

eMerge



SAPIENS

***eMerge
Course Catalog***

July 2010

4 July 2010

Copyright © 1982–2010 Sapiens International Corporation N.V.

All rights reserved.

Information in this document is provided for informational purposes only, is subject to change without notice, and does not represent a commitment on the part of Sapiens Technologies (1982) Ltd. ("Sapiens Technologies") or Sapiens International Corporation N.V. ("Sapiens International"). This manual, and the software described in it, are furnished under a license agreement or nondisclosure agreement. The software may only be used or copied in accordance with the terms of such agreement. Except as may be otherwise permitted by such agreement, no part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Sapiens Technologies. Neither Sapiens International nor Sapiens Technologies assumes any responsibility or liability for any errors or inaccuracies that may appear in this manual.

Sapiens Technologies is an indirect wholly owned subsidiary of Sapiens International.

SAPIENS is a registered trademark and Sapiens *eMerge* (including *Development Workbench*, *Modeler*, *i.way*, *Business Integrity Server*, *RuleScribe*, and *Legacy Adapter*), *FALCON*, *Sapiens Euro Migration*, and *Sapiens INSIGHT* are trademarks of Sapiens International. Other brands and product names are trademarks of their respective holders. Brand and product names are mentioned herein for reference purposes only.

Courses



The following table shows the courses that one can take in order to learn *eMerge*.

- ❑ The *Comprehensive Developer* course focuses on how to develop *eMerge* applications covering data structure, logic and presentation. Participants learn how to develop *eMerge* applications within the *Development Workbench* environment using *Modeler*, *Rule Editor* and *Form Editor*.
- ❑ For those interested in enabling non-*eMerge* user access to the world of *eMerge*, the *Business Components for .NET and Java* course explains how *Business Components* allows Java, .NET, and Java EE clients access to *eMerge*. The *XML Messaging and SOA* course explains how *XML Messaging* enables *eMerge* applications to expose a set of rule-based services to other parties, as well as invoke message based services from other parties.
- ❑ For administrators, system programmers and *eMerge* project leaders who need to know about the installation, maintenance, and tuning of an *eMerge* site, the *Business Integrity Server Administration*, *Change Management*, and *Security* courses are very useful.

| Course | | Development | Administration |
|---|-----------|-------------|----------------|
| Introduction to eMerge Principles | 4 | ✓ | |
| Comprehensive Developer | 5 | ✓ | |
| Business Components for .NET and Java | 6 | ✓ | |
| <i>XML Messaging and SOA</i> | 8 | ✓ | |
| eMerge Queries | 10 | ✓ | |
| Business Rule Language (BRL) Workshop | 15 | ✓ | |
| Legacy Adapter | 11 | ✓ | ✓ |
| DBMS Adapter: SQL | 13 | ✓ | ✓ |
| Business Integrity Server Administration | 18 | | ✓ |
| Change Management | 19 | | ✓ |
| Security | 21 | | ✓ |

Introduction to eMerge Principles

| | |
|----------------------|---|
| Duration | 1 day |
| Description | Presents a short description of <i>eMerge</i> features and capabilities: application development with <i>eMerge</i> , concept model, concepts and fields, development life-cycle, testing, business rules, presentation layer, report generation and open architecture. |
| Audience | People who want to get familiar with the <i>eMerge</i> world. |
| Prerequisites | None |

Contents

Application Development with eMerge

- Development Activities
- What is an Object?
- Parent Concepts vs. Dependent Concepts

Getting Started with Concept Model

- Develop Basic Objects
- One-to-One Weak Entity
- Changing the Probability
- Insert a Document
- Insert an Association
- Insert a Dependency Association (Link)
- The Resulting Model

Concept Fields

- Concept Fields
- Concept Field Characteristics
- Define the Concept Fields
- The *eMerge* Options
- Accessing the Data via Messages
- Message Generation
- Message Results
- Operation
- Operation Codes
- Development Workbench Window
- Parts of a Form

Physical Implementation

- The Development Life Cycle
- Development Stages
- Data Organization in the Database
- Data Element Implementation
- Test the Working Model

Presentation

- The Development Life Cycle
- Presentation
- What Application Presentation Requires
- What is a Form?
- Use the Form Editor
- Open the Form Editor
- Define a Button
- Back to Run Mode
- Testing the Form

Implement Business Behavior

- Rule-Based Development

- The Power of Inference Engine
- Example
- What Are Business Rules
- What is an Event
- Triggering Objects
- What is a Rule Model?
- Access Rule Model from Concept Model
- Add a Ruleset
- Adding Rules
- Additional Information
- Rule Editor
- Formatting Display for Rule Text
- Prompting for Functions
- Checking Rule Syntax

Report Generation using QUIX

- QUIX
- Quix Environment
- QUIX SQL
- Create a Query
- QUIX Language
- Selection Examples
- Modifying the Selection
- The PRINT Statement
- The QUIX-FORM
- The Operation Code

Open Architecture

- Open Architecture
- *eMerge* World
- Create SQL Tables
- Import - External DBMS
- Custom Adapters
- Legacy Adapter Architecture
- *i.way* Interaction Server
- What is *i.way*?
- Using *i.way*
- *i.way* Multi-Tier Architecture
- What is *eMerge Business Components*?
- *Business Components* Features
- *eMerge* Messaging Highlights
- Messaging Data Model
- Development Activities
- *eMerge* Layers

Comprehensive Developer

| | |
|----------------------|--|
| Duration | 10 days |
| Description | Focuses on the fundamental principles of <i>eMerge</i> . Participants learn how to develop <i>eMerge</i> applications within the <i>Development Workbench</i> environment using <i>Modeler</i> and <i>Form Editor</i> . The concept and rule models are covered. Provides practical hands-on experience developing an application, covering all stages of the development cycle. |
| Audience | IT professionals with experience with application development and data structure and with knowledge of the principles of application development, presentation and UI design. |
| Prerequisites | Knowledge of databases and various environments and C++, JavaScript, ActiveX, Internet/Intranet environments, etc. is recommended. |

Contents

Introduction

- Introduction
- Data objects - concepts
- Application development
- *Development Workbench*
- Development stages

Application Data

- Concept Model
- *Modeler*
- Concept fields and domains
- Design the Concept Model
- Implement the Concept Model
- Accessing the data

Application Presentation

- Presentation
- Data form definition
- Form Editor
- Block definition
- Fields in a form
- Actions & flows
- Information transfer

- Dynamic error control
- Form customization

Application Logic

- Introduction to business rules
- Rule Model
- Rule Editor
- Debugging rules
- Computation and validation rules
- Call Rule and conditional checking
- Fetch Rule
- *eMerge* Functions
- Non-Concept Triggering Objects
- Repeated operations
- Derivation Rule

Advanced Features

- *i.way* overview
- Components
- Documentation tools
- Program and batch processing
- Troubleshooting
- Refinement tools
- Introduction to *DBMS Adapter*
- Introduction to security and privacy

Business Components for .NET and Java

| | |
|----------------------|---|
| Duration | 2 days |
| Description | Opens the world of <i>Business Integrity Server</i> to non-Sapiens users. Enables participants to develop <i>eMerge</i> -based client software without a familiarity with <i>eMerge</i> . Subjects covered include: Platform and protocol support, session management, <i>Business Components</i> processing, packaging and reusability, interface and methods, development and runtime procedures. |
| Audience | Most of the course is for IT professionals involved in the development of Java-based user applications. The Development section is for <i>eMerge</i> developers. |
| Prerequisites | Knowledge of Java EE for IT professionals. Knowledge of <i>eMerge</i> for the Development section. |

Contents

Business Components Overview

- Integration Technologies
- Sapiens Solution for Integrating Technologies
- What is an eMerge Component
- BIS and eMerge Component Clients
- BC Connector
- BC Processing Steps
- BIS and BC Connector
- Development Methodology 1
- Development Methodology 2
- Components and Operations

Operation Methods

- Operation Methods
- Operation Method Definitions
- Operation Method Types
- Define Operation Methods

eMerge Component Definitions

- Component Package
- eMerge Component
- Component Instance
- Component Properties
- Component Key
- Component Home Methods
- Component Delegation Methods
- Compiling Components
- Events
- Service Component
- Package Generation

eMerge Development

- Development Methodology
- Step 1 - Define the BC Package
- Step 2 - Define Components for the Package
- Step 3 - Define Home Methods

- Step 3 - Define Delegation Methods
- Step 4 - Define New Composed Component
- Step 5 - Compile the Components
- Step 6 - Generate the Package

BC Tester

- BC Tester
- BC Tester Functionality

BC Connector - Interface Model

- BC Connector Overview
- BIS & BC Connector
- Ebc Interface Model
- EbcConnector
- BC Connector Properties
- IEbcConnector
- IEbcConnection
- IEbcHome
- IEbcObject
- I<ComponentName>Home
- I<ComponentName>
- Descriptor Interfaces
- Working with BC Connector

BC Connector - Features & Functionality

- BC Connector Features & Functionality
- Multi-Framework Support
- Connection Management
- Transaction Management
- Security & Authentication
- Typed Interfaces
- Non-Typed Interfaces
- Cache Management
- Component Metadata
- Events
- Sending Messages
- Session Handling
- Logging

- JCA 1.0 Compatible
- Other Component Indications
- Data Integrity

BC Connector Java EE

- Topics
- About Java EE
- JCA Goals
- Java EE Connector Architecture Contracts

- Model
- JCA Integration Services
- BC Connector for Java EE
- Connection Management
- Transaction Management
- Security Management
- Common Client Interface
- Summary

XML Messaging and SOA

| | |
|----------------------|---|
| Duration | 2 days |
| Description | Subjects covered include: XML, <i>eMerge XML Messaging</i> , <i>Web Services Adapter</i> , importing XML schemas, mapping a schema to a components, defining documents and parties, <i>XML Tester</i> , creating component and post rules and modular applications. |
| Audience | <i>eMerge</i> developers. |
| Prerequisites | Business Components for .NET and Java. |

Contents

Introduction

- Integration Technologies 1-3
- Business Document Messaging Requirements
- What is XML?
- XML Document
- XML Element
- XML Attributes
- XML Syntax Rules
- Namespaces
- Additional XML Features
- DTD – Document Type Definition
- XML Schema (xsd)
- XML Document
- XML Schema
- Schema Elements
- Named / Anonymous Types
- Global / Local Elements
- Occurrence Constraints
- Content Models
- Schema Target Namespace
- Additional Schema Features
- Instance Attributes

XML Messaging Overview

- eMerge Messaging Highlights
- Bi-directional Messaging
- Messaging Data Model
- Component
- Service
- Schema
- Schema/Component Mapping
- Document
- Testing
- Party and Channels
- Development Activities
- Message Rules

- Component & Post Rules
- Modular Applications
- Customization Options
- Web Services Support

Importing a Schema

- Introduction
- Import Schema - 4 Steps

Mapping a Schema to a Component

- Schema/Component Mapping
- Mapping Definition
- Root Selection
- XML Mapper Display
- XML Mapper Toolbar
- Mapping Candidates - Target
- Mapping Candidates - Source
- Filtering
- Mapping Process
- Inbound Mapping
- Outbound Mapping
- Reusing a Mapping

Defining a Document

- Define a Document
- Associate a Document with a BC
- Select Schema Mapping for the Document
- Compiling a Document

Defining a Party

- eMerge Party
- Define a Channel

XML Tester

- Introduction
- Steps

Rule Syntax

- Firing a Message
- Component Rule
- Post Rule

Messaging Application Logic

- Inbound Messages
- Outbound Messages
- Data Types
- Responding to an Inbound Message
- Notification System
- eMergeError Document
- eMergeNotification Document
- Synchronous Messages
- BLP Processing
- Message Descriptors
- Trace

Modular Applications

- Modular Applications
- Modules & Parties
- eMergeModule
- BLP Architecture

- Sending Message to an eMergeModule
- Queued Message Sending
- Invoking Component Methods from Rules
- BLP Stack

Methodology

- Development Scenarios
- Component Definition

Web Services

- What is a Web Service?
- Web Services Model
- Web Services Standards
- SOAP Message
- eMerge Web Services Support
- Web Services – Provider Architecture
- Web Services – Requestor Architecture
- SOAP Messages in eMerge

eMerge Queries

| | |
|----------------------|--|
| Duration | 2 days |
| Description | Presents a hands-on detailed description of the advanced capabilities of eMerge query and data manipulation language. Enables the participant to fully utilize queries in application development. |
| Audience | IT professionals involved in the development and maintenance of eMerge applications, who have some eMerge experience |
| Prerequisites | Comprehensive Developer |

Contents

Introduction

- Query basic principles
- Creating and executing a query
- Query structure
- DEMOQUERY
- Options

Query Selection

- Unconditional selection
- Conditional selection
- Wild card selection
- Parametric queries

Query Report

- PRINT statement
- Report annotation
- Report layout options
- Sorting

More Features

- Joins
- USING
- LAST, FIRST, ACTUAL
- Virtual fields and classes
- Computing field values
- CASE

Query Form

- Output forms
- EDIT statement

Updating the Database

- UPDATE statement

Generating External Files

- EXTRACT statement

Query Internals

- Compilation
- Execution
- Report
- Selection with range
- Using query internals for problem determination

Designer's Tips

- Selection within ranges
- Comparing two constructors
- Query with * option
- Dictionary queries
- Query security
- Query tuning

Queries for SQL

- Benefits
- Concept
- DEFINE SQL statement
- Column matching
- Implicit/explicit matching
- Matching algorithm
- User parameter
- Context parameters
- Execution modes
- Dynamic or static

Legacy Adapter

| | |
|----------------------|---|
| Duration | 4 days |
| Description | Explores the capabilities of <i>Legacy Adapter</i> . Enables the participant to fully utilize <i>Legacy Adapter</i> for nonintrusive renewal of legacy applications by mapping legacy application screens and capturing legacy application flows. |
| Audience | IT professionals involved in the development and maintenance of <i>eMerge</i> applications, who have some <i>eMerge</i> experience. |
| Prerequisites | Comprehensive Developer |

Contents

Overview

- Legacy opportunities
- Application integration
- How to reach Legacy Logic
- Data mapping
- Challenges
- Requirements
- Solution model
- *Development Workbench* environment
- Legacy Screen Editor
- Screen identification
- Model
- Flows
- Runtime
- Screen scrapers vs. *Legacy Adapter*

Concepts

- Solution model
- Basic concepts
- Workplan
- Studying the Legacy Application

Legacy Application

- Retrieving the policy
- Opening the legacy application
- Legacy data incorporated into *eMerge*

The Workbench

- Define the session, the cluster, the composite
- Init flow
- Define the function
- Application flow
- Identify the screen
- Validate the screen
- Error
- Define the class
- Define an instance

- Define a field
- Map and capture the screen
- Use existing classes
- Design and implement
- After implementation
- Re-use an existing screen
- Additional techniques
- Define a class with a parent
- Copy the instance
- Mapping
- Mapping an existing field
- Design using virtual fields
- Implementation and after implementation

Services

Hints & Tips

- Workbench tips
- RestartSessionMode option
- Virtual field
- Available functions
- Legacy Loader
- Flow status
- Update the multi-instance class
- Debugging the design
- Trace
- Live mode
- Open an existing screen

Methodology

- Planning the application for *Legacy Adapter*
- Planning function flows

Internals

- *eMerge*, general and *Legacy Adapter* architectures
- Run-time and *Legacy Adapter* components
- Legacy Screen Editor
- Defining Cluster Screen flow
- Identification/validation

- Mapping/capture
- Memory run-time knowledgebase
- Flat and context
- Flow management
- Script
- Communication: Conversation, Rumba and FEPI

- TN3270 architecture

Installation Notes

- *Legacy Adapter* and Telnet
- Requirements
- Installation steps
- Customization for FEPI

DBMS Adapter: SQL

| | |
|----------------------|--|
| Duration | 3 days |
| Description | Presents a hands-on workshop that describes the implementation of <i>eMerge</i> applications in an SQL environment. Includes interface techniques, tuning and application design considerations. |
| Audience | SQL Administrators (essential), project leaders and senior application developers who want to gain a better understanding of how to work with SQL. |
| Prerequisites | Comprehensive Developer is recommended |

Contents

Introduction to DBMS Adapter

- What is *DBMS Adapter*?
- Database and datasource access
- How the DBMS interface works
- Building the connection
- The mapping process
- Datasource definition
- Database definition

DBMS Profile

- DBMS profile
- Define the DBMS
- Define the SQL log table
- Define the DBMS profile
- Establish a default logical DBMS ID

DB2 & SQL Overview

- Relational model
- DB2 organization
- DB2 SQL
- Referential integrity
- Table manipulation
- Functions
- Indexes and views
- Bind process

DBMS Adapter to DB2

- Introduction
- Mapping to existing DB2 tables
- SQL expression
- SQL token
- Physical mapping compilation

Dependent Concepts and Indexes

- Associations and relationships
- Hierarchical structure
- Concept information
- Physical record definition

- Indexes

The I/O Module Process

- Generating the DB2 I/O module
- I/O module internal structure
- System I/O module

Creating DB2 Table

- Creating DB2 tables
- Generating DDL
- DB2 authorization requirements
- Accessing the DB2 catalog
- Importing existing DB2 tables

eMerge Applications with DB2

- Design consideration
- Performance enhancement
- Using *eMerge* SQL trace
- DB2 data types

Queries for SQL

- Benefits
- Concept
- DEFINE SQL statement
- Column matching
- Implicit/explicit matching
- Matching algorithm
- User parameters
- Context parameters
- Execution modes
- Dynamic or static

FetchSQL

- FetchSQL rule and its syntax
- Retrieve SQL data
- Tracing a fetchSQL rule

DBMS Adapter with Oracle

- Environment specifications
- Enhancements
- Limitations

DBMS Adapter with DB2/400

- Environment specifications
- Enhancements
- Limitations

Business Rule Language (BRL) Workshop

| | |
|----------------------|---|
| Duration | 5 days |
| Description | Presents a hands-on detailed description of the advanced capabilities of the <i>eMerge</i> Business Rule Language. Enables the participant to fully utilize BRL in application development. Provides ample opportunity to learn through hands-on exercises. |
| Audience | IT professionals involved in the development and maintenance of <i>eMerge</i> applications, who have some <i>eMerge</i> development experience. |
| Prerequisites | Comprehensive Developer; Actual hands-on experience. |

Contents

Review

- Types of rules
- Types of triggers
- Positive thinking

General Topics

- Global Fields
- BRL context
- Fromoper
- Oldvalue
- NoAutoTrigger
- Online/OnBatch
- DownTrigger
- OnlyActual
- Rules bypassed for efficiency
- Structure of composites 7 and 467
- Masking
- Trigger options

Triggers

- Class triggers
- Block in form triggers
- Issue operation triggers
- Edit operation triggers
- Query/Program triggers
- Error triggers

Debugging

- Trace rules
- Trace Execution
- Trace definition
- Trace fetches
- Trace derivations
- Trace fields
- Trace rulesets
- Trace pure
- Saving traces

- Count statistic

Performance Considerations

- Order of execution of rules
- Group mode
- Trigger fields
- Validation rules on delete
- Redundant fetches

Computation Rules

- Rounding
- IS TRIGGER
- While loop
- Returned Action
- Order of operations
- Functions

Date Fields

- How dates are physically interpreted
- Date fields definitions
- High value
- Masking dates
- Date functions
- Date manipulations

Call Rules

- Dynamic calls
- CreateThen/CreateElse
- Call rule options
- Calling programs

Fetch Rules

- Fetch Next/Fetch Previous
- Fetch Actual
- USE/THEN/ELSE
- Fetch Repeated
- Termination of loops
- Computations within fetch rules
- SameComposite/SameBranch
- IgnoreMissingTarget

- IgnoreMissingPath
- VERIFIED
- Dynamic error messages
- FetchSQL

Derivation Rules

- NoAutoTrigger
- Lastchange
- Violating valid values
- Source check
- Recursion
- Opposite
- InsertEqualChange
- Secondary
- MayChangeNothing
- Triggering Options
- ADDED/SUBTRACTED statements

- INVERSED statement

Queue

- Controlling physical order of derivations

Rule Analyzer

- Impact analysis
- Conflict analysis
- Sequencing analysis

Pack and Post Rules

- Syntax
- Implementation

Documentation

- Listrules
- Ruleset documentation
- Rule documentation

Business Integrity Server Administration

| | |
|---------------|--|
| Platform | z/OS, i5/OS, Windows |
| Duration | 3 days |
| Description | Explores the installation, maintenance and tuning of an <i>eMerge</i> site. Tailored to match the installation environment of the participants. |
| Audience | Administrators, system programmers and <i>eMerge</i> project leaders who need to know about the installation, maintenance and tuning of an <i>eMerge</i> site. |
| Prerequisites | Comprehensive Developer is recommended |

Contents

Introduction

Introduction

- Administrator Roles & Responsibilities
- *Business Integrity Server* Administration

Accessing eMerge

Accessing an eMerge Session via Development Workbench

- Starting an *eMerge* Session
- Ending an *eMerge* Session

Accessing an eMerge Session via Terminal Mode

- Starting an *eMerge* Session in Terminal Mode
- Ending an *eMerge* Session in Terminal Mode

Updating an eMerge Database via Batch

- Updating an *eMerge* Database in Batch
- Running *eMerge* Batch Jobs

Accessing Business Integrity Server via C Programs

- Accessing via C Programs
- API Services Provided

Database Administration

What is an eMerge Database

- What is an *eMerge* Database
- Multidatasource Database Advantages
- Standard Database
- Model Database
- Standard Logical Files
- Listing File Ranges and Key Ranges
- The CT Database—the *Catalog*

Defining a Database

- *eMerge* Database
- Database Definition Procedure
- Define a Journal
- Set Up the Lexicon
- Map a Database to an Local Database on *eMerge Client*
- Virtual *eMerge* Datasources

Interfacing with External Datasources

- *eMerge* Adapters
- *DBMS Adapter*
- *eMerge* Custom Adapter
- *Legacy Adapter*

Application Administration

Defining Application Users

- Introduction
- Defining a User
- Letting the User Maintain the Password

Configuring and Tuning Business Integrity Server for Your Site

- Configuring and Tuning *Business Integrity Server* for Your Site
- Query Tuning
- Operation Tuning
- Definitions to be Made in the CT

Task Memory

- Task Memory Tuning
- Recommended Values for the Task Memory Key Length
- Support for Task Memory

Database Memory

- Memory Database Index
- Balancing and Refreshing
- Insufficient Memory
- Multiple Database Memories and Indexes

- Estimating the Size of Memory Database
- Database Memory Tuning
- Multiple Memory Databases

Defining Environment Defaults

- General Database Options
- Customizing Special Characters
- Customizing Operation Code Values
- Changing the Current Date and Time
- Changing the Standard Color Palette

Managing Object Number Allocation

- Rangesets, Complexity and the Rules
- Default System Rangeset
- Defining Private Rangesets
- Defining User Privacy via Private Rangesets
- Viewing Allocated and Used Number Ranges

Environment Administration

Handling Transaction Allocation

- *Listener*

Business Integrity Server Administration File

- What is the Administration File
- Administration File Structure
- Building the Administration File

Optimizing Business Integrity Server Online Processes

- Diagnostic Tools
- COUNT Command
- Programs that Cause Excessive I/O
- Possible Causes of Excessive Use of CPU by the BLP
- Possible Causes of Excessive I/O by the BLP

Maintaining a Database

Business Integrity Server Utilities

- Description of the Utilities Available for the Administrator

Porting Databases

Porting Databases

- Introduction
- Porting Process
- EBCDIC vs. ASCII Character Sets
- Exporting the Database to Sequential Files
- Language Translation Issues
- Parameter File
- Specifying which Datasources to Export
- Specifying which Composites to Export
- Sequential Files Resulting from Export
- Transferring Files
- Importing from Sequential Files

Troubleshooting

Diagnosing Problems

- Introduction
- Commands
- Form-based Version of SNAP TRACE
- Diagnosing Abends
- DUMP Command and DUMP Runtime Parameters
- Formatted Batch Output
- Structures

SNAP

- Using SNAP to Display Internal Memory
- Using SNAP Trace

Change Management

| | |
|----------------------|--|
| Duration | 2 days |
| Description | A description of how to enable Change Management for an <i>eMerge</i> database, so that all changes to the application are associated with a particular task. How to determine what tasks should be migrated to a test or production database, and how to migrate the tasks. |
| Audience | Administrators who manage the migration of application development tasks to the production environment. |
| Prerequisites | Comprehensive Developer with an understanding of <i>eMerge</i> security. |

Contents

Introduction to eMerge Change Management

- Environments
- Terminology: Migration
- Change Management Migration Stages
- Features Overview
- Selective Recording in Internal Journal
- Concepts: task type, Migration Unit (MU), task, task status, tasks and MUs, memos, task users/worlds, objects, Internal Journal and its index
- Extended Task Properties: excluded and common tasks, task tracking
- Change Management Activities

Enabling Change Management

- Change Management Setup
- CM Coordinator Trees
- CM datasource

Start-Up Activities: Creating the Definitions

- CM Definition Activities: Operational and Informational Task Types, define the Task Types, building the Task Type Hierarchy, define Customized Status Designations, and define the Task: privacy restrictions, task description, hierarchy, assign users to tasks, and using ESI level 3 with CM

Reports

- Task Reports: Floating Tasks for Type, Open Tasks, Tasks by TimeFrame, Task List by UserID, Task List by Customized Status, List of Open Tasks for Task, Hierarchy for Task and All Hierarchies for Task
- Direct Objects for Task
- Shared Objects for Task

- Indirect Objects for Task

Maintenance Activities

- The Maintenance Activities Menu
- Tasks Generated by the System
- Index Integrity
- View Internal Journal, Index and Operations Associated with Object
- Loading Internal Journal from External Journal

Ongoing Activities

- The Ongoing Activities Menu

Impact Analysis

- SAPCMUTL Utility
- Detailed Task Connection Report
- Summary Task Connection Report
- Invoke the SAPCMUTL Utility

Pre-Migration Activities

- Pre-Migration Activities Menu
- Open Memos for Task

Migration Activities

- The Migration Activities Menu
- Final Steps
- Extract Operations Associated with a Task
- User IDs
- Using SPTASKDB Output Files
- Application Test Data Migration
- Clear the Internal Journal and Index Files

Local Database Synchronization

- *Development Workbench* Synchronization

Developer Issues

- Change Management Overview for the Developer
- Choosing a Task
- Collision Detection
- Error Messages from Integrity Checks

Managing i.way Components

- Development
- Development to Test
- Testing
- Deployment

- Diagram of Change Management for Components

Utilities

- Utilities: SPLDJ2DB, SPJRNDDB, SPTASKDB and SPNBKUP

Security

| | |
|----------------------|--|
| Duration | 1 day |
| Description | Covers the security and privacy capabilities of eMerge. Topics include eMerge internal security and the External Security Interface. |
| Audience | IT personnel responsible for maintaining security in development and production. |
| Prerequisites | Comprehensive Developer. |

Contents

Internal Security

- Introduction
- SAPIENS Internal Security - Concept
- Classification & Authorization
- User Definition & Privacy Profile
- Security Activation and Management
- Initial Setup

Applying Security

- Activating Security
- Unclassified Composites
- World - Definitions
- Defining Worlds
- Defining a User
- Access Default and Priorities
- User Password
- Initial Commands
- Allowed Databases & Terminals
- Query Worlds
- Security Separation

Privacy

- Privacy - Definitions
- Privacy for a User/World
- Privacy Conditions
- Hierarchical Dependency and Privacy
- Privacy in Worlds
- Users Privacy
- Privacy Conditions Check
- Security Messages

External Security Interface

- Introduction
- User Profile
- ESI Concept
- ESI Main Menu
- ESI Setup Parameters
- World Conversion
- User ID Conversion

- ESI Level 3 Activation

Security Management

- Introduction
- Supplied Users
- Supplied System Worlds

Task Administration

- Tasks in Separation Mode
- Tasks in Centralization Mode
- Changing from one mode to another

Activating Application Security

- Activating Security in Centralization Mode
- Deactivating Security in Centralization Mode
- Activating Security in Separation Mode - \$GSM or &PHYDBA
- Deactivating Security

Accessing Business Integrity Server from External Security Software

- Introduction
- ESI
- Cross-Referencing Information
- Set up ESI Worlds (Multidatabase)
- Set up ESI LikeUsers (Multidatabase)
- Modify the ESI Exit Routine
- Considerations for RACF-z/OS under IMS/DC
- Activating ESI for TCP/IP if Necessary
- Enabling ESO for TCP/IP if Necessary
- Enforcing Use of ESI

Web Client Security Considerations

- Introduction
- Web-Client Authentication
- Web Authentication Required
- To Simulate Single Signon
- Change Passwords
- Kiosk-Type Users
- Form-Based Privacy Level