IBM WebSphere Transformation Extender

Pack for SEPA

Version 1.0

Note
Before using this information, be sure to read the general information in "Notices" on page 33.

May 2007

This edition of this document applies to IBM WebSphere Transformation Extender Pack for SEPA Version 1.0; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email dtxdocs@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction

This section provides a brief introduction to the IBM WebSphere Transformation Extender (WTX) product family and introduces the IBM WTX Pack for SEPA.

The WebSphere Transformation Extender product family

The IBM WebSphere Transformation Extender (WTX) product family provides a universal transformation engine that can describe and transform any data, work directly in any architecture, and reuse the transformations across the enterprise. WTX supports specific industry segments, such as financial services, with Industry Packs, which together provide the capabilities to perform the following:

- transform, validate, and enrich any document, message or complex data
- deliver trustworthy information for critical business initiatives
- meet regulatory compliance requirements
- support codeless development; universal reuse and deployment.

Introduction to the IBM WTX Pack for SEPA

The IBM WTX Pack for SEPA supports validation of the Single European Payments Area (SEPA) scheme datasets, transformation samples between SEPA, and individual domestic formats. The components allow you to link your current payments systems and operations in their current mode to SEPA validation, thus providing processing and enrichment for your back office and intermediary systems.

This documentation provides introductory and reference material needed to implement the WTX Pack for SEPA as part of an overall SEPA solution. This documentation contains:

- introductory background on SEPA, including
- a description of how the WTX Pack for SEPA fits into your SEPA processing for credit transfer and direct debit transactions
- configuration of WTX Pack for SEPA components, including specifics of deploying the components on z/OS
- a description of the SEPA message validation framework
- a discussion of the domestic format converters.

For a description of installation and uninstallation procedures for this product, refer to **readme_sepa.txt** which installs with the product and may be found at the following location:

*install_dir***packs****sepa_v***n.n**

Note: In the location above, as well as throughout the documentation for this product, *install_dir* indicates the location of the core IBM WTX product, and *n.n* indicates the current product version.

What the WTX Pack for SEPA contains

The WTX Pack for SEPA contains the following principal components:

- ISO 200222 (UNIFI) message schemas for credit transfer and direct debit (**.xsd** files) and the WTX type trees for credit transfer and direct debit formats. The type trees are generated using the WTX Schema Importer.
- Electronics Payments Council (EPC) Technical Validation Subset (TVS) schemas for the relevant payments clearing and settlement messages, and the WTX type trees generated for these with the WTX Schema Importer.
- Additional WTX validation maps used to validate the general usage rules as defined in the EPC rule books.
- Example converter maps for the following domestic credit transfer and direct debit messages:
  - United Kingdom - BACS, see "Domestic format for the United Kingdom" on page 29.
  - Germany - DTAUS, see "Domestic format for Germany" on page 30.
  - France - MINOS and CFONB, see "Domestic format for France" on page 30.
- WTX type trees for the SWIFNet FIN MT 103 message, as well as the WTX transformation maps used to convert between SEPA formats and the SWIFTNet FIN MT 103 message.
- z/OS example with sample JCL for a SEPA passthrough map. See"Running the Pack for SEPA on z/OS" on page 14.
- Reference data templates (**bic.xml**, and **currencycodedecimals.xml** files).

Chapter 2. The SEPA initiative

Each European country has developed its own banking and payment system in accordance with local market requirements. The result has been a varying level of payments service for customers. This is particularly noticeable to citizens who cross borders regularly to shop and holiday within Europe. In today's financial marketplace, customers are demanding improved services and are increasingly seeking a prompt payment processing system that is consistent both within their home country, and in other European markets.

To address this problem, the EPC, in 2002, proposed a single payment process. At that time, 42 banks, and three European Credit Sector Associations (ECSAs), along with the Euro Banking Association (EBA) came together to release a proposal formally launching the SEPA concept.

The definition of SEPA was approved by the EPC in December of 2004 and states that:

"SEPA will be the area where citizens, companies and other economic actors will be able to make and receive payments in euro, within Europe (currently defined as consisting of the 25 European Union (EU) member states plus Iceland, Norway, Liechtenstein and Switzerland), whether between or within national boundaries under the same basic conditions, rights and obligations, regardless of their location."

SEPA message standards

This section describes SEPA message standards and provides an overview of the UNIFI ISO 20022 (UNIFI) standard upon which the SEPA standard is based.

Note: The terms ″UNIFI″ and ″ISO 20022″ are used somewhat interchangeably throughout this and other SEPA documentation, with both referring to the ISO 20022 **UNI**versal **F**inancial **I**ndustry message scheme.

This platform was proposed by the International Standards Organization (ISO) to be used to develop all financial industry messages. It must be understood that UNIFI by itself does not describe messages. Rather, it should be thought of as a recipe to be used to develop financial industry message standards. The main ingredients of this recipe are a development methodology, a registration process, and a central repository.

About the UNIFI message standard

At the beginning of the decade, the general interest in Internet Protocol (IP) and XML was perceived as a unique opportunity for the industry to move to a common XML-based language. There was a problem, however, due to the fact that XML is not a language at all, but rather a meta-language which everyone can use to define their own dialect. As a result, several standardization initiatives grew from the common XML concept, each with messages using their own dialect, or adaptation of XML. This led to not only an increased risk of yet more 'languages' being introduced, but a waste of resources in the form of duplicated effort, and a further risk of divergence when more than one initiative is brought to focus on the same business area.

In an attempt to solve this problem, ISO proposed a single standardization approach, or ″recipe″ which includes a common development methodology, a common process, and a common repository, that would be used by all financial standards initiatives. This recipe, called UNIFI or ISO 20022 is the message standard around which the WTX Pack for SEPA is developed.

For a complete description of ISO 20022 message flow, refer to the International Standardization Organization, ISO 20022 UNIversal Financial Industry message scheme. This information is available at:

www.iso20022.org/

EPC rulebook validation of SEPA messages

The EPC publishes various documents related to SEPA, including a data model reference, rulebooks, and implementation guides for both credit transfers and direct debits. The SEPA data model describes how three layers are recognized:

- a business process layer
- a logical data layer to define the datsets and their attributes, and
- the physical layer which specifies the actual document formats and messages.

The rulebooks contain the business requirements and rules for the operation of the SEPA schemes.

The implementation guides contain detailed specifications on the physical layer of the data model, and thus define SEPA specific validation that is applied to ISO 20022 (UNIFI) messages to ensure compliance with the SEPA.

The WTX Pack for SEPA includes maps to apply these validation rules to the various ISO 20022 SEPA messages so that customers can validate their messages to ensure compliance. These messages are listed in the “Credit transfer SEPA messages” on page 7 and in the“Debit transfer SEPA messages” on page 10.

The maps provide a check-digit validation of the following fields:

- IBAN
- Identification of the Creditor (AT-02) - direct debit only

They also perform validation checks against XML lookup files to verify the following items:

- Bank Identification Code (BIC)
- Currency codes
- Country codes

CORE and CORE plus Additional Optional Services (AOS) schemas

The EPC publishes a set of Technical Validation Subsets (TVS) that support the rulebooks and implementation guidelines. These TVSs were developed by the Society for Worldwide Interbank Financial Transactions (SWIFT) at the request of the EPC and contain ten schemas in all. Five are called CORE schemas, and five are called CORE plus Additional Optional Services (AOS).

Core schemas

The CORE TVSs are based on the ISO 20022 (UNIFI) schemas, but have been modified to apply some of the usage rules defined in the EPC rulebooks and implementation guides.

It should be noted that not all of the usage rules are included in the CORE schema. The validation maps provided in the WTX Pack for SEPA ensure that all rules are followed in a particular instance document. The five CORE schemas are:

- FIToFICustomerCreditTransfer.EPCCoreV02
- FIToFICustomerDirectDebit.EPCCoreV02
- PaymentStatusReport.EPCCoreV02
- PaymentReturn.EPCCoreV02
- FIToFIPaymentReversal.EPCCoreV02

Additional Optional Services (AOS)

Some participants provide complementary services based upon the SEPA scheme in order to meet specific customer expectations. Such participants are called AOS participants, of which there are two types. One type of AOS can be described as those that are offered by individual participants, and the second type of AOS are those offered by communities of participants.

The AOS message elements are included in the CORE plus AOS schemas, per the implementation guidelines. AOS communities can define and document rules for certain elements within the SEPA message. If a message is sent to a member of the AOS community, these rules must be followed. Also, senders must ensure that elements defined an AOS are not included in a message sent to a non-member. If the receiving bank is a non-member, they can ignore the AOS fields, but must preserve them if the payment message is forwarded.

The remaining five schemas are for CORE plus AOS are the following:

- FIToFICustomerCreditTransfer.EPCCoreAOSV02
- FIToFICustomerDirectDebit.EPCCoreAOSv02
- PaymentStatusReport.EPCCoreAOSV02
- PaymentStatusReport.EPCCoreAOSV02
- FIToFIPaymentReversal.EPCCoreAOSV02

Message flow overview

The core messages defined for use by the EPC comprise a subset of the ISO 20022 Payments Clearing and Settlement (**pacs.nnn.nnn.nn**) and Payments Initiation (**pain.nnn.nnn.nn**) business areas. The **pacs** messages are for use in the bank-to-bank processing space, and have been designated mandatory by the EPC and they must be used by all SEPA participants. The **pain** messages are recommended for use in the customer-to-bank space.

It should be noted, that within SEPA, there are numerous terms that can be used interchangeably. In addition to the terms UNIFI and ISO 20022 XML which are nearly synonomous, SEPA messages can be referred to either by its UNIFI message number, or dataset identified in the EPC rulebook. For example the following mean the same:

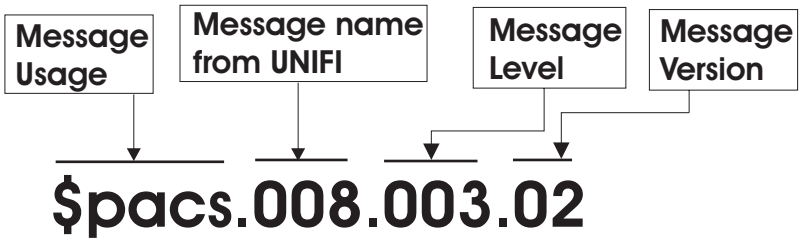
- FI to FI Customer Credit Transfer

- pacs.008.001.01
- Inter-bank Payment Dataset – DS-02

See the "Credit transfer message flow" on page 7 and "Direct debit message flow" on page 9 for diagrams that show the major processing steps for credit transfer and direct debit message flows.

Validation framework map terms

When referring to the SEPA distributed schemas, the following terms are used throughout the documentation, as well as the validation framework map set. The example below assumes a typical schema name of **$pacs.008.003.02**:



Message usage

Message usage may be one of two types:

- **pacs** - (as shown in the example) - indicates that the message is for bank-to-bank use
- **pain** - indicates that the message is for customer-to-bank use

UNIFI message name

This portion of the schema name indicates the message name that has been assigned by UNIFI. In the case of the example, the message name is 008.

Message level

This portion of the schema name indicates the message level. Possible values are:

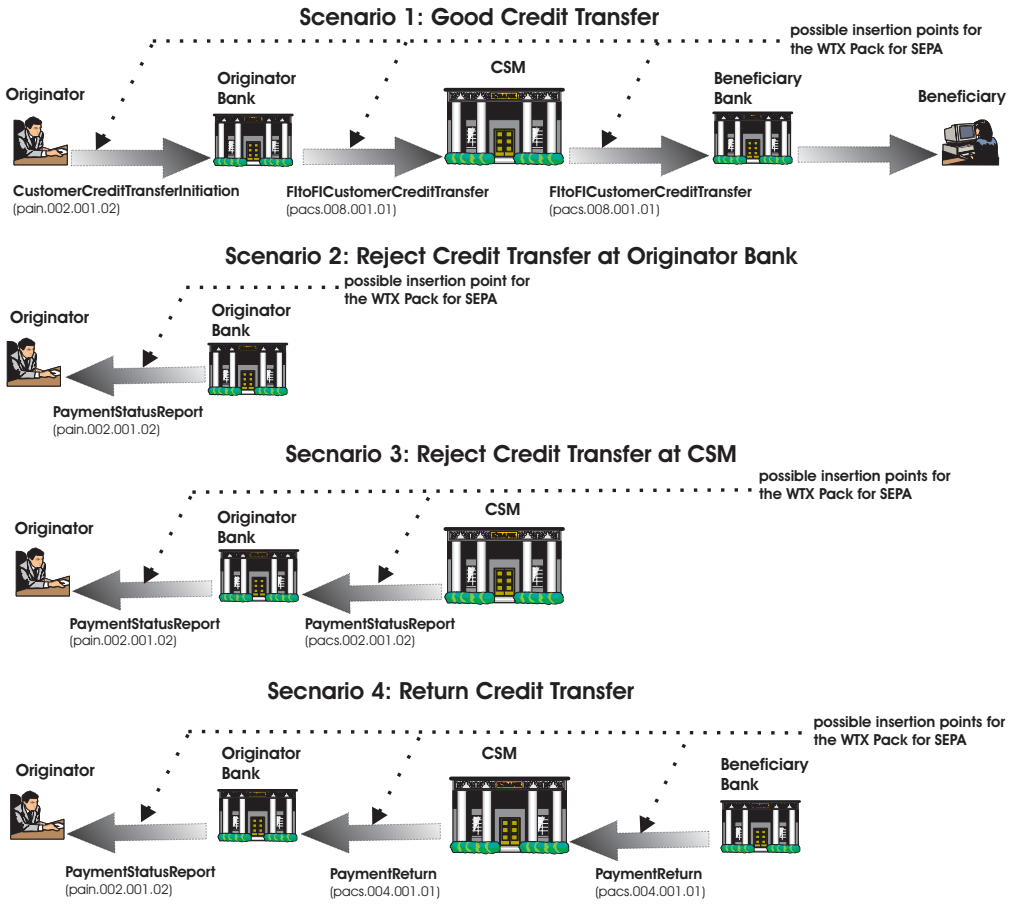
- 001 = UNIFI level
- 002 = SEPA core level
- 003 (as shown in the example) = AOS level

Message version

This portion of the schema name indicates the message version. In the case of the example, the message version is 02.

Credit transfer message flow

The following diagram shows four credit transfer scenarios.



Credit transfer SEPA messages

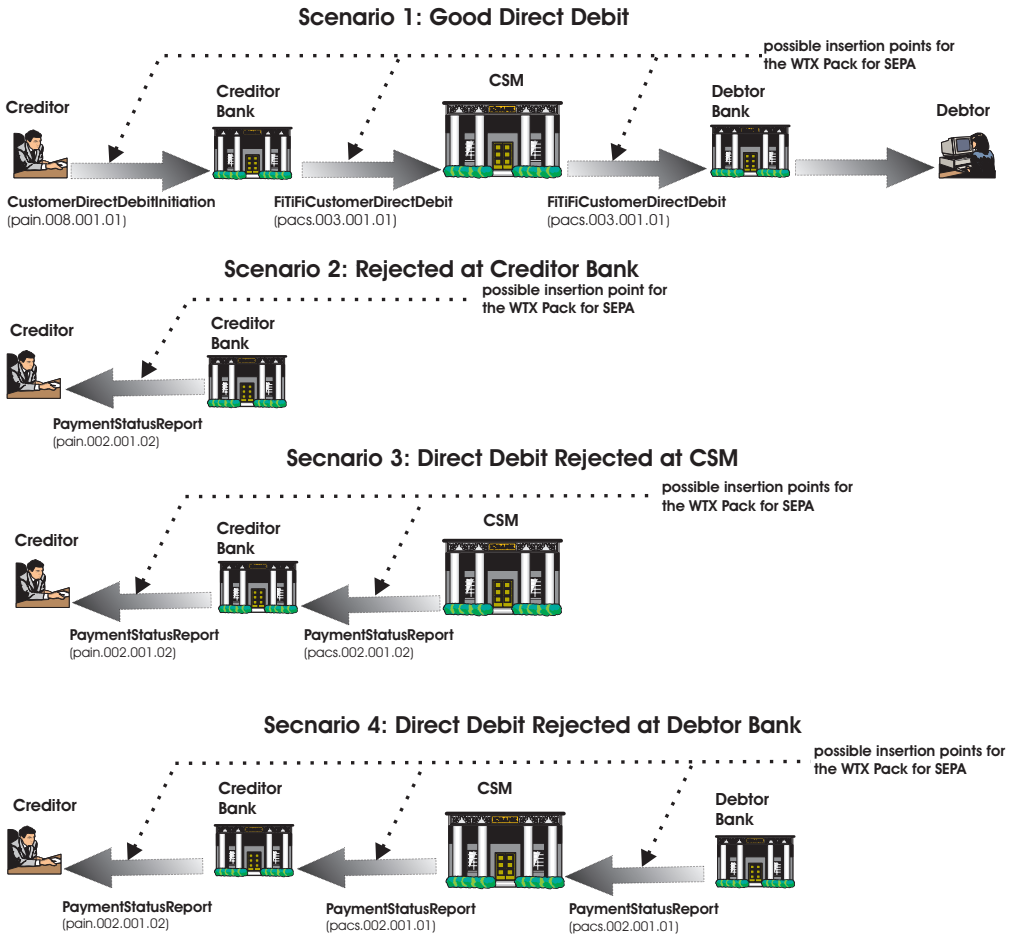
The following are the credit transfer SEPA messages.

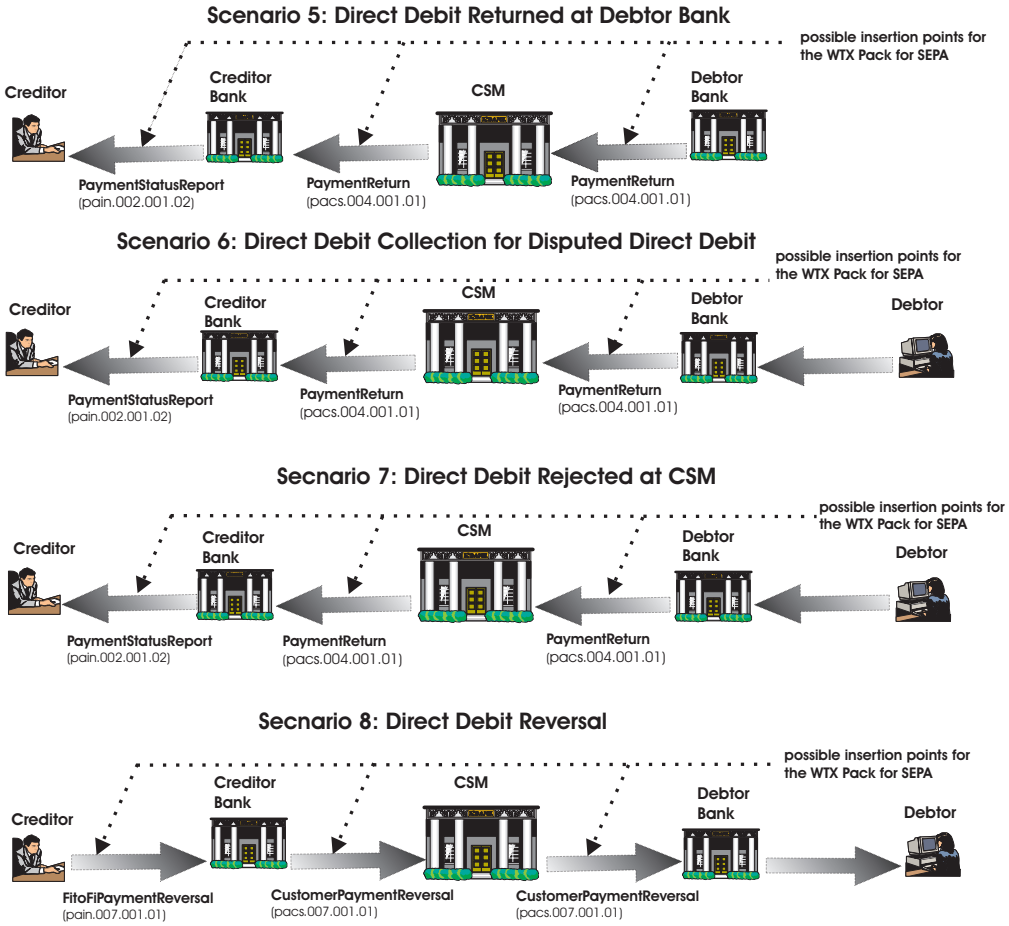
Message Id	Description	Usage	Purpose
pain.001.001.02	CustomerCreditTransfer Initiation	Customer - Bank	Transports customer to bank CT information sent by the originator to the originator bank. Caters for single and bulk instructions.
pacs.008.001.01	FItoFICustomerCredit Transfer	Bank - Bank	Transports payment instruction from originator bank to beneficiary bank directly or through intermediaries. Caters for single and bulk instructions.

Message Id	Description	Usage	Purpose
pacs.002.001.02	PaymentStatusReport	Bank - Bank	Transports the CT reject instruction between banks directly or through intermediaries. Caters for single and bulk instructions.
pacs.004.001.01	PaymentReturn	Bank - Bank	Transports the CT return instructions between banks, directly or through intermediaries. caters for single and bulk instructions.
pain.002.001.02	PaymentStatusReport	Customer - Bank	Transports CT return instructions between banks and remitting customers. Caters for single and bulk instructions. Note: The EPC IG does not explicitly specify how to transport CT return instructions to the originator, even though the process flow diagram in the EPC Rulebook indicates that it may be required. AT-R3 does however differentiate bewteens Returns and Rejects, so this seems the logical way to do it.

Direct debit message flow

The following diagrams show debit message flow.





Debit transfer SEPA messages

The following table defines the debit transfer SEPA messages. It should be noted, that pacs.004.001.01, pacs.002.001.02 and pain.002.001.02 are used in both Credit Transfer and Direct Debit scenarios. The contents of certain fields within the messages indicate which scenario is being used, as shown in the table.

Message Id	Description	Usage	Purpose
pain.008.001.01	CustomerDirectDebit Initiation	Customer - Bank	Transports the direct debit collection instruction from the Creditor to the Creditor bank. Caters for single and bulk instructions.
pacs.003.001.01	FItoFICustomerDirect Debit	Bank - Bank	Transports the direct debit collection instruction from the Creditor to the Debtor bank, directly or thru intermediaries. Caters for single and bulk instructions.

Message Id	Description	Usage	Purpose
pacs.002.001.02	PaymentStatusReport (original message name id=pacs.003.001.01) and transaction reason = RJCT)	Bank - Bank	Transports the direct debit reject instruction between banks, directly or thru intermediaries. Caters for single and bulk instructions.
pain.002.001.02	PaymentStatusReport (original message name id = pacs.008.001.01) and transaction reason = RJCT) and status reason code contains AT-R3 reject reason code)	Customer - Bank	Transports CT reject instruction between banks and remitting customers. Caters for single and bulk instructions.
pacs.007.001.02	FiToFiPaymentReversal (Original message name id = pacs.003.001.01)	Bank - Bank	Transports the interbank reversal instruction for a collection from the creditor bank to the debtor bank, directly or thru intermediaries. Caters for single and bulk instructions.
pain.007.001.01	CustomerPaymentReversal Original message name id = pain.008.001.01)	Customer - Bank	Transports the customer to bank reversal instruction for a collection from the creditor to the creditor bank. Caters for single and bulk instructions.
pacs.004.001.01	PaymentReturn Original message name id = pacs.003.001.01	Bank - Bank	Transports the direct debit return / refund instruction from debtor bank to creditor bank, directly or thru intermediaries. Caters for single and bulk instructions.
pain.002.001.02	PaymentStatusReport Original message name id = pain.008.001.01 and transaction status reason = RJCT and status reason code contains AT-R3 return reason code	Customer - Bank	Transports direct debit return instruction between banks and remitting customers. Caters for single and bulk instructions.

Chapter 3. Configuration

The documentation provided here describes SEPA system configuration. This includes a detailed description of the major components that comprise the WTX Pack for SEPA, followed by detailed instructions on how to run the WTX Pack for SEPA on z/OS. The z/OS section includes sample batch scripts and JCL.

Major components of the WTX Pack for SEPA

When you install the WTX Pack for SEPA, three subdirectories are created under *install_dir***packs****sepa_v***n.n.* These subdirectories are:**mapsandschemas**; **type_trees**; and **examples**.

In addition, a readme file describing installation and uninstallation procedures is provided under *install_dir***packs****sepa_v**. Release Notes for the WTX Pack for SEPA are located at: www.ibm.com/software/integration/wtx. The Release Notes contain the most current information about the release, including additional requirements, configuration instructions, and known issues

mapsandschemas

This subdirectory contains validation framework maps, schemas, and reference data files and maps. The components of the **mapsandschemas** subdirectory are described in detail below:

- **Validation framework maps.** – These maps perform validation of the SEPA message by checking for: 1) Basic UNIFI compliance and 2) EPC-mandated usage rules for Credit Transfer and Direct Debit. The main map for the validation framework (**sepvalid**) resides under the **sepa_validation.mms**. All of the validation maps reside within this source file, so they can all be built at the same time, by using the **Build All** option in the WTX Map Designer.
- **Schemas.** – All needed schemas are provided for the validation of all the different message types and levels (UNIFI, SEPA, CORE and AOS). Any new or modified schema must be located here. Also, there is a SEPA generic schema (**sepagen.xsd**) that contains a list of all of the messages that are supported by the validation framework maps.
- **Reference data files and maps.** – The WTX Pack for SEPA comes with XML templates used by the validation framework for validating BICs, currency code, country code and allowed decimal places for currency amounts information. The WTX Pack for SEPA also contains maps to populate these templates with live data supplied by SWIFT (**sepa_library.mms**).

type_trees

This directory contains type trees for the validation framework.All schemas required for the validation framework have a corresponding tree in this subdirectory. These schema type trees are created using the WTX type tree schema importer. A tree for the SEPA generic schema (**sepagen.mtt**) is also provided.

These type trees were imported using both the Xerces and the Classic importer.

examples

The examples subdirectory is located at:

*install_dir***packs****sepa_v***n.n****examples**.

This subdirectory contains examples for converting between SEPA formats and various national formats, as well as other WTX Pack for SEPA components. The domestic formats supported by examples in the WTX Pack for SEPA are:

- Germany, DTAUS – see **\examples\germany**
- France, MINOS, CFONB – see **\examples\france**
- United Kingdom, BACS – see **\examples\uk**
-

In addition to the domestic format subdirectories, the examples subdirectory also contains the following subdirectories:

- **passthrough** subdirectory – see **\examples\passthrough**. Contains an example for validating a message against a schema in the z/OS environment along with a sample of the JCL required to execute it.
- **pain_pacs** subdirectory – see **\examples\pain_pacs** Contains example maps used to create interbank messages from customer-to-bank messages and bank-to-customer messages.
- **bundler** subdirectory – see **\examples\bundle**. Contains an example that shows how to use a simple C language module, or a WebSphere Transformation Extender (WTX) map, to split a large bundled SEPA message file.
- **sample_jcl** subdirectory – see **\examples\samples_jcl**. Contains an example that shows how to run the WTX Pack for SEPA validation maps on z/OS.

Please note the following points about the examples:

- They are supplied as examples only, and are not to be considered a full set of all possible conversions. The domestic type trees supplied with the WTX Pack for SEPA support basic syntactic validation, but do not provide for full validation.
- The SEPA type trees used in the converters are the EPC core trees. The transformation rules are best estimates. Pay particular attention to BIC and IBAN conversions, since these may require the use of additional reference data that only you can access.

Also note that each examples subdirectory contains a readme file that describes how to run the example.

Running the Pack for SEPA on z/OS

Before deploying the WTX Pack for SEPA on the z/OS platform, make note of the following factors:

- the SEPA framework consists of a main map and several executable run maps that use several schemas to validate UNIFI, SEPA and AOS rules
- all compiled maps and schemas, as well as input XML files, must be ported in binary to the z/OS platform. The schemas and maps should reside in the same location, like a library dataset.
- a DTD must be ported, in binary, to the z/OS environment, but this DTD does not need to be in the **mapsandschemas** subdirectory.

- all of the type trees in the **type_trees** subdirectory, that are related to a schema or DTD, must be (manually) modified to include the DDNAME which will point to the corresponding schema or DTD
- after modification of the type trees, the maps have to be built and then ported to z/OS
- every map, schema, and file that is ported to z/OS must have a unique 8 character DDNAME in the JCL.

Preparing work for execution on z/OS

In order to view the XML error log produced on z/OS, you can use 'ICONV', if available, or you can modify the JCL for the map run command to add '/VX15 *<name of third output card for the sepvalid map>*

You may want to create a list of the maps, files, and schemas that will reside on z/OS. In the list, indicate the original file name, the type of file, the name that the file will have on z/OS, and the DDNAME. The following table shows how a typical list could be constructed. For a more complete description of all of the maps, files and schemas being deployed on z/OS, refer to the example FTP script and the example JCL contained in the **examples\sample_jcl** directory.

Original file name	Type	File name on z/OS	DDNAME
$pacs.002.002.02.xsd	schema	sepa.maplib($PC00202)	$PC00202
xliff.dtd	dtd	xliff.dtd	XLIFF
currencycodedecimals.xml	INPUT CARD	currency.xml	CURRENCY

Chapter 4. SEPA validation overview

One of the primary components of the WTX Pack for SEPA is the validation framework map. The validation framework map is designed to simplify the calling of IBM provided validation steps and the optional integration of custom validation steps. This map orchestrates the validation flow and enables the logging to debug validation flow.

The validation framework map will take in any UNIFI 20022 formatted XML message and validate UNIFI rules as well as the set of rules published by the EPC. The validation framework is completely operated by the call of one map: **sepvalid.mmc**, which is discussed in detail in the following section.

It should be noted that no matter what the SEPA message type, the validation framework map will pass correct data. The validation framework map also has the ability to determine if the data is ISO20022 UNIFI schema data, or if the data is part of the European Council Technical Validation Set, which has two sets of schemas:

- CORE schemas, which allow only the fields that are required by SEPA and,
- CORE + AOS schemas, which allow additional elements

In order for a SEPA message to pass through the validation framework without error, the following conditions must be met:

- the XML must be well-formed
- all published applicable UNIFI rules must be followed
- all published applicable SEPA IGs and rulebooks must be enforced
- XML must validate against the published UNIFI/SEPA schemas

Optionally, the validation framework allows for validation of a message against an original SEPA message, where appropriate. The SEPA implementation guides have rules that require the original payment information to be validated in the current data document. Specifically, the contents of the Original Transaction Reference must match the original transaction information. This occurs in the **pacs004** return refund and **pacs002** reject, for both credit transfer transactions and direct debit collections. The **pacs007** direct debit reversal also has this requirement. For all of these cases, the original message is the **pacs008** FI to FI customer transfer, or the **pacs003** FI to FI customer direct debit.

sepvalid

This validation framework will take in any UNIFI 20022 formatted XML message and validate UNIFI rules and the set of rules published by the EPC. The framework is completely operated by the call of one map, **sepvalid.mmc**. No matter what SEPA message type, this map will pass the data through the validation framework. The framework has the ability to determine if the data is ISO 20022 UNIFI schema data, or if the data is part of the EPCTechnical Validation Set, which has two sets of schemas, CORE schemas, which allow only the fields that are required by SEPA, and the CORE plus AOS schemas, which allow additional elements. Custom schemas can also be added.

The main map that is called is: **sepvalid.mmc**. Three inputs are allowed:

- The first input will be the SEPA message that needs to be validated.
- The second input will be the original **pacs003** FI To FI Customer Direct Debit message, or the **pacs008** FI To FI Customer Credit Transfer. If no original data file is provided, then the matching rules in the SEPA implementation guide will not be processed.
- The third input is a true false value that is passed to turn the UNIFI rules validation on or off. If you want to validate the data for UNIFI rules, then no input is needed, as this is the default. If you wish to turn off UNIFI validation, pass in an echoin 0 (zero), as the third input.

There are 3 Output cards for the **sepvalid** map. Cards one and two do not produce output files, and card three produces the **seperror.xml** file that allows you to see all of the validation errors and framework messages produced from the data file. If debugging of the process is required, change the output for card 1 from SINK, to File and assign it a file name. This debug file will then serve as a debugging tool,, as it contains steps that document where in the process it has been, and what was performed.

SEPA validation customization

Within the SEPA validation framework maps, you can customize validation for the following:

- add additional error reporting, or add checks that are outside of the scope of the SEPA or UNIFI schemas
- add validation for a custom schema
- change individual validation rules, especially useful for those rules in the customer-to-bank messages that are only ″recommended″ by SEPA.

Unenforced usage rules

The Validation Framework does not enforce usage rules that involve uniqueness over time, because the Pack for SEPA does not provide a full end to end view of all the messages handled. Rules that imply that a field can only be used if there is agreement between originator and beneficiary are also not enforced. The SEPA rules involved are:

SEPA message	Element	Rule
pacs.008.(CT)	Transaction Identification	Must contain a reference that is meaningful to the Originators Bank and is unique over time.
pacs.008.(CT)	Remittance Information - Unstructured	Unstructured may carry structured remittance information, as agreed between the Originator and the Beneficiary.
pacs.002(CT)	Original Transaction Identification	Must contain a reference that is unique over time.
pain.001(CT)	Remittance Information - Unstructured	Unstructured may carry structured remittance information, as agreed between the Originator and the Beneficiary.
pacs.003.(DD)	Transaction Identification	Must contain a reference that is meaningful to the Creditors Bank and is unique over time

SEPA message	Element	Rule
pacs.003.(DD)	Remittance Information - Unstructured	Unstructured may carry structured remittance information, as agreed between the Creditor and the Debtor.
pacs.004.(DD)	Transaction Identification	Must contain a reference that is meaningful to the Creditors Bank and is unique over time
pacs.007.(DD)	Original Transaction Identification	Must contain a reference that is meaningful to the Creditors Bank and is unique over time
pain.008(DD)	Remittance Information - Unstructured	Unstructured may carry structured remittance information, as agreed between the Creditor and the Debtor.

Chapter 5. Validation details

The following topics contain a detailed discussion of the WTX Pack for SEPA validation processes. It provides:

- a description of the maps inside the validation framework
- a detailed description of validation framework map naming conventions
- instructions on customizing validation steps
- a discussion of how failures and unexpected results should be handled.

Maps inside the Validation Framework

One purpose of the validation framework is to simplify the calling of the IBM-provided validation steps in the WTX Pack for SEPA. In the world of SEPA/XML the information needed to determine the cause of invalid data may not always be straight forward, and at times it may require several levels of checking in order to achieve meaningful error reporting.

Stepped map process

The stepped map process validates the data against different criteria. For example, the first step may be to validate against a schema and the second step may be to validate against some additional rules. Each map step reports any errors that are found. If the data is good, according to all the map steps performed, then no output is produced by the validation framework. If the data fails a map step, one or more errors will be produced from that step. It is important to understand the independent nature of the map steps, and that each step is designed to look for certain failures. In some cases, data may fail on multiple mapsteps; a failure of an enumeration value in a schema may also present itself as a failure in the map rules.

For example, to process a **pacs_008_003_002** message, the message is validated against the following:

- **pacs_008_003_002 schema**
- SEPA rules not in the schema
- UNIFI rules

Validation framework map naming conventions

To navigate easily through the maps in the validation framework, it is important that you understand the map naming conventions that are used.

The naming convention used in the validation framework maps is a combination of an eight character coded reference, plus a string that matches the SEPA schema naming convention. When looking at the validation framework map names, notice that the dollar sign (**$**), present in the schema name, is not used and periods are replaced by underscores (_).

The first eight characters are significant, as they must be unique for mainframe processing. The naming scheme for the first eight characters in the validation framework map names is as follows:

″se″ + map level + bank/customer + message + level + version + ct/dd

This is illustrated in the following example. Data conforming to **$pacs_008_003_02** schema to be validated against the Credit Transfer Implementation Guide would first pass through the level zero map named:

se0b832c_pacs_008_003_02

″se″ + map level=″1″ + bank/customer=″bank″=″b″ + message = ″008″=″8″ + level = ″003″ = ″3″ + version=″02″=″2″ + ct/dd=″ct″=″c″

The map name is shown below with the first eight characters indicated:

Characters 1 through 8 are defined as follows:

- **Characters 1 and 2** – These characters always begin with the literal **se**.
- **Character 3** – This character indicates the validation framework mapping level. It may be a **0**, or a **1**. There are two validation framework mapping levels:
  - **Level-zero maps** – At the highest level are framework level maps, called level-zero maps. A level-zero map always contains a **0**.
  - **Level-one maps** – At the next level of detail are validation level maps, or level-one maps. These maps perform specific validation for the messages. A level-one map always contains a **1**.
- **Character 4** – This character indicates whether the message is for bank-to-bank (**pacs**) or customer-to-bank (**pain**) use. If it is for **pacs** use, a **b** appears in the character 4 position (as shown in the example). If the message is a **pain** message, a **c** appears in the character 4 position.
- **Character 5** – This character indicates the UNIFI message name with the zeros suppressed. For example, 008 is shown as simply **8**.
- **Character 6** – This character indicates the message level to the primary schema used to validate a message. Zeros are suppressed. Message levels are:
  - 001 UNIFI level, abbreviated as 1.
  - 002 SEPA core level, abbreviated as 2.
  - AOS level, abbreviated as 3.
- **Character 7** – This character indicates the message version of the primary schema used to validate the message. Zeros are suppressed.
- **Character 8** – This character indicates whether the message is used for a credit transfer, or for a direct debit. Credit transfers are indicated when a **c** appears in this character position. A direct debit is indicated when a **d** appears in character position 8.

The remainder of the map name (i.e. **_pacs_008_003_02**) is a direct transformation of the primary schema name. The **$** is dropped, if used, and periods are replaced with underscores. Therefore, **$pacs.008.002.02** would become: **_pacs_008_003_002**.

Types of customization for SEPA validation

This section describes how to:

- add a new schemas
- add a new validation step
- modify a validation step
- modify or add additional validation rules

Note: Customization requires modification of the WTX Pack for SEPA validation maps. Make a copy of the validation maps prior to customization to avoid the need to reinstall the WTX Pack for SEPA if you need to go back to the original maps.

Adding new schemas

If you are not using the published ISO 20022 UNIFI schemas, or the EPC revised schemas published for SEPA, you will have to add any customized schemas to the framework. This may be necessary if you are implementing recommended and mandatory fields in the customer–to bank messages. It could also be appropriate if the message is being sent to another member of an AOS community, and the schema has been modified to more specifically define those message element fields that are permitted by SEPA, for this reason. This process requires tree and map modifications.

Follow these procedures to add a new schema:

1. Add the new schema name to the **sepagen.xsd** schema located in the **mapsandschemas** directory, and save the file. This a simple schema, and contains a choice group with the list of schemas currently used by SEPA. A new element can be inserted as follows:

   ```
   <xs:element name="custom_schema_name" type="xs:string">
   ```

2. Open the WTX Type Designer and modify the existing **sepagen.mtt** file by importing the **sepagen.xsd** schema that you edited in step 1 as a Classic type tree.
3. Place the modified type tree in the **type_trees** directory.
4. Check to make certain that your customized schema is located in the **mapsandschemas** directory.

   Note: You must edit **sepagen.mtt** after it is created to remove any restriction lists that may have been created by the import process. At this point in the validation, we need to treat the bulk of the instance document as a blob of data (disregarding XML tags), until it is passed to the lower level maps that perform the validation

The previous steps will allow an instance document based on your new schema to be processed by the validation framework. However, to perform specific validation, you will need to add a new validation step for this schema, which will be explained in “Adding new validation steps.”

Adding new validation steps

If you are using customized SEPA schemas, you will have to customize the framework, and possibly the validation maps, that the validation framework calls. Here you will find a description of how to add new schemas or customize existing

ones. Even if you are using only the existing SEPA schemas, you may want to add a new map step. One example could be that your overall implementation requires you to process SEPA messages that only contain a certain number of payment transactions, which is less than the number allowed by SEPA.

Creating a new map step involves updating the validation framework maps to:

- Optionally extract a subset of information from the message to make downstream mapping more straightforward, with the use of a NVP WorkTemp. See "About NVP WorkTemps(s)."
- Set up the new step as a call to a functional map.

About NVP WorkTemps(s)

The NVP WorkTemp(s) are used to extract information from the data, and from previous steps, without the need for excess hierarchical functional map calling structure. In other words, there are some things you cannot easily do all on one map rule. NVP WorkTemp(s) are found at both the **sepvalid** map level and the zero level maps (for example **se0b212d_pacs_002_001_02)**. As a general rule, if the information being extracted in a NVP WorkTemp(s) is useful to most of the messages, it should be extracted a the sepavalid map level and passed into the zero level maps as parameters. In the case of adding new custom mapsteps, if you are adding multiple steps, and this information will be used in all the validation maps for those steps, extract at the **sepvalid** map level. For new steps, however, many of the anticipated new pieces of information needed are expected to pertain just to that new mapstep. In this case the NVP WorkTemp(s) in the new step should be used

Add a custom schema

The process for adding a custom schema uses the **sepvalid** map. This process will add a new run map call to the existing validation maps, or could call a new or modified validation map. Follow the procedures below to add a validation step

1. Open **sepa_validation.mms** in the map designer, then select output card 1 of the **sepvalid** map.
2. If desired, created a new NVP WorkTemp under the Global WorkArea. This can then be passed to any of the new validation steps created.
3. Add an index to the FileType(s) group. To do this, first go to the bottom of the **sepvalid** map. Right click on the **FileType(s)** and add an index. This index will have a corresponding call to a level zero (se0) functional map that will call the validation level one (se1) run map for validation. The corresponding level zero map can also contain framework rules.
4. In the FileTypes element group, complete all elements as needed for the new FileType. The following table provides a description of the table groups and associated elements.
5. Create the required functional maps and run maps to perform the validation required. You may wish to copy other maps within the validation framework as a starting point.
6. If you wish to use an NVP WorkTemp at the functional map level, these are found in the output card as an indexed group under the Step WorkArea element.

Group	Action
StepDescription Element Group	Add a description of the step.

Group	Action
StepWorkArea Element Group	Create indexes as needed in the **NVP WorkTemp**.
Operation Group	**RUNCall** Element = Used for running map commands. **PUTCall** Element = Used for putting files or queues etc., using a WTX adapter. **DBCall** Element = Used for connecting to an external database for framework activity.
Step Results Element Group	Used for reporting step results.

Modifying validation steps

These steps describe how to modify validation steps.

1. 1. Navigate to the desired functional map in **sepvalid** that needs to be updated. Notice that the functional map names begin with **se0**. The framework maps all have this naming convention, while the validation run maps all start with **se1**. These maps are also referred to as level zero and level one maps.

   Note: The 8 character prefix name for all of the maps must be consistent in order to facilitate deployment to z/OS
2. 2. Add an index to the **STEP(s)** element. **The STEP (s)** element is always at the bottom of the output card in the functional map.
3. 3. In the **STEP [n]** element, complete the following groups and associated group elements as described in the table in the **Adding a validation step** section.

Modify or add validation rules

In many cases, you will have to either add additional validation rules, or perhaps in the case of the **pain.xx** message types, where all of the recommended rules in the SEPA implementation guidelines are included, you may need to remove some rules.

Refer to the following section**Guidelines for modifying or adding validation rules** when modifying or adding validation rules. To add error messages, refer to the **Adding error messages** section.

Guidelines for modifying or adding validation rules

Observe these guidelines when modifying or adding validation rules.

- all of the rules are in the **se1*** run maps
- rules are added by creating a 'Rule' index
- rules are organized in the order that they appear in the SEPA Implementation Guides
- comments in each existing rule refer to the correct index in the SEPA Implementation Guides

Adding error messages

All of the rules call the **seplib05_extract_error_msg** functional map. This map is used for the framework and the validation maps to create error messages in the **seperror.xml** output file. Two values are echoed. The first value, **SEP0019E**, in this rule corresponds to the error number in the **sepamsgs.xml** file that contains all of the error messages used in the framework and validation maps. The second value is passed into the {0} variable in the **sepamsgs.xml,** and can be any text string data.

When you want to add additional error messages to the sepamsgs.xml, make certain that you add them in the same format as the existing messages. Each message will have 3 trans-unit-ids:

- Error Message (EM)
- Extended Error (EE)
- Error Action (EA)

String values from the map are passed into the EM id using the {0} variable. Any additions to this file must be added in ascending trans-unit id order.

Handling failures and unexpected results

Between the validation framework built-in debugging tools and general WTX debugging techniques it is possible to obtain information about map and data format failures, as well as and unexpected results.

The first step that should be taken if any unexpected results are encountered, is to consult output card 3 (**seperror.xml** by default). This output contains an XML structure that describes any framework or validation errors in end-user terms. The next section describes output card 3 validation errors. If output card 3 is checked, and the reason for the failure is not apparent, consult output card 1. Refer to the Output card 1 section for details of this procedure.

For a description of the SEPA error parser and instructions on how to run the **sepa_error_parser** utility map, see "SEPA schema error parser" on page 27.

Checking output card 3

Output card three shows the name of the file being validated, the name of the (optional) original file to be validated, the start and end date/times and any errors. Please note that if there are no errors reported by the validation framework for a map run then this output will not be created. The following is an example of an output card 3 showing output errors:

Example of output card 3 validation errors

Output card 3 validation errors are illustrated in the following example:

```
<MapInfo>
<FileToValidate>C:\Program Files\IBM\WebSphere Transformation Extender
8.1\packs\sepa_v1.0\mapsandschemas\sep000.xml</FileTovalidate>
<OriginalFile>C:\Program Files\IBM\WebSphere Transformation Extender
8.1\packs\sepa_v1.0\mapsandschemas\sep000.xml</OriginalFile>
<Start>20070320154850</Start>
<End>20070320154853</End>
<Validation>
<Description>
<Message id> = "SEP0008E Service Level(SvcLvl) must be present in the
Document/pacs.008.001.01/GrpHdr/PmtTpInf or all occurrences of the
Document/pacs.008.001.01/CdtTrfTxInf/PmtTpInf, but not both</Message>
<Explanation>SEPA error: (AT-40 Identification code of the Scheme) 'Service
Level' must be present at either the Group level - Payment Type Information
or all occurrences of the Payment Type Information at the Credit Transfer
Transaction Information level, but not both</Explanation>
<Action>Ensure the Service Level from either the Group Header - Payment
Type Information or all occurrences of the Credit Transfer Transaction
Information - Payment Type Information group are present, but not
```

```
both</Action>
</Description>
</Validation>
</MapInfo>
```

Checking output card 1

If the reason for the failure is still not apparent after consulting output card 3, check output card one. First, create this file, if calling **sepvalid** from outside of Map Designer, or in Map Designer change the output card settings from **sink** to **file** and specify an output file location (a suggested name is **out.xml**). By default, this card is suppressed (card setting of **sink**) from creation. This XML instance document will contain a log of all processing during the run of the **sepvalid** map. It will also list all of the temporary variables used and show the various map steps performed and their results.

You will be able to use this output to track the steps of the framework and determine on which map the failure is occurring, or the unexpected results are being introduced.

SEPA schema error parser

If you encounter a SEPA schema validation error when SEPA validation is performed using the maps in **sepa_validation.mms**, you will be presented with a **SEP0300E** or **SEP0301** error message in the error log. If the data being passed in is a CORE or CORE plus AOS message, there will also be error messages that will allow you to determine what the schema error was. If, however, you are using the ISO 20022 UNIFI schemas, there are many elements in these schemas that SEPA does not use. In this event, only the **SEP0300E** or **SEP0301E** errors will appear in the error log. To help to determine what these errors are, you can use the utility map called **sepa_error_parser.mms**.

The **sepa_error_parser.mms** will run the SEPA data file through schema type tree validation with the TX trace option turned on. In order to expedite diagnosis of errors, this map will create a subset data file containing only the bad transactions. The XML trace log will be parsed, which will then be used by this framework to retrieve the error data information for the input data. The list of data errors will be presented as the output to the framework, which will show the line numbers of the data that is in error, as well as the element that failed validation. After the framework completes the process, refer to the subset bad data file which will be presented as card 4 in the utility map (**errordat.xml**) and the error log to correct schema errors in the data.

Running the error parser

Follow these instructions to run the **sepa_validation.mms** map:

1. Pass your SEPA data file into the **errparse** run map with an override to input card 1.
2. Retrieve the error log, which will be output card 3. The contents will refer to the subset data file that was used to create the error log. Use this subset data file to locate and correct the schema errors.

   Note: If you are a z/OS user, you must modify the **sepa_error_parser.jcl** located in the **mapsandschemas** directory. You can then deploy to your system.

Chapter 6. Domestic format support

At the present time, the WTX Pack for SEPA provides domestic support for the following countries:

- the United Kingdom
- Germany
- France

Domestic format for the United Kingdom

This section describes the BACS domestic format converter for the United Kingdom (U.K.).

Note: This format is available for purchase from APACS at www.apacs.org.uk, and references Standard 18 - 1 October 2002.pdf BACS Interchange Standards (v18/9 dated 1 October 2002)

The BACS standard contains two formats for data records: BACS input and BACS output. Both of these formats have the same basic field structure of 100 bytes, but the BACS output format is extended by additional fields. The BACS standard also defines Volume and File and User Header records which vary in size. The BACS input and output formats are described in as follows:

- **BACS input** - This format is used by Banks, and their customers, to send payment data to BACS by electronic transfer, or other means. After initial validation, the data is forwarded to the relevant bank(s) using BACS output format. The BACS input format can be either 100 or 106 bytes. The additional 6 bytes are used to specify individual processing dates within BACS. For full details on this process, refer to the BACS User Manual.
- **BACS output** - The BACS output is always 120 bytes. The additional 20 bytes contain fields added after validation by BACS: Error Code, BACS User Number and BACS Reference (unique reference for each payment; used by BACS for query purposes).

The SEPA type tree and converter maps are based upon the output format. If conversion to and from the input format is required, it can be easily achieved since the base 100-byte format is identical, although the extra data provided by the BACS output format is useful in providing additional unique identifiers for each transaction.

There are several types of data records in BACS format. These data records are identified by Transaction Code (TX Code) and vary depending upon whether the sender is a Bank or a Bank's Customer. The SEPA type tree contains a single file description containing all possible record types.

Domestic format example for the United Kingdom

Example files that describe the United Kingdom domestic format are located in the following directory:

*install_dir***packs****sepa_v***n.n****examples****uk**

Refer to the **readme_uk.txt** file for complete instructions on how to use these example files.

Domestic format for Germany

The DTAUS data format was defined within the overall online banking standard that was announced in 1995 by the German banks represented by their associations participating in the Central Banking Committee (ZKA - Zentraler Kreditausschuss). The full standard is the ″Homebanking Computer Interface (HBCI)″. This format is used for both interbank and customer-bank payments processing in Germany.

In the DTAUS format, a physical file contains:

- a single file header also called Record A
- one or more payment exchange messages called Record C
- a single file trailer called Record E)

There is a single format used for both direct debit or credit transfer, identified by Record A (File Header) Field 3 (Identifier). However, a logical file must contain only credits or only direct debits (no mixed payment records). Record A contains the sender and receiver of the file, and is a fixed length of 128 bytes. Each Record C contains details of the orders to be executed (credits or debits) and contains a constant and a variable length section. Record E is used for performing checks.

Domestic format example for Germany

Example files that describe the domestic format for Germany are located in the following directory:

*install_dir***packs****sepa_v***n.n****examples****germany**

Refer to the **readme_germany.txt** file for complete instructions on how to use these example files.

Domestic format for France

There are two domestic format converters for France:

- MINOS
- CFONB

CFONB domestic format converter for France

The CFONB organization defines additional formats used in payments processing for the Customer-Bank processes.

This format, used in the examples for France, is described in *Remises informatisees d'ordres de paiement International, au format 320 caracteres*. This document is available through a catalogue on the web site: **www.revue-banque.fr/editionCatalogue.do?shortcut=edition_catalogue**. This format will be replaced by an XML based format on which the French banking community works currently, and by the SEPA PAIN (payments initiation) format in the case of SEPA payment operations.

It should be noted that the CFONB format is close to the MINOS format, but each bank has the freedom to adapt the format to its specific needs.

Payment message record types

Each CFONB320_PI payment message is identified by the value of PI in positions 3 to 4 of each record and is composed of the following record types:

- Mandatory header line identified by the first two characters of: **03**.
- Group(s) of detail records consisting of the following:
  - Mandatory detail line identified by the first two characters of: **04**.
  - Optional (dependant) line for details related to the Beneficiary Bank. The first two characters are: **05**.
  - Optional line for details related to the Intermediate Bank. The first two characters are: **06**.
  - Optional line for additional information related to the transfer. The first two characters are: **07**.
- Mandatory total line. The first two characters are:**08**.

MINOS format converter for France

The MINOS domestic format converter for France is defined in the Handbook of SIT Interbank Transaction Standards, and is administered by GSIT, the French Interbank Teleclearing System.

This handbook contains the structure of the transactions used for interbank payment information. This document is available to order from the web site: **www.gsit.fr/Gb/information/minos.htm**.

For the MINOS format, there are various payment types handled through the current system, but only a very few correspond to the payments processing defined by the SEPA rules and datasets. The first payment types of interest are the Virements (sweeps, credit transfers), which are the most common kind of operations handled through the interbanking in France. These map to SEPA Credit Transfer processing. The credit transfer is identified by the 'code opération' in the SIT part of the record = 120. The second payment type of interest is the Prelevement (direct debit) which correspond to SEPA Direct Debit processing. Different types of direct debits exist, however, the direct debit example included with the WTX Pack for SEPA uses the message with a 'code operation' = 180 ('Prélèvements 4 jours).

MINOS distiguishes two message directions:

- 'Aller' – from the Bank to the Clearing House (ACH)
- 'Retour' – from the Clearing House to the Bank

Each of these directions can handle different types of operations (Credit transfer, Direct Debit)

'Remise aller' is a payment file containing the following:

- A header record
- 1-N payment operations of a same category and for the same settlement date:
  - Same category means : same type of operation (credit/debit), same currency, same urgency, same end of day (time period)
- A summary record containing control information:
  - Number of operations
  - Total amount

– Signature

'Remise retour' (remittance ACH to bank)) : a payment file containing:

- A header record
- 1-N payment messages to be posted in the same basket, composed of :
  – A header record containing the installation number of the issuer
  – 1-N operations issued by the same sender
  – A summary record (number of operations, total amount
- ⌂ A summary record containing control information
  – Number of messages
  – Total amount
  – Signatures

Domestic format example for France

For the SEPA converter examples for France, two type trees are provided:

- CFONB320 which contains the type tree definitions for the formats that can be defined by "Remises Informatisées D'Ordres de Paiement International" (CFONB320_PI), and "Remises Informatisées D'Ordres de Virement National France" (CFONB320_VF). Only the CFONB_PI format is used in the example maps.
- MINOS which contains the type tree definitions for many of the MINOS formats

Example files that describe the domestic format for France are located in the following directory:

*install_dir***packs****sepa_v***n.n****examples****france**

Refer to the **readme_france.txt** file for complete instructions on how to use these example files.

SEPA format converters for SWIFTNet FIN

This documentation describes the SEPA format converter for the SWIFTNet FIN format.

At the present time, mapping guidelines exist only for the SWIFTNet FIN MT 103 message.

SWIFTNet FIN format example

Example files that describe the SWIFTNet FIN format are located in the following directory:

*install_dir***packs****sepa_v***n.n****examples****swift**

Refer to the **readme_swift.txt** file for complete instructions on how to use these example files.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Trading Partner
Tivoli

WebSphere
z/Architecture
z/OS
zSeries

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

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender Pack for SEPA Version 1.0

Index

T

U

V

X

Z

Printed in USA