 'Find Context In' section has radio buttons for 'Entire Interoperability Environment' and 'Select Files and/or Context Maps', with the latter selected. The 'Show Context For' section has radio buttons for 'All Entities' and 'Entity', with 'All Entities' selected. The 'Attribute' and 'Entity For Reference Attribute' fields are empty. The table below shows the context mappings for the selected files.

CXM File	Context Map	Entity(Usage)	Attribute	Value
euro-travel.cxm	Euro Travel	assignment_r...	name	trip end date
euro-travel.cxm	Euro Travel	assignment_r...	name	trip start date
euro-travel.cxm	Euro Travel	product{1}	context_discip...	tour package
us-tours1.cxm	US Tours	assignment_r...	name	cost in dollars
us-tours1.cxm	US Tours	product{1}	context_discip...	tour package

The 'Context' field at the bottom is set to 'tour package'. Buttons for 'Apply', 'Remove', and 'Close' are visible.

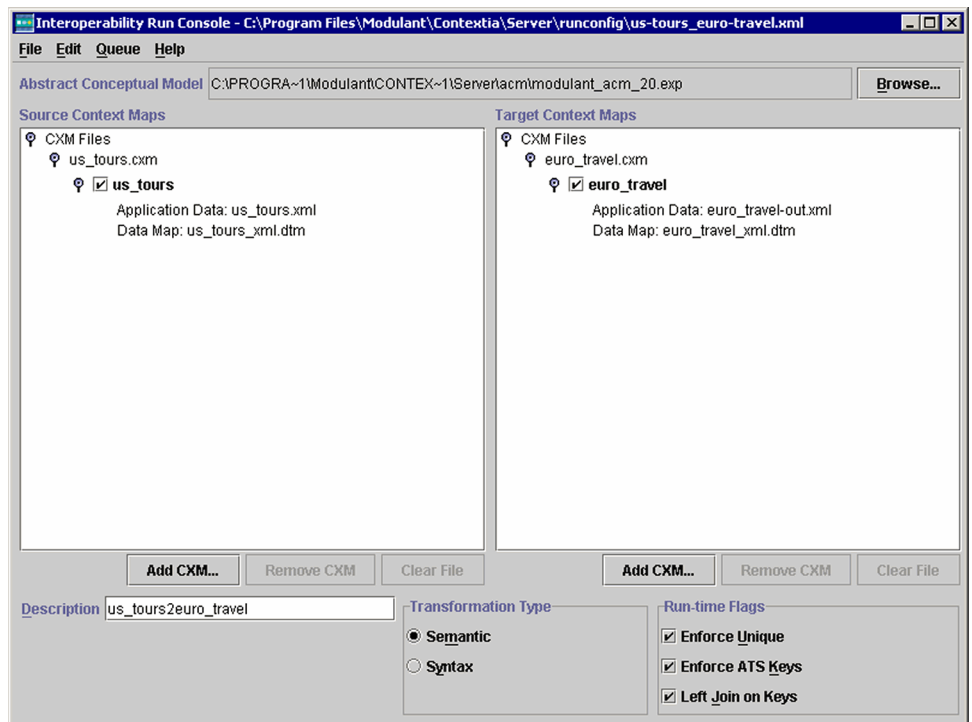
After you have defined context maps for each of the applications in your interoperability environment, you can perform a test interoperability run to verify that your mappings are effective.

## Performing an Interoperability Run

To perform an interoperability run for the purpose of providing information about tour packages offered by US Tours to Euro Travel for display to their customers, you specify the following information for each application:

- the name and location of the CXM file
- the name of the context map to use for the transformation
- the location of the application data:
  - ◆ for the source application (US Tours), you specify where to find the source data
  - ◆ for the target application (Euro Travel), you specify where to put the target data after a successful interoperability run
- the name and location of the data map file that describes the format of the application data

Using the Interoperability Run Console, the configuration looks like this:



## 5 An Interoperability Example

When you start the interoperability run, the Queue Monitor displays the progress of the interoperability job:

The screenshot shows the Queue Monitor application window, which is divided into two sections: 'Active' and 'Completed'. The 'Active' section contains a table with one row of data, and the 'Completed' section contains a table with five rows of data. Both sections have a 'Refresh Now' button and a timer set to 5 seconds.

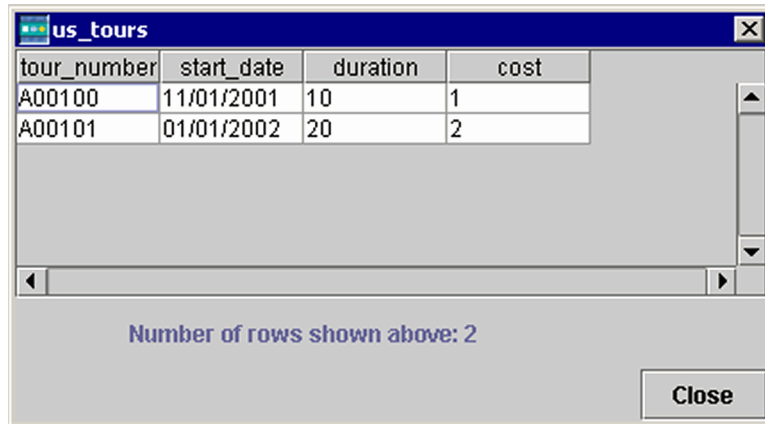
ID	Description	Status	Status Message	Submit Time	Start Time	End Time
64	us_tours2euro...	PROCESSING		Jul 26, 2002 2:...	Jul 26, 2002 2:...	

ID	Description	Status	Status Message	Submit Time	Start Time	End Time
59	Acme2AutoSo...	SUCCESS	Completed su...	Jul 25, 2002 1...	Jul 25, 2002 1...	Jul 25, 2002 1...
60	AcmeAuto test2	SUCCESS	Completed su...	Jul 25, 2002 1...	Jul 25, 2002 1...	Jul 25, 2002 1...
61	AcmeAuto test2	SUCCESS	Completed su...	Jul 25, 2002 1...	Jul 25, 2002 1...	Jul 25, 2002 1...
62	us2self-inline	SUCCESS	Completed su...	Jul 25, 2002 2:...	Jul 25, 2002 2:...	Jul 25, 2002 2:...
63	us2self-inline	SUCCESS	Completed su...	Jul 25, 2002 2:...	Jul 25, 2002 2:...	Jul 25, 2002 2:...

When the interoperability run has been completed successfully, the Queue Monitor moves the job from the **Active** queue to the **Completed** queue. At this point, you can have the Interoperability Server move the target data from its internal cache to the location you specified.

To verify that the interoperability run was successful, you can compare the source and target data values to see that the proper data was transformed correctly. You can use the Interoperability Run Console to display these values. In this example, the source data from the US Tours application looks like:

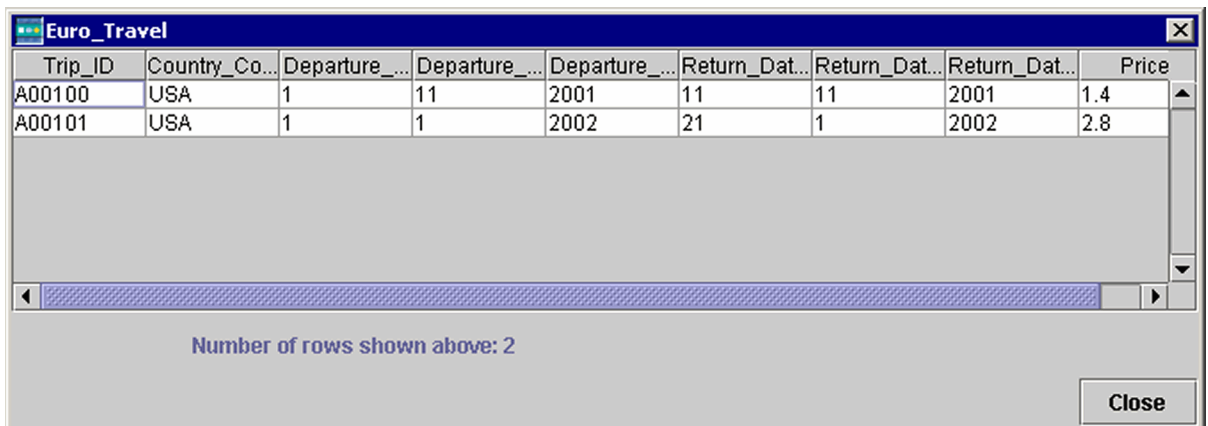


The screenshot shows a window titled 'us\_tours' with a table containing two rows of data. Below the table, it indicates 'Number of rows shown above: 2' and a 'Close' button.

tour_number	start_date	duration	cost
A00100	11/01/2001	10	1
A00101	01/01/2002	20	2

Remember that the US Tours source data supplied to this interoperability run is in XML format, as shown in [“Describing the Data” on page 42](#).

The target data, in the format of the Euro Travel application, looks like:



The screenshot shows a window titled 'Euro\_Travel' with a table containing two rows of data. Below the table, it indicates 'Number of rows shown above: 2' and a 'Close' button.

Trip_ID	Country_Co...	Departure_...	Departure_...	Departure_...	Return_Dat...	Return_Dat...	Return_Dat...	Price
A00100	USA	1	11	2001	11	11	2001	1.4
A00101	USA	1	1	2002	21	1	2002	2.8

Notice that this target data reflects the values from the source data, using the field names of the target application, as defined in the data map. In fact, the data file created by the Interoperability Server at the end of this interoperability run, in flat-file format, looks like:

```
A00100, USA, 01112001, 11112001, 1.4
A00101, USA, 01012002, 21012002, 2.8
```

Even though the source data from US Tours does not include an explicit country code, the context map for this application supplies the string **USA** in the form of context for this data. This information came from the context discovery method.

## 5 An Interoperability Example

---

# 6

## Architectural Considerations

The Interoperability API provides a diverse set of deployment options. This section explores these options and provides details about each one so that you can choose which combination of options best suits the needs of your enterprise.

This section contains the following topics:

- [Synchronous vs. Asynchronous Interoperability Runs](#)
- [Remote vs. Embedded Servers](#)

---

### Synchronous vs. Asynchronous Interoperability Runs

The Interoperability Server provides two ways to execute interoperability runs: synchronously, where the client program waits for the results before proceeding, and asynchronously, where the client program sends the job to the Interoperability Server and continues its operation.

The Interoperability Run Console provides only asynchronous operation. If your application would benefit from synchronous operation, you must implement it using the Interoperability API.

The following topics describe these modes:

- [Synchronous Operation](#)
- [Asynchronous Operation and the Job Queue](#)

### Synchronous Operation

Synchronous execution of interoperability runs provides immediate results to your client program. Using this mode, you do not have to interact with a job queue; instead, the client program initiates an interoperability run, which the Interoperability Server performs using the first available execution thread. This enables greater efficiency for interoperability jobs with small amounts of data, such as transaction-based applications. In addition, if you have defined a large number of working threads, then multi-threaded execution of synchronous jobs provides an even more attractive option.

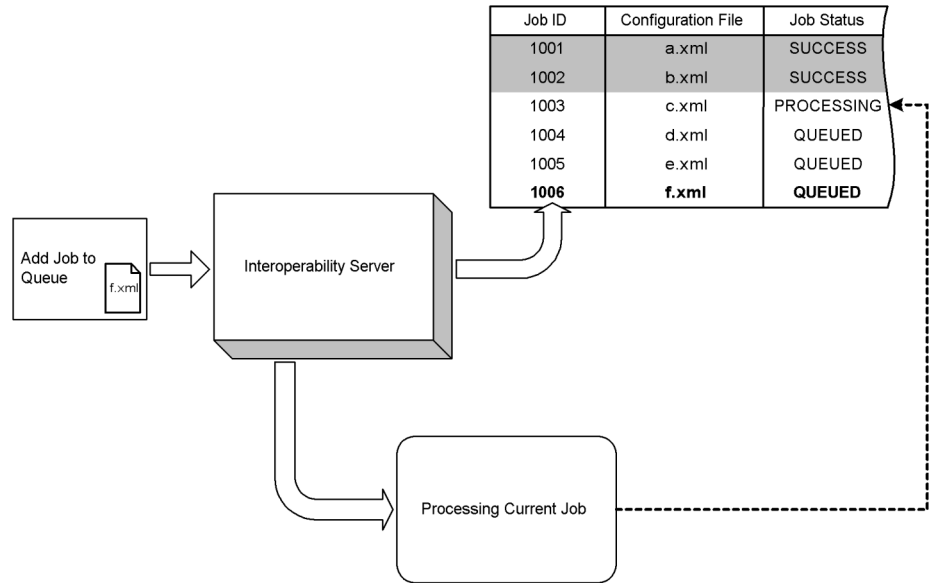
## Asynchronous Operation and the Job Queue

Asynchronous execution lets you run interoperability jobs in the background while your client program continues operation. In this mode, you can queue large interoperability runs and retrieve the results as each job finishes.

When you choose asynchronous operation, the Interoperability Server creates a queue of interoperability jobs. You can write code to query the Interoperability Server to see the status of specific jobs, and also to cancel jobs in the queue.

Figure 18 shows the asynchronous job queue and what happens when you add a new job.

**Figure 18: The Interoperability API Job Queue**




---

## Remote vs. Embedded Servers

The Interoperability API provides a variety of ways for your client program to connect with the Interoperability Server.

This section examines the following configuration options:

- Remote Server Using JMS
- Remote Server Using Web Services and SOAP
- Embedded Server



## Remote Server Using JMS

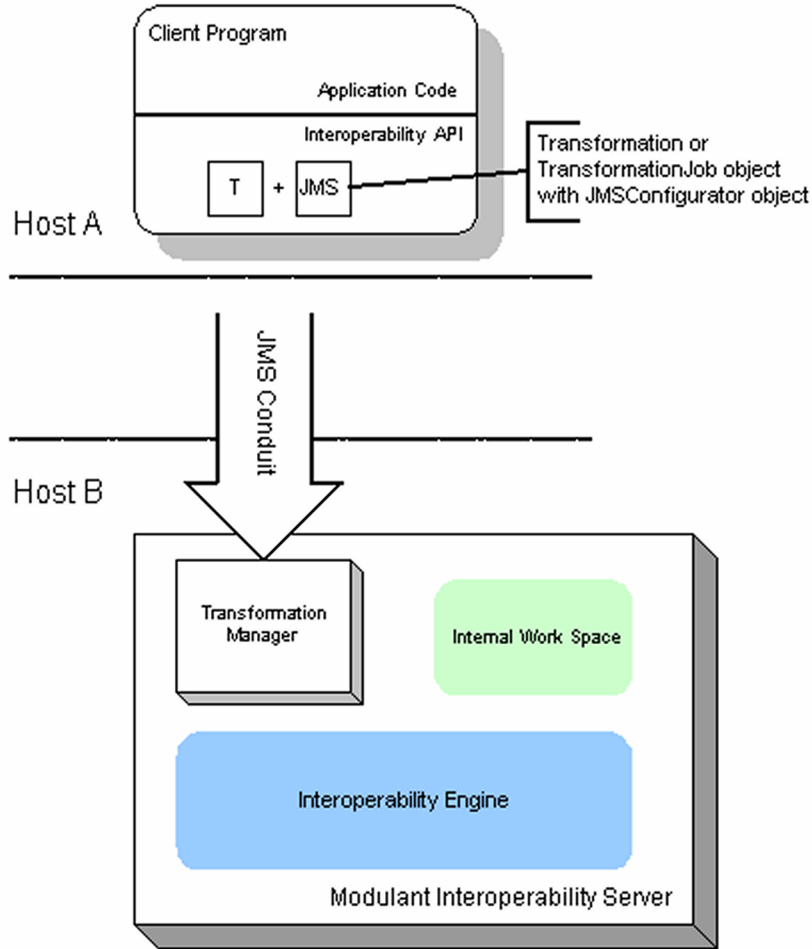
To create a distributed client-server application, you can use JMS (Java Message Service) to communicate between client programs and the Interoperability Server. Using a messaging system allows separate, loosely-coupled clients and servers to reliably communicate asynchronously or synchronously.

The JMS Conduit, part of the Interoperability Server, enables client programs to run transformations from a remote location. This component transfers the necessary data and files between the server and client. The JMS Conduit interacts with the Transformation Manager component of the Interoperability Server to execute the interoperability runs.

The **JMSConfigurator** class selects a JMS Conduit for interacting with the Interoperability Server. Using this configuration option enables a client program to initiate interoperability runs across a network—in a separate JVM than the client application—via JMS.

Figure 19 shows an example of a client program using the JMS Conduit to communicate with the Interoperability Server.

Figure 19: Using the JMS Conduit



## Remote Server Using Web Services and SOAP

SOAP (Simple Object Access Protocol) enables exchange of information in a decentralized, distributed environment, using messages formatted using agreed-upon XML structures.

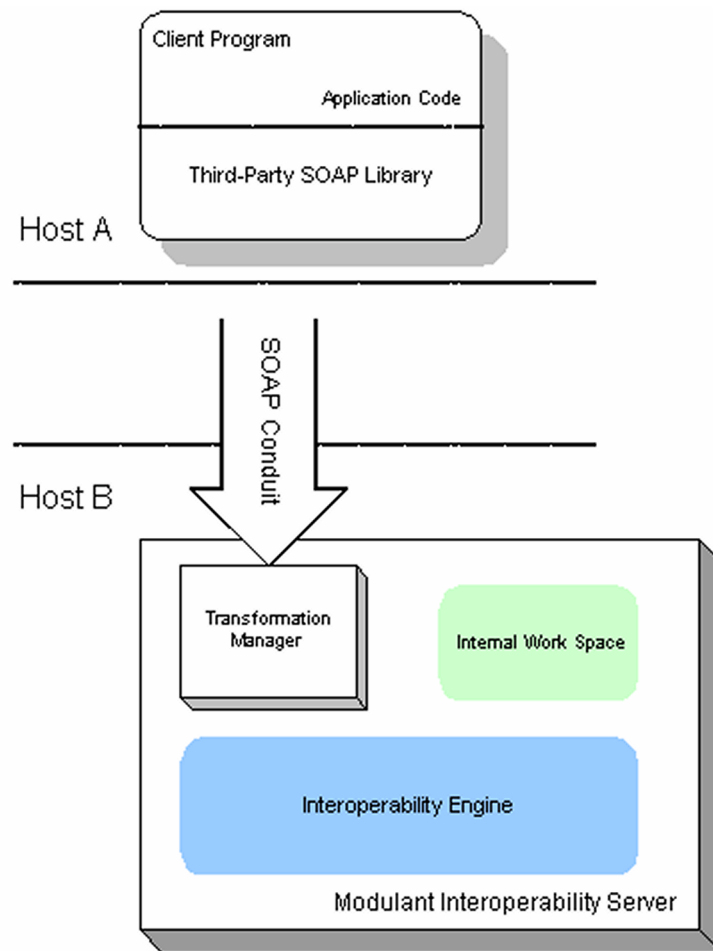
The Interoperability Server contains a SOAP Conduit component, which makes it Web Services-enabled. The SOAP Conduit allows client programs to initiate interoperability runs remotely using SOAP messages.

Client programs create SOAP messages, whose formats are defined in WSDL (Web Services Description Language). In some cases, all of the necessary information appears in the message, and in other cases the message can have additional files attached to it.

Soap client programs can generate SOAP requests either with the Interoperability API or directly according to a supplied WSDL file. The SOAP Conduit, part of the Interoperability Server, receives messages from client programs and decodes them. The Interoperability Server returns a response to the client as a SOAP message.

Figure 20 shows an example of a client program using the SOAP messages to communicate with the Interoperability Server.

**Figure 20: Using the SOAP Conduit**

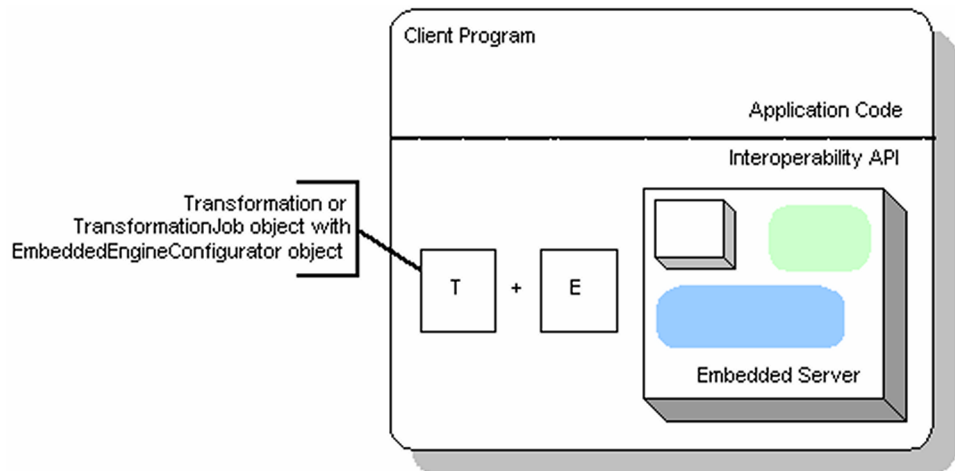


## Embedded Server

Depending on your circumstances, it might be more efficient to perform transformation runs in-process (in the same JVM) as the client program. This eliminates the need for network communication. If you are running the embedded server in several programs at once, only one should have queue dispatching enabled for asynchronous jobs. All of the servers can submit jobs, but only one should manage the queue. All embedded servers can, however, process synchronous jobs independently.

The **EmbeddedEngineConfigurator** class selects an embedded Interoperability Engine. Modulant recommends this option to advanced users only. For more information, see [“EmbeddedEngineConfigurator Class”](#) on page 113.

**Figure 21: Using an Embedded Server**



---

# Glossary

**Abstract Conceptual Model** Also ACM. A description of an information domain (also known as an *ontology*), in the form of a data model, that can completely and accurately represent the information, including *context*, created or used by many different applications.

**ACM** See *Abstract Conceptual Model*.

**application** The functionality of a computer system and the data that it operates on. Within the *CIIM*, an application is the source or destination for information interoperability.

**application context** The environment in which application data is created and used (see also *context*). Without knowledge of application context, the data created and used by an application is not fully understandable and cannot be meaningfully communicated between applications.

**application data** Data instances that are created and used by an *application*.

**application schema** A *schema* that governs or describes the data that is created or used by an application. An application schema can include the names of the data collections and *data elements* used in an application (for example, entities and attributes or tables and columns), the structure of the application data, data types, constraints, etc. Application schemas are external to the Modulant Contextia tools and components.

**application transaction set** . Also ATS. A description of the data that is created or used by an application and is within the scope of an interoperability environment. An ATS serves as the “footprint” of an application in the Modulant toolset.

In the *CIIM*, an ATS is represented by:

- ◆ one or more *data maps* that describe the application’s physical data structure
- ◆ an ATS schema (ATS *data elements*—the logical representation of the application’s physical data structure in the form of entities and attributes)
- ◆ one or more *context maps* that relate the data elements in the ATS schema to an *Abstract Conceptual Model*

Information that contributes to the specification of an ATS includes:

- ◆ an *application schema* that describes the structure of the application data
- ◆ an *application context* description that describes the context of the application data
- ◆ sample *application data* that is used in the process of describing the application context

**ATS** See *application transaction set*.

**ATS data** Representation of a set of application data inside the *internal work space* of the *Interoperability Server*. The logical structure of a set of ATS data is described by an *ATS schema*.

**ATS schema** A description of the logical structure of *application data*, used by the Data Mapper, the Context Mapper, a context map file, and the *Interoperability Server*. An ATS schema consists of a set of data elements described in the form of logical entities and their attributes.

**CIIM** Pronounced *simm*. See *Context-based Information Interoperability Methodology*.

**combine** A type of *format conversion*, in which you create a *compound element* by concatenating two or more *data elements* in the domain of a *context map*.

**comma-separated values** Also CSV. A file format in which text fields in the file are separated by commas.

**compound element** An ATS *data element* that is composed of multiple *sub-elements*, as defined by a *format conversion*.

**compound mapping** Creating more than one *mapping statement* for a single *data element* in a *map domain*.

**constraint** A restriction on the values of an ACM *terminal attribute* in a *structure mapping* in a *context map*.

**context** The environment in which data is created and used. Context provides a frame of reference for data and includes the users and processes that create and use data, together with the background knowledge, assumptions, and terminology that contribute to the meaning of data. See also *semantics*.

**Context-based Information Interoperability Methodology** Also *CIIM*. The Modulant methodology that enables information interoperability by explicitly recognizing and accommodating the information and its context as it is used and produced by applications.

**context accommodation** One of the methods of the *CIIM*. The process of using an *Abstract Conceptual Model* to precisely capture the application information (including *context*) described by one or more *ATSs*.

The context accommodation method consists of relating the *information units* in an *ATS schema* to *structure mappings* in one or more *context maps*.

**context formalization** One of the methods of the *CIIM*. The process of identifying sets of *data elements* (including *root data elements*) that characterize the definitions, aggregations, and business process usage of application data.

The context formalization method consists of analysis of sample application data and application data descriptions (including application schemas) to create an *ATS schema*, and results in an understanding of the *application context* sufficient to apply the *context accommodation* method.

**context map** A formal, computable representation of an *application transaction set*. A context map starts with an *ATS schema* and contains a reference to an *Abstract Conceptual Model* and *mapping statements* for each data element in the *map domain*, including the *map structure*. Context maps are stored with the *ATS schema* in *CXM* files.

**Context Mapper** Part of the Modulant Contextia Interoperability Workbench, a tool that enables an *interoperability architect* to map the data and the semantics of any application system to an abstract representation.

**control data source** Part of the *internal work space* used by the *Interoperability Server*, the **control** data source stores a queue of asynchronous *interoperability runs*.

**conversion definition** Includes both *format conversion* and *data conversion*. Definition of a function that computes the values of one or more data elements in the domain of a *context map* from the values of one or more other data elements.

**CSV** See *comma-separated values*.

**CXM** A context map file in XML format.

**data** Symbols, such as numbers and characters, without inherent meaning.

**data conversion** A *formula* that defines arithmetic operations to perform on one or more data elements in the domain of a *context map* (for example, to create derived attributes).

**Data Converter** Part of the *Interoperability Server*, the Data Converter performs transformations on ATS data in the *internal work space*.

The Interoperability Server invokes the Data Converter at two different stages of an *interoperability run*:

- ◆ As a pre-processor, performing transformations on imported source data before the *Populator* transfers the data to the *Abstract Conceptual Model*.
- ◆ As a post-processor to transform *ATS data* after the *Extractor* transfers the data from the ACM in preparation for exporting that data to target applications.

**data element** Also known as *ATS data element*. An attribute of a logical entity in an *ATS schema*. A data element that identifies the subject of a collection of related data elements can be a *root data element* for the purpose of mapping.

**Data Exporter** Part of the *Interoperability Server*, the Data Exporter moves target *ATS data* from the *internal work space* to target applications after a successful *interoperability run*, using the instructions in a *data map*.

**Data Importer** Part of the *Interoperability Server*, the Data Importer moves source application data into the *internal work space* as *ATS data*, using the instructions in a *data map*, as the first step of performing an *interoperability run*.

**data map** A file in XML format that describes the physical layout of application data files to be used as the source or target of a *interoperability runs*.

**delimited file** A file that uses one or more ASCII characters to identify the boundaries of field values in a file. See also *comma-separated values*.

**domain expert** A person who understands the context and relationships of the data structures in an application domain, and who works with *interoperability architects* to map the data structures from an *application transaction set* in that domain to an *Abstract Conceptual Model*.

**element structure** The *structure mapping* for a single *mapping statement*—a subset of the *map structure*.

An element structure consists of the *mapping target* for the *data element*, the mapping target for its *root data element*, an unambiguous network within the ACM that connects these mapping targets, and any constraints necessary to represent the contextual meaning of the data element.

**EXPRESS** A data specification language, defined as a part of the STEP standard (ISO 10303–11:1994).

EXPRESS has both lexical and graphical forms. EXPRESS is used in the *CIIM* to specify the *Abstract Conceptual Model* and is one of several possible methods of specifying an *application schema*. See also *STEP*.

**EXPRESS-G** The graphical form of the *EXPRESS* language.

**Extractor** Part of the *Interoperability Server*, the Extractor transfers data from the *Abstract Conceptual Model* to a target *application transaction set* during an *interoperability run*, based on one or more *context maps*.

**FirstSTEP XG** A Modulant tool for generating and editing EXPRESS models. This tool was created by *PDIT*.

**FirstSTEP EXML** A Modulant tool that generates a DTD (Document Type Definition) in a specific format from an EXPRESS schema. This tool was created by *PDIT*.

**format conversion** A *conversion definition* in which the format of a *data element* is specified in terms of other data elements. For example, you can define a format conversion to specify the format of a date in terms of three other data elements (day, month, and year).



There are two types of format conversions:

- ◆ *split*, where you start with a *compound element* and break it into *sub-elements*
- ◆ *combine*, where you define a compound element by concatenating existing data elements

**formula** An expression that defines arithmetic operations to perform on one or more data elements in the domain of a *context map* (for example, to create derived attributes). See also *data conversion*.

**information** The meaning derived from *data* about objects (such as facts, events, things, processes, ideas, and concepts) through the interpretation of the data within a certain *context*.

**information unit** An atomic fact about an identified concept or subject of interest. In the *CIIM*, an information unit is the combination of a *data element* and its associated *root data element*.

**input conversion** A *data conversion* to perform during the pre-processing phase of an *interoperability run*. See also *Data Converter*.

**input mapping statement** A *mapping statement* that tells the *Interoperability Server* how to populate an attribute of an *Abstract Conceptual Model* from a *data element* in the domain of a *context map*. Input mapping statements are processed by the *Populator* and ignored by the *Extractor*.

**integration** Combining software and hardware components into an overall system in a tightly coupled manner.

**internal work space** Used by the *Interoperability Server*, the internal work space consists of two types of data sources: one to manage the job queue (known as the *control data source*), and one to store internal data for *interoperability runs* (known as *work data sources*)

**interoperability** The ability of people, organizations, and systems to work together.

**Interoperability API** A Java API that lets you perform *interoperability runs* programmatically. Using the Interoperability API, you can define jobs that execute interoperability runs and queue them to run independently.

**interoperability architect** An expert in the *Context-based Information Interoperability Methodology*. An interoperability architect understands the structure of the *Abstract Conceptual Model* thoroughly enough to see how the structure of an *ATS schema* maps to it. Interoperability architects work closely with *domain experts* to understand the specifics of a particular application's structure and context.

**Interoperability CL** A command-line tool for initiating and monitoring *interoperability run*.

**interoperability environment** A collection of applications that must share information with each other.

**interoperability run** The Modulant process in which the *Interoperability Server* transforms data from one or more source systems through an *Abstract Conceptual Model* to one or more target systems.

**Interoperability Run Console** A client tool that supports the *Interoperability Server* and enables users to design, perform, and analyze *interoperability runs*.

**Interoperability Server** Part of the Modulant Contextia Interoperability Platform that populates data from source *application transaction sets* to the *Abstract Conceptual Model* and extracts the transformed data to the target application transaction set.

See also *Data Importer*, *Data Converter*, *Populator*, *Extractor*, *Data Exporter*, *Transformation Manager*, *JMS conduit*, and *SOAP conduit*.

**interoperability strategy** An agreed-upon approach used by *interoperability architects* that will ensure consistent mappings, based on analysis of common application information requirements, for the applications in an *interoperability environment*. Also known as *mapping strategy*.

**JMS conduit** Part of the *Interoperability Server* that communicates with client programs using the Java Message Service.

**management resource** Entities in the Modulant *Abstract Conceptual Model* that describe information that can apply to any other type of entities. This category includes information about dates, documents, people, organizations, and groups.

**map domain** The set of *data elements* whose context is described by a *context map*.

**mapping** The process of specifying the relationships between *data elements* in the domain of a *context map* and the entities and attributes of an *Abstract Conceptual Model*.

**mapping direction** Property of a *mapping statement* that indicates whether to use the mapping information only to populate the *Abstract Conceptual Model* (where the direction is *input*), only to extract information from a populated ACM (where the direction is *output*), or both (where the direction is *input/output*).

**mapping statement** A part of a *context map* that associates a *data element* with a *mapping target*. Each mapping statement is associated with part of a *structure mapping*, called the *element structure*.

**mapping strategy** See *interoperability strategy*.

**mapping target** Part of a *mapping statement*. An attribute of the *Abstract Conceptual Model* that is associated with a *data element* in the *ATS schema*.

**map structure** A set of *element structures* that form the *structure mapping* for a set of related *data elements* in a *context map*.

For a collection of related data elements (a *map domain*), the structure mapping includes the context and relationships of all of those data elements, and is therefore the union of the individual *element structures*.

- meta-data** Data about data. Meta-data can include descriptive information about the context, quality, and condition, or characteristics of the data. In the *CIIM*, a *context map* describes how the ACM is used to represent application data values and meta-data that capture the meaning, including *context*, of the *application data*.
- ontology** An explicit representation of the objects and concepts that are assumed to exist in some area of interest and the relationships that hold among them.
- output conversion** A *data conversion* to perform during the post-processing phase of an *interoperability run*. See also *Data Converter*.
- output mapping statement** A *mapping statement* that tells the *Interoperability Server* how to extract the value of an ACM attribute into an ATS *data element*. Output mapping statements are processed by the *Extractor* and ignored by the *Populator*.
- PDIT** Product Data Integration Technologies, Inc. A leader in the field of interoperability. A wholly-owned subsidiary of Modulant Solutions, Inc. that created the tools *FirstSTEP XG* and *FirstSTEP EXML*.
- Populator** A part of the *Interoperability Server* that transfers imported source data to an *Abstract Conceptual Model* based on one or more *context maps*.
- reference attribute** An attribute used to relate entities in an *Abstract Conceptual Model*. A reference attribute allows an entity to refer to another entity.
- root data element** A *data element* that represents the subject or central concept of a collection of related *data elements* in a *map domain*. The *mapping target* of a root data element serves as the anchor for the *structure mapping* of other data elements that are related to that root data element.
- root mapping statement** The combination of a *root data element* and its *mapping target* in the ACM. A root mapping statement is the starting point of the *structure mapping* for the data elements that are directly related to the root data element.
- root structure** The part of a *map structure* that defines the context and relationships for the *element structures* of *data elements* directly related to a *root data element*.
- schema** A representation, in a formal syntax, of the structure and data types for a collection of related data.
- Semantic Transformation Method** A patent-pending structural data mapping methodology developed by *PDIT*. It involves transformation of data between two or more software programs. It includes both syntactic and semantic transformation. It forms the core of the Modulant *CIIM*.
- semantics** The meaning of *data* within appropriate *context*.
- SOAP conduit** Part of the *Interoperability Server* that communicates with client programs using SOAP (Simple Object Access Protocol) messages.

- source system** An application that is the source of data to be transformed in an *interoperability run*.
- split** A type of *format conversion*, in which you separate a *compound element* into two or more *sub-elements*.
- STEP** See *Standard for the Exchange of Product Model Data*.
- Standard for the Exchange of Product Model Data** International Standard ISO 10303 “Product data representation and exchange,” known colloquially as STEP. STEP is a large, multi-part standard for exchange of data among engineering applications in several different industry sectors. Modulant has enhanced several components of STEP. The *EXPRESS* language is part of STEP.
- structure mapping** Specification of the ACM usage that represents application data and its context. This ACM usage represents the relationship between a data element and its root data element, together with additional population requirements and constraints that capture the contextual information necessary to enable interoperability. See also *element structure*, *map structure*, *root structure*.
- sub-element** Result of the decomposition of a *compound element* into multiple parts based on a *format conversion* definition. See also *split*, *combine*.
- subtype** A specialization of a *supertype* in a data *schema*. A subtype inherits all of the properties (attributes, constraints, and meaning) of its supertype. An entity can be a subtype of more than one *supertype* (multiple inheritance). An entity can be both a supertype of one or more entities and a subtype of one or more entities.
- supertype** A generalized entity from which other entities (known as *subtypes*) inherit properties (attributes, constraints, and meaning). An entity can be both a supertype of one or more entities and a subtype of one or more entities.
- syntax** Rules that define how symbols can be combined independent of meaning.
- target system** An application that is the destination for data to be transformed in an *interoperability run*.
- temporary data element** A *data element* in a *map domain* that is created as part of a *conversion definition*. Temporary data elements do not appear in the *application schema*.
- terminal attribute** An attribute of an *Abstract Conceptual Model* whose data type is a simple data type, such as **STRING** or **INTEGER**. Terminal attributes can become *mapping targets* during the mapping process. These are the attributes into which application data can be populated.
- Transformation Manager** Part of the *Interoperability Server* that manages the processing of *interoperability runs*.

**work data source** Part of the *internal work space* used by the *Interoperability Server*, work data store internal data for *interoperability runs*.



---

# Index

## A

- Abstract Conceptual Models
  - about 17
  - developing 16
- application data
  - about 19
- application information analysis 11
- application schema files 19
- asynchronous interoperability runs 30
  - using the Interoperability Run Console 29
- ATS
  - about 18
- ATS schemas
  - about 20
  - creating 22

## B

- business process context analysis 11

## C

- command-line tools
  - icmd** 30
- context 6
- context accommodation 12
- context discovery methods 11
- context formalization 12
- context map domains 10
- context maps 20
  - about 20
- Context-based Information Interoperability
  - Methodology 7
    - architecture 9
    - framework 8
    - methods 11
    - principles 7
- conversion definitions 22
- CXM files 20
  - about 20

## D

- data conversions 12
- Data Converter 35
- data elements 10
- Data Exporter 35
- Data Importer 34
- data maps
  - creating 20
  - formats 26
- defining population rules 13
- domains, of context maps 10

## E

- embedded server
  - using 56
- Enforce ATS Keys** flag 29
- Enforce Unique** flag 29
- EXPRESS modeling language 18
- extraction rules 13
- Extractor 35

## F

- FirstSTEP EXML 28
- FirstSTEP XG 28
- format conversions 12

## I

- icmd** 30
- information aggregations 10
- information interoperability 2
- information units 10
- internal work space 34
- Interoperability CL 30
- interoperability runs
  - flow 36
  - performing 24
  - required files 23
  - synchronous vs. asynchronous 30
- Interoperability Server
  - components 33

## Index

- internal work space 34
- run-time flags 29

interoperability strategy, *see* mapping strategy

ISO standard 10303 18

## J

JMS Conduit 35

job queue

- lifecycle of an asynchronous job 52

## L

**Left Join on Keys** flag 29

lifecycle of queued job 52

## M

map domains 10

mapping process

- basic example 21
- creating ATS schemas 22
- specifying conversion definitions 22
- specifying mapping targets 22

mapping strategy 16

- example 40
- for US Tours and Euro Travel 42

## P

PDIT 28

performing interoperability runs 24

population rules 13

Populator 35

## Q

queued job

- lifecycle 52

## R

root data elements 10

root structures 10

run-time flags 29

## S

**Semantic** transformations 29

sidebars

- about sidebars xi

- EXPRESS modeling language 18

SOAP Conduit 35

- specifying conversion definitions 22
- specifying mapping targets 22

STEP methodology 18

structure mappings

- root structures 10

synchronous interoperability runs 30

**Syntax** transformations 29

## T

Transformation Manager 34

transformation types 29