IST/MAS System Operations Guide

V1.4
January 2018
XMSOG0118

Table of Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Appendix B

Appendix C

1

Introduction

Purpose of this Document

The *IST/MAS* System Operations Guide describes the architecture and the server processing of the Merchant Accounting System. This manual also instructs administrators how to install, configure, and manage the system.

How this Manual is Organized

The structure of this document provides three tiers whose contents are directed to different audiences. These modules are as follows:

1. *High-level overview* - describes the architecture of the Merchant Accounting System. The information in this tier is chiefly intended for system administrators and operators who require information on the *IST/MAS* product architecture.
2. *Application-level module* - provides instructions for system installation, configuration, and setup. This second tier lists the necessary parameters to be set up and the sequence of entry according to configuration flows.
3. *Technical specification* - provides processing details on the *IST/MAS* Server. This tier is intended for a technical audience.

Related Documents

Readers should refer to the following documents for information that supplements this manual:

Documents	Description
Entitlement Service Guide	Provides a comprehensive technical description of the setup and configuration of the Entitlement Security Server. Administrators should refer to this manual for installation instructions, defining permissions, setting up user groups, and setting up operator profiles.
IST/MAS Business Operations Guide	Provides overview information and business setup information about IST/MAS. Administrators and operators should refer to this manual for product features, definitions of business concepts, and acquiring entity setup and administration.
IST/MAS Table Reference Guide	Provides descriptions for all database tables in the *IST/MAS* data model. Administrators should refer to this manual for descriptions of the tables containing setup or statistics information.
IST/Foundation Administration Guide	Provides information on the *IST/Foundation*. System administrators should refer to this manual for the IST/Foundation installation and configuration and for details on its components.
IST Product Installation Guide	Provides information on the installation procedures for *IST/Foundation* and *IST/MAS*.

Revision Log

Date	Effective W/Version	Chapter	Change
January 2018	V1.4	3	Updated the Platforms and Databases.
November 2016	V1.4	3 4 5	• Added the following: ❍ Direct Import ❍ Direct Import from Database ❍ Direct Import Transaction from IST/Switch
			• Updated the following figures: ❍ IST/MAS Processing—Architecture ❍ *IST/MAS—Logical Processing Flow* ❍ Transaction Processing—Logical Processing Flow
			• Updated the following sections: ❍ Transaction Processing on page 14 ❍ Batch Validation Request on page 32 ❍ Processing Overview on page 43 ❍ Main Processing on page 44

Date	Effective W/Version	Chapter	Change
			❍ Processing Flow on page 58 ❍ Transaction Pricing on page 61.
July 2015	V1.4	Appendix A	● Added the following fields to Input File: ❍ orig_ext_trans_id ❍ trans_date_time ❍ retrieval_ref_nbr ● Added the following fields to Input File, Adjustment Input File and Remittance Input File: ❍ lookup_key ❍ curr_code1 ❍ conv_amt1 ❍ curr_exp1 ❍ curr_code2 ❍ conv_amt2 ❍ curr_exp2 ❍ curr_plan_inst_id
June 2014	V1.4		Initial release of the manual.

2

IST/MAS System Overview

Architecture

The *IST/MAS* is a client-server application. The *IST/MAS* Client provides a user-friendly interface to the system, and the *IST/MAS* Server provides the actual transaction processing.

The client-server architecture provides controlled data access and data security, scalability, flexibility, and interoperability. The server can be accessed remotely and across multiple platforms, so accessibility is another advantage of this architecture.

See "Component Relationships" on page 11 for a brief overview of the *IST/MAS* components and their relationships.

See "IST/Foundation Subsystems" on page 12 for a list of IST/Foundation.

Component Relationships

The *IST/MAS* is a client-server application. The *IST/MAS* Client provides the interface to the system, and the *IST/MAS* Server provides the actual transaction processing.

The *IST/MAS* Client is a Java-based application that provides the Graphical User Interface (GUI) to the system. The *IST/MAS* Client itself has a client-server architecture containing the GUI Client and the GUI Server. The GUI Client provides a set of forms which allow users to enter data required by the acquiring entity setup. The GUI Server, which runs in a Web Server environment, manages several GUI Servlets that connect the GUI Client with the *IST/MAS* Database.

The following figure shows the relationships among these components:

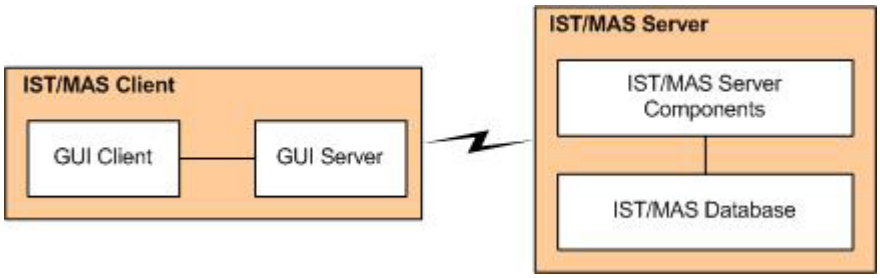


Figure 1 *IST/MAS* General Architecture

For details on the GUI architecture refer to *Developing Java GUI Applications*.

See *"IST/MAS Server Processing" on page 43* for details on the *IST/MAS* Server processing.

IST/Foundation Subsystems

Since *IST/MAS* is built on an IST/Foundation architecture, several IST/Foundation subsystems have been used to create *IST/MAS* processing components. Knowing how these components are implemented is not required for the end-user, that is the operator. System administrators and application developers, however, must know which subsystems have been used if they want to performs additional system configuration and customization.

- The Oasis Virtual Machine (OVM) has been used as the base processing engine and the Mailbox Subsystem as the inter-process communication system. For details on these subsystems, see the *OVM Subsystem Guide* for details.
- The Universal Agent Subsystem has been used to handle input and output files. For details on this subsystem, see the *Universal Agent Guide*.
- The Run-time Management (RTM) Subsystem has been used to implement the Logging/Auditing component. For details on this subsystem, see the *Run-Time Management (RTM) Subsystem Guide*.
- The GUI Infrastructure Subsystem has been used to create the Merchant Account Management GUI. For details on this subsystem, see *Developing Java GUI Applications*.
- The Authentication Subsystem and the Entitlement Subsystem have been used to handle the user authentication and entitlement. For details on these subsystems, see the *Authentication Subsystem Guide* and the *Entitlement Subsystem Guide*.

If you require more information on the IST/Foundation product, refer to *IST/Foundation Administration Guide*.

IST/MAS Components

This section provides a brief description of the *IST/MAS* processing components and their role in transaction processing. As seen in Figure 2 on page 13, several client and server components are involved in processing.

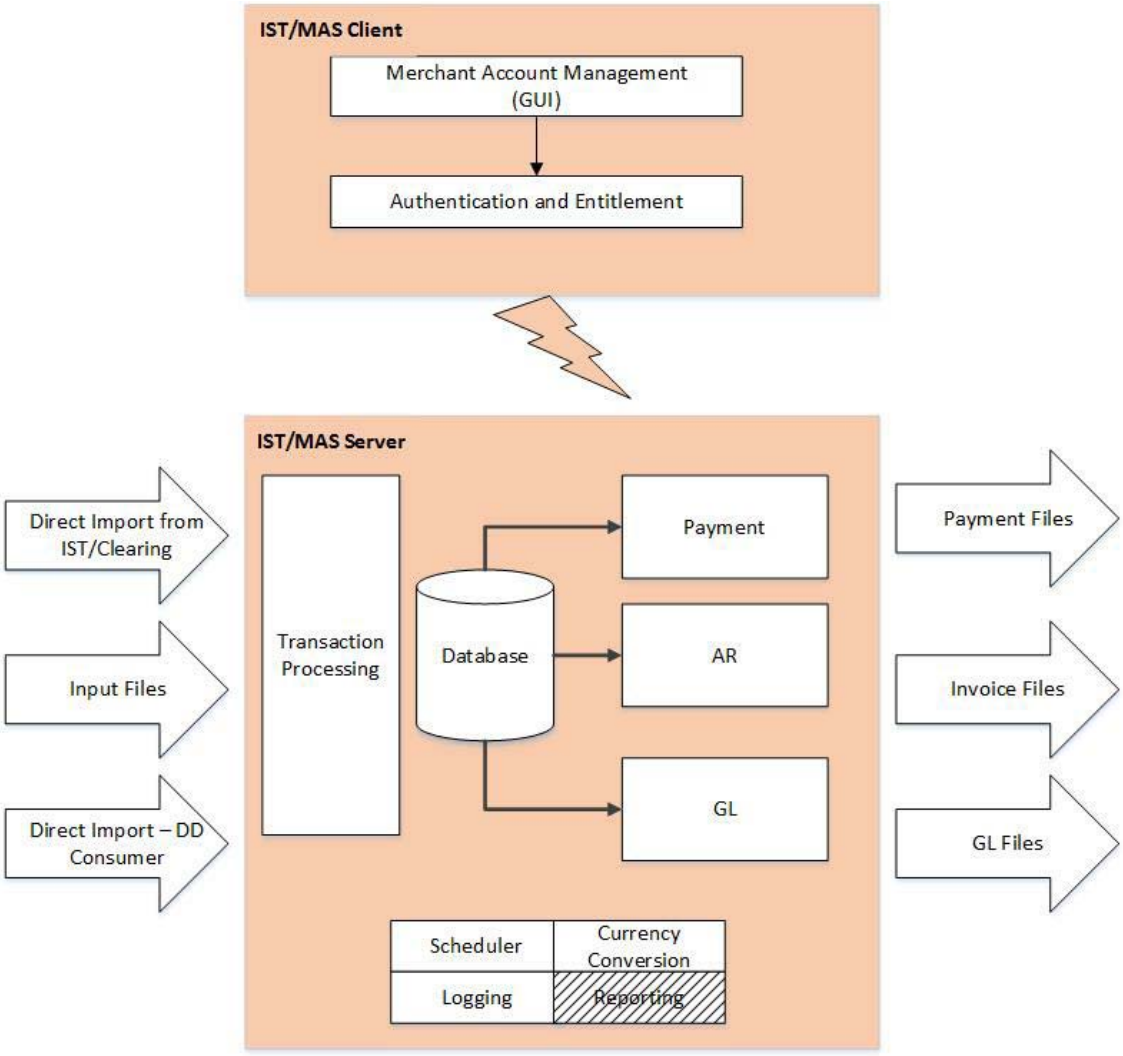


Figure 2 *IST/MAS* Processing—Architecture[1]

1. The Package ID of the default fee package will be taken from the configuration file. This package will be probably assigned to the top level-Entity in the merchant hierarchy.

The system is accessible via a Graphical User Interface (GUI), which restricts the access through the use of an authentication and entitlement component. The actual processing is performed by the *IST/MAS* server, which is always running in the background.

IST/MAS Server Components

As seen in the architecture figure (Figure 2 on page 13), the server side of the system performs the actual transaction processing. Two groups of components can be identified on the server side: processing and supporting components.

Processing Components

These components are directly involved in transaction processing:

- Transaction Processing—Transactions coming from input files or directly from database are processed by the Transaction Processing component and the output information is saved into the database.
- Payment Component—Payment information from the database is further processed by the Payment component to generate settlement records, which are saved in Payment Files.
- Account Receivable (AR)—Fee information from the database is further processed by the Account Receivable (AR) component to generate Invoice Files.
- General Ledger information from the database is further processed by the General Ledger component to generate GL files.

See *"IST/MAS Server Processing" on page 43* for details on the processing performed by the Server's components.

Supporting Components

These components perform various tasks which assist transaction processing:

- Scheduler—Based on the business rules of the acquiring entity[1], this component dictates the processing window for the system and performs tasks scheduling for all system components.
  See "Scheduler" on page 154 for details.
- Currency Conversion—This component performs currency conversion for the transaction amounts and fee amounts posted to the acquiring entity's accounts when these amounts are not in one of the currencies defined for the entity.
  See "Currency Conversion" on page 161 for details.
- Logging—This component performs low-level logging by saving processing events and errors into a log table (for example, errors writing to the database).
  See "Logging" on page 160 for details.

1. Note that throughout the rest of the manual, *merchant* and *acquiring entity* have been used interchangeably.

- Reporting—Although it does not have a reporting model, the system prepares reporting data into the database, and a third-party report generator can be used to create various reports. Also, the system creates several reports files which are used to monitor the system's processing.
  See "Reporting" on page 164 for details.

IST/MAS Client Components

As seen in the architecture figure (Figure 2 on page 13), the client side of the system performs the acquiring entity management via a Graphical User Interface (GUI). The access to the system is restricted through the use of an authentication and entitlement component that sits between the GUI and the rest of the system.

Merchant Account Management (GUI)

Via the Merchant Account Management interface, operators and administrators may perform the following:

- General system setup and maintenance
- Acquiring entity account setup and maintenance
- Task scheduling
- System reporting and inquiry
- Payment, posting, reporting, and logging configuration

For configuration and processing purposes, the system provides multiple roles within a hierarchy of entities, and the roles are definable through a multi-tier parent-child relationship. Additional levels may be defined by the application. One or more roles may be attributed to each entity, supporting (but not limited to) the following roles:

- Billing destination
- Payment destination
- Revenue allocation
- Reporting delivery destination
- Database query access
- Command access

Note that roles and entities in the acquiring entity hierarchy may be synonymous with roles defined through the Entitlement Server's User Access Control. This would support all of the access and control operations for both internal and external (merchant, ISO, institution) roles in a consistent and continuous fashion. This may also be interpreted as multi-institutional support.

Acquiring entity information supported includes, but is not limited to, the following:

- Configuration parameters for payment, billing, and reporting according to standard entities and roles within the acquiring entity hierarchy
- Calendar specifying business working days and other time driven processing characteristics
- Accounting (MAS) codes in use for the acquiring entity
- Rates associated with each rate group within each MAS code
- Card types allowed for the acquiring entity
- Associated General Ledger Accounts
- Settlement Direct Deposit Accounts (DDAs)

For details on the functionality and the processing capabilities of the Merchant Account Management component, refer to the *IST/MAS Business Operations Guide*.

Authentication and Entitlement

The Entitlement Subsystem works together with the Authentication Subsystem to ensure that users entering the system and accessing various information are authorized to do so. However, their responsibilities do not overlap, but complement each other. The Authentication Subsystem authenticates all users attempting to login, i.e., verifies whether the user is known to the system. Once a user is authenticated, the Entitlement Subsystem is used to restrict access to *IST/MAS*.

The Entitlement Server is a security interface which may be configured by the system administrators to permit users, based on their membership within specific groups, to access such information as account activity, acquiring entity fees, and configuration parameters. The following examples illustrate the general restriction of operations:

- Database access operations (inquiry, update, and delete) are restricted to specific user roles. For example, a sales representative could be entitled to view completed payment transactions, but might not be entitled to view pending payments.
- Passwords are not viewable by any role at any time. Passwords automatically expire according to a standard configurable time duration, but users are able to change passwords at will. The role of the security administrator is enabled to reset a password to a standard value.

For more information on how to create users and assign privileges, see the *Entitlement Subsystem Guide*.

3

IST/MAS Setup and Administration

General Installation Procedure

Installing, configuring, and running *IST/MAS* depends on the platforms and databases selected, the requirements of the system, and the tools offered by the application. Refer to the following topics for details:

- See "Platforms and Databases" on page 18 for a list of platforms and databases supported by *IST/MAS*.
- See "*IST/MAS Software Requirements" on page 18* for a list of application required to run the *IST/MAS* Server and *IST/MAS* GUI.
- See "Installing and Configuring the System" on page 19 for the general procedure to follow to install the entire system. Since *IST/MAS* has a client-server architecture, several installation scenarios are possible. For convenience, one scenario has been selected.
- See "Configuring the Authentication and Entitlement Subsystems" on page 20 for instructions on how to customize the two subsystems responsible for the secure access to the *IST/MAS* system.
- See *IST Product Installation Guide* for installing and configuring the Web Server and the GUI Server.

Platforms and Databases

The following combinations of operating systems and databases are supported by *IST/MAS*:

O/S Version	C++ Compiler	DB Version
Solaris 11	C++ compiler Version 5.12	Oracle 11g, 12c DB2 10.1, 11.1
AIX 7.1,7.2	C++ compiler Version 13.1.3	Oracle 11g, 12c DB2 10.1, 11.1
HP-UX 1131	C++ compiler Version 6.25.02	Oracle 11g, 12c DB2 10.1, 11.1
RHEL 6.7, 6.8, 6.9	C++ compiler Version 4.4.7	Oracle 11g, 12c DB2 10.1, 11.1

IST/MAS Software Requirements

IST/MAS Server Requirements

The following applications are required to run the *IST/MAS* Server.

Application	Comments
Perl Interpreter	Required for running the installation script.
C++ Compiler	Required only if you need to start the Oasis Virtual Machine (OVM), that is, not required for the run-time version of *IST/MAS*.

IST/MAS GUI Requirements

The following applications are required to run the *IST/MAS* GUI.

Application	Comments
Java Environment	Java Development Kit (JDK) or Java Runtime Environment (JRE)
Web Server/Servlet Container	Use a combination of Web Server and Servlet container (for example, iPlanet Web Server, Tomcat)
Web Browser	Internet Explorer with a Java plug-in
	Netscape with a Java plug-in

Installing and Configuring the System

NOTE:	Since *IST/MAS* has a client-server architecture, several installation scenarios are possible. For the purpose of this installation procedure, IST/Foundation, *IST/MAS* Server, and *IST/MAS* Client are all installed on the same UNIX machine. The Java environment and the Web browser are installed on a PC running Windows 2000.

For *IST/MAS* to be completely operational, a strict installation and configuration sequence must be followed:

1. Install and configure IST/Foundation.

   See *IST Product Installation Guide* for installation details of IST/Foundation and *IST/MAS*.

   See *IST/Foundation Administration Guide* for installation and configuration details. Note though that two subsystems require special attention.
   See "Configuring the Authentication and Entitlement Subsystems" on page 20 for details.

2. Install and configure *IST/MAS* Server and its database.

   See *IST Product Installation Guide* for installation details of *IST/MAS*.

   See "Configuring the *IST/MAS Server" on page 21* for installation details.

3. Install and configure the Web Server and the GUI server.

   See *IST Product Installation Guide* for installing and configuring the Web Server and the GUI Server.

4. Install a Java environment.

   Since the Merchant Account Management is a Java application, Java Run-time Environment (JRE) or Java Development Kit (JDK) must be installed. Refer to the appropriate product installation for details (*http://java.sun.com*).

5. Install a Web Browser.

   The Web browser is used to run the GUI Client and display the forms defined for the Merchant Account Management GUI.

   Both Internet Explorer and Netscape are supported by *IST/MAS*.

Configuring the Authentication and Entitlement Subsystems

Access to the *IST/MAS* is restricted via Authentication and Entitlement. With their installation default, however, these subsystem are of little use. Further customization and configuration is required.

Authentication Subsystem

To authenticate users, the Authentication Subsystem requires a database of users, a connection to the database, an administrator, and values for the authentication rules, like password life time or connection life time.

For details on setting up the Authentication Subsystem, see the *Authentication Subsystem Guide*.

Entitlement Subsystem

The Entitlement Subsystem comes pre-configured with generic and specific objects (like institutions, forms, or merchants) and their associated permissions. The system administrator, which is the only pre-configured user, must define additional users, select the object to which the user has access, and grant permissions.
See "Entitlement Setup" on page 216 for details on entitlement objects currently available in *IST/MAS*.

Based on their roles, users defined in the system may belong to groups. Several default groups are pre-configured ("Predefined Groups" on page 219); however, you may need to create additional groups to satisfy your specific requirements. The following roles may provide a basis for building system privileges:

- Sales Representative
- Agent User
- Merchant End User
- Bank End User
- Report Operator

- System Administrator
- Accountant
- Manager
- Auditor
- Update User
- Clerk

For details on user entitlement configuration and administration, see the *Entitlement Subsystem Guide*.

Configuring the *IST/MAS* Server

Configuration Parameters

The system is set up via several configuration parameters in the *$OSITE_ROOT/istparam.cfg* file. If not found in this file, the system searches for configuration parameters in the istparam.cfg file from the IST/Foundation installation.

Although most configuration parameters already have defaults supplied, some parameters may need customization before the system can function properly.

1. Mailbox Subsystem—The mailbox subsystem, which is the main process initiated at system startup, requires two task.tsk parameters to initialize the Scheduler and the Account Accumulator.

   ```
   task.tsk sch_mon temp
   task.tsk mas_acc
   ```

   The Scheduler process is required to schedule all processes involved in transaction processing, like Validation, Pricing, or Posting. The Account Accumulator process is required for rate re-assessment.

   For details on other configuration parameters you may want to customize, see the *Mailbox Subsystem Guide*.

2. Database Subsystem—The system uses database tables to save the system configuration and system processing information. No changes are required here since the database configuration is done through the installation script.

   For details on other database parameters you may want to customize, see the *Database Manager (DBM) Subsystem Guide*.

3. *IST/MAS* processes—The following table lists the product-specific configuration parameters, which are located in the *$OSITE_ROOT/cfg/istparam.cfg* file.

Parameter Name	Description
	File Recognition
mas.base_institution	Sets the ID of the institution owning *IST/MAS*.
mas.fr_ua_dir	Sets the location of the setup files for the Universal Agent that implements the File Recognition process.
mas.fr_in_dir	Sets the directory where incoming files are saved. Default value: *$OSITE_ROOT/data/in*.
mas.fr_done_dir	Sets the directory where File Recognition saves processed incoming files. Default value: *$OSITE_ROOT/data/done*.
mas.fr_reject_dir	Sets the directory where File Recognition saves rejected incoming files. Default value: *$OSITE_ROOT/data/reject*.
mas.fr_proc_dir	Sets the directory where File Recognition saves incoming files that are currently being processed. Default value: *$OSITE_ROOT/data/processed*.
mas.fr_dll	Defines the name of the callback library holding the file recognition callback function. Used for defining on-line file processing windows.
mas.fr_func	Defines the name of the callback function executed by the Scheduler at 60 seconds intervals. This function checks for incoming files for processing. Used for defining on-line file processing windows.
	Administrative Parameters
mas.tracelevel	Sets the trace level for all *IST/MAS* processes.

Parameter Name	Description
mas.*n*_dir	This parameter sets the directories where *IST/MAS* stores the output files it generates. The *n* variable within the name of the parameter represent the file type and is given by values configured via the MAS_FILE_TYPE table. Currently, the following values are configured: ● 51—GL Export File (mas.51_dir). ● 52—ACH Payment File (mas.52_dir) ● 53—Wire Payment File (mas.53_dir) ● 54—Outbounder File (mas.54_dir) ● 55—Invoice File (mas.55_dir) ● 56—Manual Wire Payment File (mas.56_dir) ● 57—GL End of Day and End of Cycle report files (mas.57_dir) ● 58—Payment Log report file (mas.58_dir) If this parameter is not set, the file is saved in the current directory.
mas.rtmeventlevel	Defines the RTM event level. Values: ● 1—information and error events of level 1 are saved to the debug files. ● 2—information and error events of level 1 are saved to the debug files. See "Event Levels" on page 219 for details on event levels.
mas.cmd_mailbox	Defines the name of the mailbox the business server (i.e., the mas_daily server) uses to receive messages.
mas.use_msq	1/0 flag setting the system to use an alternative IPC communication for Tru64 platforms. Using the message queue over shared memory between ensures that read and writes to the shared memory are synchronized.
mas.hashsize_entity_acct	Size of the memory cache used to store the ENTITY_ACCT table records. Default value: 4096.
mas.hashsize_acct_accum_gldate	Size of the memory cache used to store the ACCT_ACCUM_GLDATE table records. Default value: 4096.
mas.hashsize_acct_posting_plan	Size of the memory cache used to store the ACCT_POSTING_PLAN table records.
mas.hashsize_batch_summary	Size of the memory cache used to store the BATCH_SUMMARY table records. Default value: 4096.
mas.hashsize_fee_pkg_mas_code	Size of the memory cache used to store the FEE_PKG, MAS_CODE table records. Default value: 4096.
mas.hashsize_fee_pkg_tid	Size of the memory cache used to store the FEE_PKG_TID table records. Default value: 4096.

Parameter Name	Description
mas.hashsize_field_value_ctrl	Size of the memory cache used to store the FIELD_VALUE_CTRL table records. Default value: 4096.
mas.hashsize_gl_chart_of_acct	Size of the memory cache used to store the GL_CHART_OF_ACCT table records. Default value: 4096.
mas.hashsize_acct_accum_det	Size of the memory cache used to store the ACCT_ACCUM_DET table records. Default value: 4096.
mas.hashsize_settle_plan_tid	Size of the memory cache used to store the SETTL_PLAN_TID table records. Default value: 4096.
mas.hashsize_mas_code_grp	Size of the memory cache used to store the MAS_CODE_GRP table records. Default value: 4096.
mas.hashsize_inst_calendar	Size of the memory cache used to store the INST_CALENDAR table records. Default value: 4096.
mas.hashsize_acq_entity	Size of the memory cache used to store the ACQ_ENTITY table records. Default value: 4096.
mas.hashsize_fee_pkg	Size of the memory cache used to store the FEE_PKG table records. Default value: 4096.
mas.hashsize_calendar	Size of the memory cache used to store the CALENDAR table records. Default value: 4096.
mas.hashsize_business_day	Size of the memory cache used to store the BUSINESS_DAY table records. Default value: 4096.
mas.hashsize_holiday_link	Size of the memory cache used to store the HOLIDAY_LINK table records. Default value: 4096.
mas.single_curr	A 1/0 flag that sets the system to enable or disable multi-currency feature. Default value: 1.
mas.zero_file_dir	This parameter sets the directory where IST/MAS stores files with zero file size (empty incoming file). Default value: *$OSITE_ROOT/data/done*.
mas.dbcommit_interval	This parameter allows user to configure the time period interval in seconds when transactions are committed to the database.
mas.hashsize_mas_trans_log	Size of the memory cache used to store the MAS_TRANS_LOG table records. Default value: 4096.
mas.hashsize_holiday_list	Size of the memory cache used to store the HOLIDAY_LIST table records. Default value: 4096.
mas.max_reserve_shm_rec	Size of the shared memory used to store the RESERVE PAYMENT records. Default value: 100.
mas.acc_commit_interval	Number of transactions to be processed by MAS Accumulator, before committing them to DB. Default value:20000.

Parameter Name	Description
mas.post_flush_interval	Number of transactions to be processed by MAS POST, before flushing them to DB. Default value: 20000.
	Scheduler
sch.default_institution	Provides a default institution ID. The value is used when the -i parameter is not used in the Scheduler command line.
sch.default_usage	Provides a default product code. The value is used when -u parameter is not provided in the Scheduler command line. Values: MAS, CLR.
sch.tracelevel	Sets the trace level for the Scheduler.
sch.rtmeventlevel	Defines the RTM event level for the Scheduler.
sch.stop_dir	Directory where the Scheduler looks for the file called *stop*, which is used to stop file processing.

Example:

```
mas.tracelevel                          0x00000023
mas.rtmeventlevel                       1
mas.base_institution                    1234567890
mas.fr_ua_dir                           $OSITE_ROOT/ua
mas.fr_in_dir                           $OSITE_ROOT/data/in
mas.fr_proc_dir                         $OSITE_ROOT/data/process
mas.fr_done_dir                         $OSITE_ROOT/data/done
mas.fr_reject_dir                       $OSITE_ROOT/data/reject
mas.cmd_mailbox                         MAS_DAILY
mas.fr_dll                              libomas.DLL
mas.fr_func                             process_incoming
mas.51_dir                              $OSITE_ROOT/data/gl
mas.52_dir                              $OSITE_ROOT/data/ach
mas.53_dir                              $OSITE_ROOT/data/wire
mas.54_dir                              $OSITE_ROOT/data/out
mas.55_dir                              $OSITE_ROOT/data/inv
mas.56_dir                              $OSITE_ROOT/data/mwire
mas.58_dir                              $OSITE_ROOT/data/pay
mas.use_msq                             1
mas.hashsize_acct_entity_acct           4096
mas.hashsize_acct_accum_gldate          4096
mas.hashsize_acct_posting_plan          4096
mas.hashsize_batch_summary              4096
mas.hashsize_fee_pkg_mas_code           4096
mas.hashsize_fee_pkg_tid                4096
mas.hashsize_field_value_ctrl           4096
mas.hashsize_gl_chart_of_acct           4096
mas.hashsize_acct_accum_det             4096
mas.hashsize_acct_settle_plan_tid       4096
sch.default_institution                 1234567890
sch.default_usage                       MAS
sch.tracelevel                          0x00000023
sch.rtmeventlevel                       3
sch.stop_dir                            $OSITE_ROOT/data/in
```

Database Tables

Database tables are used to set up acquiring entities or customize the system processing. When *IST/MAS* is installed, the installation utility creates the database for the entire system and handles all default values required for the system to be operational. However, no transaction processing can be done without first setting up acquiring entities via the Account Management GUI.

For setup details, see *IST/MAS Business Operations Guide*.
For a complete database description, see *IST/MAS Table Reference Guide*.

Administering *IST/MAS*

Once configured and brought up, *IST/MAS* needs little or no supervision from a system administrator. If individual processes or the entire system terminates or reports errors, however, the system administrator must intervene.

This section presents procedures that may be used to administer the system:

- Bringing up and down the system.
  See "Bringing Up the System" on page 27.
  See "Bringing Down the System" on page 28.
- Updating the system's configuration.
  See "Updating the System Configuration" on page 28.
- Setting debug levels.
  See "Setting Debug Levels" on page 29.
- Stopping and resuming processing.
  See "Stopping and Resuming Processing" on page 29.
- Scheduling tasks.
  See "Scheduling Tasks" on page 30.
- Defining file processing windows.
  See "Defining File Processing Windows" on page 35.
- Scaling the system.
  See "Scaling the System" on page 36.
- Recovering from processing errors.
  See "Recovering from Processing Errors" on page 38.

See the *Mailbox Subsystem Guide* for details on other commands used to administer the system's infrastructure.

Bringing Up the System

IST/MAS has a client-server architecture, and when the entire system is brought up, both Client and Server must be started.

1. Start the database server.

   Refer to the appropriate documentation for details on how to run the database server.

2. Bring up the *IST/MAS* Server by issuing an *mbinit* command at the UNIX prompt.

   This command brings up the mailbox subsystem, the OVM, and all other *IST/MAS* components.

3. Bring up Web Server either via its URL or directly from UNIX.

   The Web Server will automatically run the GUI server and load the Servlets. Refer to the appropriate Web Server documentation for the appropriate commands.

4. The GUI Client is brought up by accessing the Client's URL from a Web browser. The URL has the following format:

   ```
   <machine_name>.<Web_server_ID>.startMAS.html
   ```

   where -

machine_name	is the name of the machine where *IST/MAS* is installed.
Web_server_ID	is the Web Server identification, which consists of the URL and the port number where the server is connected (URL:<port_no>).
startMAS.html	is the root HTML file.

Bringing Down the System

Although they are independent, both Client and Server must be shut down to bring down the entire system.

The Server is brought down by issuing the *mbkill* command at the UNIX prompt.

The *IST/MAS* Client is brought down in two steps.

1. First the Web Server is brought down either via its URL or directly from UNIX.

   The Web Server will automatically shut down the GUI server and the Servlets. Refer to the appropriate Web Server documentation for the appropriate commands.

2. The GUI Client is brought down by closing the browser window.

   After confirmation is provided by the user, the system closes down the client.

Updating the System Configuration

IST/MAS reads its configuration information during system initialization. To replace the current configuration setup you must update istparam.cfg and bring the system down and up again.

Setting Debug Levels

The system saves by default the debug information in several debug files under the *$OSITE_ROOT/log/debug* directory.

The amount of debug information saved in these files depends on the debug levels set by the mas.tracelevel configuration parameter and the RTM event level set by the mas.rtmeventlevel configuration parameter. For the Scheduler, the corresponding parameters are sch.tracelevel and sch.rtmeventlevel.
See "Event Levels" on page 219 for details on event levels.

The following debug files are generated by the system:

- mas_fr[n].debug—Each File Recognition instance has a debug files associated with it. The name of the debug file includes the instance number.
- mas_pos[n].debug—Each Posting instance has a debug files associated with it. The name of the debug file includes the instance number.
- mas_daily[n].debug—Each Daily Process instance has a debug files associated with it. The name of the debug file includes the instance number.
- sch_mon.debug—This is the debug file for the Scheduler process.
- sch_cmd.debug—This is the debug file for the Scheduler command line interface.
- mas_ua.debug—This is the debug file for the Universal Agent component.
- mas_acc.debug—This is the debug file for the Accumulator process.
- error.debug—This is the debug file where *IST/MAS* saves severe errors, that is, errors that stop processing (like database or mailbox errors).

NOTE:	Since no automatic cleanup is performed, it is recommended to purge these files on a daily basis by renaming the debug files and saving them in a separate directory.

Stopping and Resuming Processing

As long as input files are present in the incoming directory, *IST/MAS* keeps processing these files. This situation is not desirable when the database has been corrupted or cannot be accessed, for example.

To stop the system from processing, place in the stop directory a file named *stop*. The Scheduler scans constantly this directory for the stop file. As long as the file is present, the transaction processing is stopped.

The name of the stop directory is given by the sch.stop_dir configuration parameter. By default, the parameter points to the incoming directory.

To resume processing, remove the file from the incoming directory.

NOTE:	The scheduled tasks that are not executed during the no-processing time have to be manually rescheduled.

Alternatively, you can do either:

- Start or stop file recognition by manually initiating the Start/Stop File Recognition commands from the Scheduler command line.
  See "Manual Scheduling" on page 34.
- Start or stop file recognition by defining file processing windows.
  See "Defining File Processing Windows" on page 35.

Scheduling Tasks

The Scheduler module provides the capability of launching tasks at pre-defined times of the day, days of the week or month. Through this module you are able to schedule, manually or automatically, the on-line file processing and the daily operation and maintenance.

To schedule tasks, you must first set up the Scheduler. You must ensure that the Scheduler will be initialized when *IST/MAS* is brought up, and that tasks are scheduled for execution.
See "Scheduler Setup" on page 31 for setup details.
See "Task Execution Sequence" on page 32 for a task scheduling sample.

Once the Scheduler tasks are set up, starting and stopping the on-line file processing and daily operations and maintenance can be done several ways:

- File Processing—Stop and resume transaction processing by placing or deleting a *stop* file in a directory monitored by the Scheduler. This method requires file recognition to be either always on, or processing windows be defined.
  See "Stopping and Resuming Processing" on page 29.
- Ad hoc File Recognition—Start or stop transaction processing by manually initiating the Start/Stop File Recognition commands from the Scheduler command line.
  See "Manual Scheduling" on page 34.
- Automatic File Recognition—Start or stop transaction processing by setting up appropriate records in the TASKS table via *IST/MAS* GUI.
  See "Automatic Scheduling" on page 35.
- Processing Windows—Allow processing within pre-defined intervals via file processing windows.
  See "Defining File Processing Windows" on page 35.

The Scheduler command logs information and errors in the *$OLOGDIR/sch_cmd.log* file.

Scheduler Setup

To setup the Scheduler, you must ensure that the Scheduler server is running and schedule the desired commands.

The Scheduler is automatically brought up when the *IST/MAS* is launched if a task entry is defined in istparam.cfg. An example entry looks like this:

```
tsk.task        CurrentNode sch_mon temp
```

The Scheduler continuously peeks into the Scheduler database and identifies tasks that are scheduled to be run. Scheduling tasks requires GUI input into the database.

1. Although the TASK_CMD_LIST table comes pre-loaded with default *IST/MAS* supported commands, ensure that all commands required for processing are listed there.

   A task is identified by command number, usage, and command parameters. The Usage column identifies the business application. The same command number between different business applications can perform very different tasks.

2. Schedule in the TASKS table all commands and their execution times.

   A task is uniquely identified by a Task Number, and a tasks number is not related to a command number. Many server tasks may execute the same command. Scheduling is done considering the run type and the calendar used as follows:

   - When RUN_TYPE is set to ‘D’, RUN_SUN through RUN_SAT is used to identify the day of week to run.
   - When RUN_TYPE is set to ‘M’, RUN_DAY is used to identify the day of month to run.
   - When RUN_TYPE is set to ‘O’, RUN_MONTH and RUN_DAY is used to identify the date to run.
   - All RUN_TYPE values except ‘N,’ observe the RUN_HOUR and RUN_MIN.
   - When RUN_TYPE is set to ‘N’, the task is executed immediately.
   - If CAL_ACTIVITY_ID contains a value, the Scheduler calls the Calendar API to resolve whether to run based on the current date (that is, holiday breaks).

3. View details on the tasks execution in the TASK_LOG table.

   The SCHEDULE_START_DT column is populated with the date-time when the Scheduler decides that a particular task should be executed. When the task command message has been sent successfully, the Scheduler populates the ACTUAL_START_DT and ACTUAL_FINISH_DT columns. The scheduler also sets the RESULT to ‘EXECUTED’. In the case of an incorrect setup, which leads to the Scheduler not able to send a task launch message, the RESULT column is set to ‘NO CMD MAILBOX’.

Task Execution Sequence

The sequence in which the Scheduler runs processes is very important since this order dictates the *IST/MAS* processing flow. This is a sample task sequence for a normal scenario:

1. Transaction Processes
    a. File Recognition Request (every 60 seconds). When the file recognition processing is done, the system automatically starts the Posting process.
    b. Batch Validation Request (every 60 seconds). When the Batch validation processing is done, the system automatically starts the Posting process.
    c. Non-Activity Fees Request. When the non-activity fees processing is done, the system automatically starts Validation, Pricing, and Posting.
    d. Adjustment Request (can be run multiple times a day)
2. Payment, AR, and GL Processes—The payment and AR processing, which ends with a cycle balancing request, is repeated throughout the day as many times as needed to handle all payment cycles defined in the system. Payment cycles are defined to distribute processing. Basically, merchants are assigned to a payment cycle, and all merchants assigned to the same payment cycle are handled together.

a. Processing for Payment Cycle 1

   i. Invoice Generation Request. This request starts the generation of the Invoice, Outbounder and Payment Log files.

   ii. Remittance Request (can be run multiple times a day)

   iii. Aging Request

   iv. ACH Payment Request. This request also generates the Outbounder and Payment Log report files.

   v. Wire Payment Request. This request can be also scheduled for the end of the day so that only one Wire Payment file is created. This request also generates the Outbounder and Payment Log report files.

   vi. Manual Wire Payment Request. This request also generates the Outbounder and Payment Log report files.

   vii. Cycle Balancing Request

b. Processing for Payment Cycle 2

c. Processing for Payment Cycle 3

d. ... (Processing for all payment cycles must be finished before the next step is performed)

e. GL Export Request

f. EOD Balancing Request

3. Ad hoc Processes—These processes are run whenever required.

   a. Resubmit Suspended Transactions Request

   b. Write off Suspended Transactions

   c. Resubmit MAS Transaction Error Request

   d. Expected Merchant's Transmission Request

4. End of Day Processes—These processes are run at the end of the day.

   a. Next Settlement Date Request

   b. Rate Re-assessment Request

   c. Fee Group Rate Update Request

   d. Mas Fee Future Effective Date Request

5. End of Month Processes—These processes are run at the end of the month.
    a. MAS Summary Request
    b. Payment Count Request
6. Schedule the task invocation through the GUI.

Parallelism in MAS EOD and Export Process

IST/MAS have the ability to handle the tasks:

- in parallel for different institution
- in sequential for same institution

Let us consider the following scenario:

- If a task is running in the system for one of the institution and if some other/same task is scheduled for the same institution it waits until the current task ends. For example: Schedule a task for the payment file generation for an institution and schedule the invoice file generation for the same institution. The invoice file generation should wait until the payment file generation completes.
- If the task is running in the system for one of the institution and if some other/same task is scheduled for another institution then the task is executed in parallel to the current task in the system. For example: Schedule a task for the payment file generation for an institution and schedule the invoice file generation for different institution.The invoice file generation should process in parallel with payment file generation.

Manual Scheduling

To manually schedule tasks, run sch_cmd from the command line. The format of the command line is as follows:

```
sch_cmd -i<institution_ID> -u<usage> <task_ID>
```

where -

institution_ID	provides the ID of the institution for which the command is run.
usage	provides of the business application using the Scheduler. Sample values: MAS (for *IST/MAS*), CLR (for *IST/Clearing*).
task_ID	provides the numeric ID assigned to a Scheduler task.

All commands take two options, institution ID (-i) and usage (-u). These options assign an institution and a product to a scheduler command. If this information is not provided, default values are taken from istparam.cfg, sch.default_institution and sch.default_usage parameters. For each command however, the system may prompt for additional information.

Example:

```
sch_cmd -i1234567890 -uMAS 100
```

The available tasks are defined in the TASK_CMD_LIST table.
Refer to *IST/MAS Table Reference Guide* for a complete list of defined tasks.

Automatic Scheduling

Each individual business application employing the Scheduler must have a server running that accepts task-launching ELF messages and launches the tasks accordingly. For *IST/MAS*, the mas_daily server accepts the ELF messages, and the server is automatically initialized via apm.src.

To receive scheduled commands, each of the business applications employing the Scheduler have to set-up its individual application server. To identify these servers to the Scheduler server, the application must specify the servers' mailbox. The mailbox name is given via a configuration parameter in istparam.cfg with the following syntax:

```
<usage>.cmd_mailbox <mb_name>
```

where -

usage	application domain name of the business server.
mb_name	mailbox name of the business server.

For example, the following configuration identifies that the task server of *IST/MAS* is called MAS_DAILY.

```
mas.cmd_mailbox MAS_DAILY
```

When the Scheduler prepares and tries to launch a task, it needs to find out the mailbox name of the server that should receive the task launch message. This is done by mapping the <usage>.cmd_mailbox to the configured mailbox name of the application's task server. Following the above example for *IST/MAS*, the Scheduler sends all task launch messages belonging to *IST/MAS* to the mailbox MAS_DAILY.

Defining File Processing Windows

If an on-line file processing window is desired, the Scheduler provides a way to have such processing windows defined and activated. This is done through a command toggling mechanism with special commands.

For *IST/MAS* the special commands are 100 (toggle File Recognition on) and command number 200 (toggle File Recognition off).

IST/MAS must provide calling information on its file recognition method via the following istparam.cfg parameters:

```
mas.fr_dll                  libomas.DLL
mas.fr_func                 process_incoming
```

If such parameters are defined, the Scheduler maintains in memory a file recognition status for *IST/MAS*. When a command number 100 (toggle FR on) is scheduled, the Scheduler change the recognition status to ON, while a command number 200 (toggle FR off) will change the status to OFF. Upon restart, the initial file recognition status is always set to OFF.

If the file recognition status is set to ON, the Scheduler calls the corresponding callback function as defined in istparam.cfg at 60 second intervals.

Direct Import

Similar to on-line file processing mentioned in above, the transaction can be imported from database also.

This is done through a command 101 toggling mechanism with parameter "DIR_IMP_START" from field_value_ctrl table. When the parameter DIR_IMP_START set to 'Y' then batch validation will be started, while DIR_IMP_START set to 'N' then it stop batch validation.

Scaling the System

To handle sites with a high volume of transactions, the system must be scaled by adjusting the number of processes that are running simultaneously. This is configured in the apm.src configuration file.

Since performance is influenced by many factors, like available memory or number of central processing units (CPU), there is no fast rule about how many processes the system should run to achieve the best performance. However, you can scale the system with the following procedure:

1. Using the default apm.src values, determine the system's performance by calculating the number of transactions processed per second (tps).

To calculate the system's tps, run the system for a day and determine from the Batch Summary table the total number of records processed and the number of seconds the system has run (use the date-time stamp in the record). The tps is calculated by using the following sql:

```
SELECT (begin_rec_cnt),
(MAX(end_date_time) - MIN(begin_date_time)) * 24 * 3600,
SUM(begin_rec_cnt) / ((MAX(end_date_time) - MIN(begin_date_time))
* 24 * 3600)
FROM BATCH_SUMMARY
WHERE file_id = '?'
and suspend_flag is null;
```

2. Open the *$OSITE_ROOT/cfg/apm.src* file and modify the entry corresponding to the process requiring scaling.

   To run five Validation processes, find the entry that has MAS_VAL in the Mailbox field and change the Number of copies field to five. (The fields are shown in bold and red.)

```
mas_val test2 localhost null oasisvm null null OSITE_ROOT ovm/
mas_val.cfg MAS_VAL null 5 0 0x0000ffff 10 001003
```

The following mailbox names are defined in *IST/MAS*:

Process	Mailbox Name
File Recognition	MAS_FR
Validation	MAS_VAL
Validation (Txn Level)	MAS_VLD
Non-activity	Fees MAS_FEE
Posting	MAS_POST
Daily Processes	MAS_DAILY
Universal Agent	MAS_UA
Direct Import	MAS_DIR_IMP
Direct Import Validation	MAS_DIR_VAL

For details on apm.src, see the *IST/Foundation Administration Guide*.

1. Bring down the system and up again to reload the new apm.src values.

2. Recalculate the system's tps.

   a. If the tps is higher than the one calculated before, consider increasing again the number of processes.

   b. If the tps is the same or has dropped, reverse the increase.

NOTE:	The minimum requirement to achieve parallelism is to keep the number of instance equal for MAS_DAILY and MAS_UA.

Recovering from Processing Errors

This topic presents the common points of failure in the *IST/MAS* processing and the actions taken by the system. To place the recovery procedures in context, each topic briefly describes the performed processing.

Rejected File—File Recognition

In File Recognition, a file can be rejected if it does not satisfy certain physical checks:

- Incorrect file name
- Incorrectly formatted file header or trailer
- Incorrect data type in a field in a record
- Invalid checksum

File Recognition uses the Universal Agent to check the integrity of the incoming file. It reads each record, performs various checks as outlined above, and sends it to Transaction Validation where it is validated and queued until all records from the file have been received.

If File Recognition finds a problem with any record, it sends an administrative message to Transaction Validation instructing it to discard the records (transactions) being validated and queued. The file is subsequently moved to the reject file directory for manual inspection.
See "File Recognition" on page 49 for details.

No loss can occur since all transactions are queued before pricing processing begins, not even when File Recognition has to be restarted. A file is not logged as processed until all checks are successful (e.g. checksum) and therefore will remain in the "in" directory. The File Recognition process always looks in the "in" directory for files to process. Once the file recognition process is restarted the file will be reprocessed automatically.

If a file is rejected (i.e., moved to the "reject" directory), locate and repair the file, and send it to the inbound file directory for reprocessing.

Rejected Batch - Direct Import Process

In direct import process, a batch can be rejected if it does not satisfy certain physical checks:

- Incorrect data type in a field in a record
- Invalid checksum
- Duplicate batch

Direct import uses the Universal Agent to check the integrity of the batch records from database. It reads each record to corresponding batch, performs various checks as outlined above, and sends it to Transaction Validation where it is validated and queued until all records from the batch have been received.

If Direct import finds a problem with any record, it sends an administrative message to Transaction Validation instructing it to discard the records (transactions) being validated and queued. Then value of imp_status from the table mas_in_batch_log will remain 'R'.

No loss can occur since all transactions are queued before pricing processing begins, not even when Batch validation has to be restarted. A batch is not logged as processed until all checks are successful (example: checksum) and therefore will remain in the status 'R' for imp_status.

The direct import process always looks in the mas_in_batch_log table for record process. Once the direct import process is restarted, the batch will be reprocessed automatically.

Suspended Transactions—Validation

All transactions in the file/database is validated and buffered before release to Pricing. Validation verifies the following:

- The data is validated against the stored database values (example: valid MAS code, entity ID)
- Transactions are enriched with required data for efficiency.
- A record is created in MAS_FILE_LOG to record the receipt of the file. This record also contains the starting and ending Transaction Sequence Number assigned to transaction records in the file. The number of batches in the file is also recorded.
- A record is created in BATCH_SUMMARY for each batch in the file. This entry contains the number of records in the batch, and it is used to determine the Transaction Sequence Numbers of each transaction should transactions need to be recovered due to a failure.

If Validation encounters problems individual transactions, batches, or even entire files may be suspended.
See “Transaction Validation” on page 53 for details.

If Validation must be re-run, the following scenarios may occur:

- If no transactions are released to Pricing, then the file is still in the “in” directory and will be processed automatically.
- If Validation is in the middle of releasing transactions to Pricing, then an unbalanced situation will be encountered.
  See “Lost Transactions—Posting” on page 41 for the procedure to follow to recover an unbalanced situation.

Suspend Level

The level of suspension applied depends on the indicator received in the incoming activity file.

Levels	Description
File level	Suspend the file if any one transaction fails validation. All transactions are written to MAS_TRANS_SUSPEND, where they can be reported on or inspected via an inquiry screen. In this case all transactions in the file are validated before release to Pricing and Posting.
Batch level	Suspend the batch if any one transaction in the batch fails validation. A batch is released only when all transactions in the batch pass validation. If a transaction fails validation, all transactions within that batch are written MAS_TRANS_SUSPEND, where they can be reported on or inspected via an inquiry screen.
Transaction level	Suspend the transaction if it fails validation. In this case, only transactions that fail validation are suspended.

Recovery Procedure

If transactions are suspended in Transaction Validation (logged in MAS_TRANS_SUSPEND), find the offending batch, fix the transaction and resubmit it for processing. *IST/MAS* will locate and process only those transactions that have not yet been processed.

The stored Transaction Sequence Numbers that were allocated during Validation is used to determine which transactions have already been priced and posted.

Error Transactions—Pricing

Activity transactions received from Validation have already been validated and enriched with required data for processing. However, a failure may occur when a fee transaction is generated. For example, failure may occur if an entity’s Settlement Plan does not contain the specific fee entry.

In this case the offending transactions are logged in MAS_TRANS_ERROR, where they can be updated and resubmitted to Posting.

Lost Transactions—Posting

These are transactions that were present in the file but were never posted to the MAS_TRANS_LOG. In this case, it is necessary to reprocess the file that contained the lost transaction. Files that were processed with lost transactions are placed in the “done” directory, since the entire file would have been processed successfully.

If an alert message, indicating that processing was shutdown is received, it means that the cause of the lost transactions is due to a database failure:

Transactions may be lost due to several reasons:

- Stopping processes inadvertently—If a process crashes and has to be re-run, then an unbalanced situation will be encountered.
- Inability to post to the database—In Posting, exceptions may occur if the process cannot post to the database. To assist in the recovery process, the SQL statement is logged in the debug file. *IST/MAS* provides an SQL extract process that builds an SQL script used to manually post these entries.

  If the number of database failures exceeds a predetermined number, the Posting process sends an administrative message to the other processes to cease processing and drop all transactions. In addition, the system, via RTM, logs an alert in one of the debug files. This information can be picked up by a monitoring interface for an operator to take appropriate action. Transactions dropped in this manner can be recovered by reprocessing the file.
- Problems in the mailbox—Under certain exception conditions, messages can be queued in a mailbox. When the mailbox is killed, transactions may be lost. The number of transactions lost depends on the number of servers and the value of the mb.default_throttle parameter.
- Disk exhausted—This can occur when the database is unable to extend table space, which eventually leads to a lost transaction scenario. To assist in the recovery process, the SQL statement is logged in the debug file.
- RAM Exhausted—This is not common in normal processing since, during system initialization, memory is allocated depending on the number of processes and the maximum size of the input file. When sizing the system, ensure that the amount of RAM specified is enough to process the largest expected file times 3. If this situation occurs, however, an unbalanced situation will be encountered.

To recover lost transactions, follow the procedure:

1. Run the recover process *$OSITE_ROOT/log/debug/recover.pl*, a Perl script, to generate the recover.sql script, which will be saved in the current directory.
2. Audit the entries in the recover.sql script. Entries in the SQL script are automatically recorded, so each entry must be carefully examined to decide whether they are appropriate for posting.

3. Post the recovered transactions by running the SQL script. You may use, for example, sqlplus.

4. Find the name of the file that was not completely processed, and reprocess the file using the reprocess command (CMD 120). This will find the offending transactions and process them.

   If a transaction cannot be accounted for during balancing, it means that the origin of the lost transaction is unknown. For example, the rare occurrence of a transaction lost in mailbox will result in an unbalanced system.

   To recover the lost transactions, find the name of the file that was not completely processed, and reprocess the file using the reprocess command (CMD 120). This will find the offending transactions and process them.

4

IST/MAS Server Processing

Processing Overview

The server portion of *IST/MAS* performs the actual transaction processing based on the parameters received through the GUI. Even so, several server processes are directly involved in the processing, and some are only providing support for the first group of processes.

The main processes are responsible for validating the activity transactions received from input files/database, assigning fees to these transactions, and exporting to files the amounts that should be credited to or debited from the entities participating in the chain. See “Main Processing” on page 44 for details.

Supporting processes are responsible for providing currency conversion, logging and scheduling services to the main processes. See “Supporting Processing” on page 45 for details.

Main Processing

IST/MAS operates according to configured times defined through the Scheduler. According to the system calendar, the Scheduler instructs the system to read prepared input files from the incoming directory at specified periods in case File Recognition is scheduled, if Direct import is scheduled then scheduler instructs to read batch records from database table. IST/MAS scans the file path to determine if incoming files are present, checks the files and logs them.

The following diagram presents the transaction (input file) processing flow:

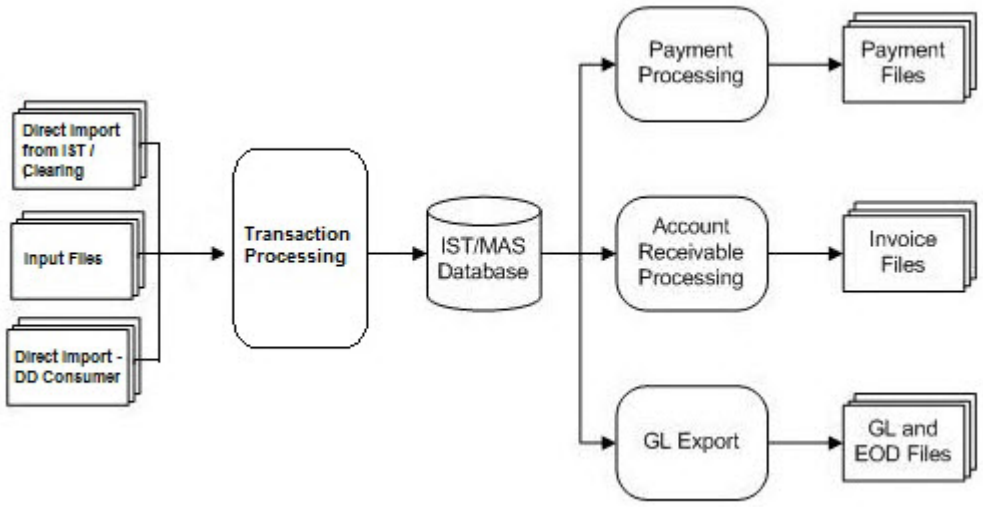


Figure 1 *IST/MAS*—Logical Processing Flow

The system receives information via input files/database, which contain merchant-generated (activity) transactions. These files are handled by internal processes to create output files. For this purpose, the following processing is performed:

1. Transaction Processing

   During transaction processing, the input files and the transactions within the file are validated against a set of processing criteria, like file format and transaction ID. Once validation is passed, transactions are priced, meaning that processing fees are attached to transactions, and fee and payment amounts are posted to the merchant's accounts defined in the *IST/MAS* database.
   See "Transaction Processing" on page 46 for details.

2. Payment Processing

   During payment processing, the information saved by the Transaction Processing is used to create payment files. Basically, these files contain the amounts that have to be paid to merchants for all processed transactions. Note, however, that fees charged for processing may be directly deducted from payments, in which case the

paid amounts represent both payments and fees. Using a third-party transfer utility, Payment files are sent to a financial institution for the actual money transfer to be performed.
See "Payment Processing" on page 125 for details.

3. Account Receivable (AR) Processing

   During account receivable processing, the information saved by the Transaction Processing is used to create invoice files. Basically, these files contain the amounts that merchants have to pay for the processing fees. Using a third-party transfer utility, Invoice files are sent to a financial institution for invoice generation.
   See "Account Receivable (AR) Processing" on page 137 for details.

4. GL Export

   During GL processing, the information saved by the Transaction, Payment, and Account Receivable processing is used to create GL and End of Day (EOD) files. Using a third-party transfer utility, GL files are sent to a GL system for further processing.
   See "General Ledger (GL) Processing" on page 147 for details.

Supporting Processing

As identified by the architecture section ("IST/MAS Server Components" on page 14), several components are used to assist the main processing components.

- Scheduler—based on the business rules of the acquiring entity, dictates the processing window for the system and performs tasks scheduling for all system's components.
- Logging and Auditing—performs low-level logging by saving processing events and errors into a log table (for example, errors writing to the database).
- Currency Conversion—performs currency conversion for the transaction amounts and fee amounts posted to the acquiring entity's accounts when these amounts are not in one of the payment currencies defined for the entity.

See "Supporting Processing" on page 154 for details.

Transaction Processing

Transaction processing prepares the information for Payment, Account Receivable and GL processing.

Transaction Processing Flow

The following diagram presents the transaction processing flow.

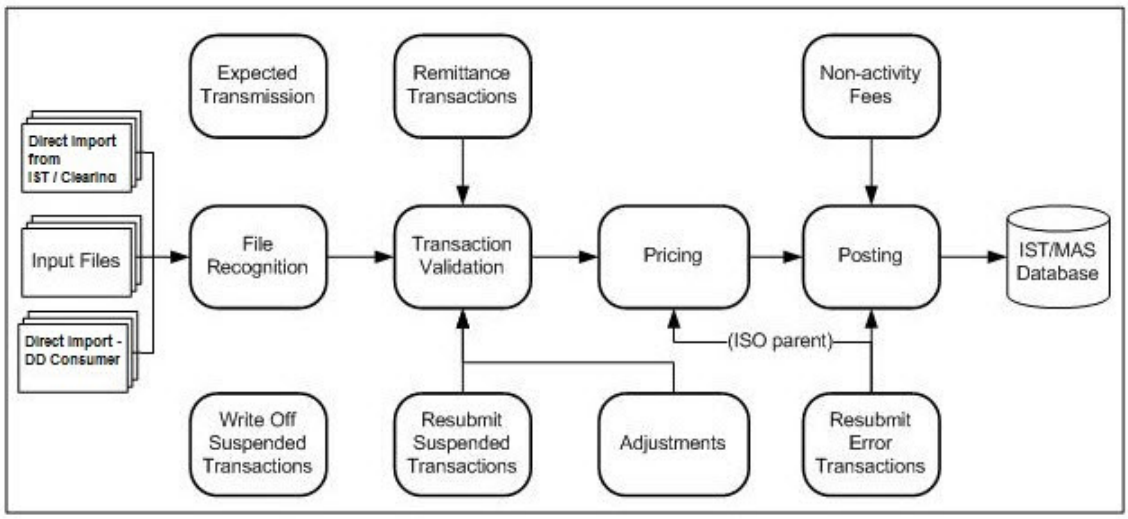


Figure 2 Transaction Processing—Logical Processing Flow

IST/MAS operates according to configured times defined through the Scheduler. According to the system calendar, the Scheduler instructs the system to read prepared input files from the incoming directory at specified periods in case File Recognition is scheduled, if Direct import is scheduled then scheduler instructs to read batch records from database table. IST/MAS scans the file path to determine if incoming files are present, checks the files and logs them.

1. File Recognition

   The first step in the transaction processing flow is the File Recognition processing. File Recognition checks the type of the input file, ensures that it conforms to the expected file formats, and checks the file's integrity. Recognized files are passed on to the transaction Validation process, while unrecognized files are rejected.

Within files, transactions can come either in single or multiple batches, and files may contain one or more types of transactions. Transactions typically received within input files include, but are not limited to, the following:

- Clearing transactions—merchant activity, at detail or summary levels (Merchant, MAS Code, Transaction Code, and Process date-time).
- Accepted chargeback transactions—calculated fees to charge merchants for the number of chargeback transactions.
- Authorization transactions—number of authorizations by Merchant, Authorization Type, Card Types, and Vendor.
- Remittance Transactions—payments received from the merchants for invoices issued.

See "File Recognition" on page 49 for details.

2. Transaction Validation

   Transactions within the Input File are individually validated and enriched with information required by transaction Validation. Transactions that do not pass the validation step are suspended, and the Resubmit Suspended Transactions process must be invoked to resubmit them.
   See "Transaction Validation" on page 53 for details.

3. Transaction Pricing

   After the transaction is validated, it is sent to transaction Pricing to calculate fees and charges to be applied to the transaction and generates fee transactions.
   See "Transaction Pricing" on page 61 for details.

4. Transaction Posting

   Following transaction Pricing, transaction Posting posts the priced activity transaction and the associated fee transactions to the various merchant entity and GL accounts. Also, non-activity rates and fees are calculated periodically by the Non-Activity Fees and submitted to the Posting process.
   See "Transaction Posting" on page 112 for details.

5. Resubmit Suspended Transactions

   Transactions which are suspended due to errors in setup or transaction validation can be resubmitted to the Validation process through the Resubmit Suspended Transactions.
   See "Resubmit Suspended Transactions" on page 60 for details.

6. Write Off Suspended Transactions

   Suspended transactions that cannot be resubmitted are written off through the Write Off Suspended Transactions process.
   See "Write Off Suspended Transactions" on page 61 for details.

7. Resubmit Error Transactions

   Transactions which are suspended due to pricing and posting errors can be resubmitted to Pricing and Posting through Resubmit Error Transactions.
   See "Resubmit Error Transactions" on page 93 for details.

8. Adjustments

   The Adjustments process can be used to adjust payments and fees if the merchant was improperly charged, transactions have been improperly priced, or billing requisitions are generated.
   See "Adjustments" on page 92 for details.

9. Remittance Transactions

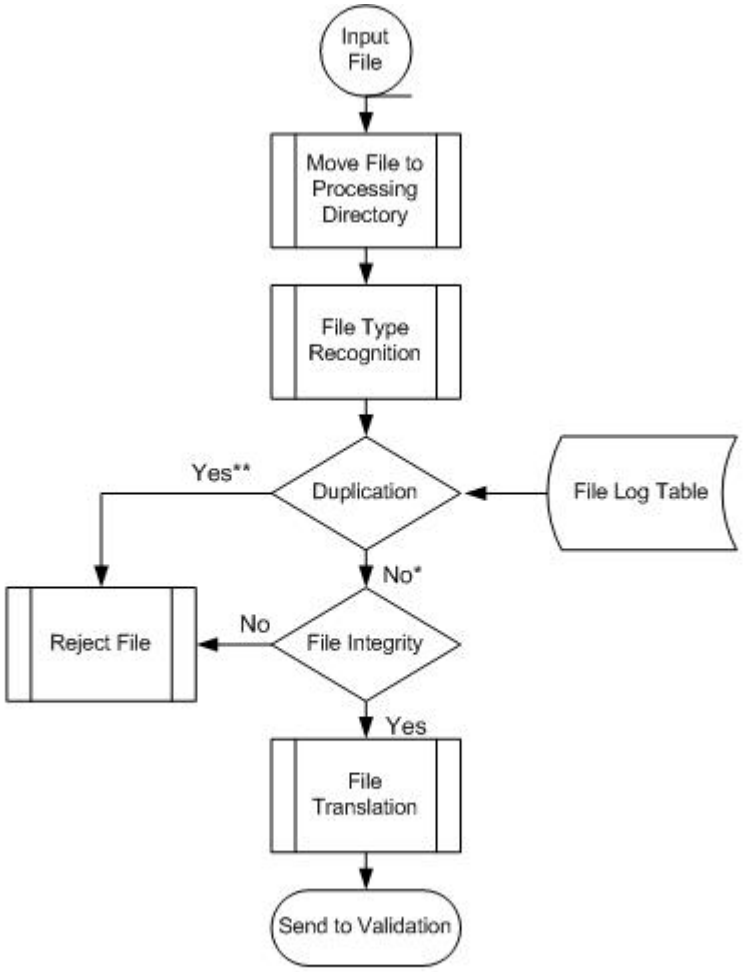
   Remittance Transactions are created from the remittance amounts received for invoices. These transactions are entered using the Remittance Entry screen in the GUI and are used to clear invoices after invoice amounts have been fully paid.
   See "Remittance Transactions" on page 145 for details.

10. Transaction processing terminates when all Input Files/batches from table are processed. As a stand-alone process, the Expected Transmission verifies whether the input files expected from a merchant have arrived at the configured date and time. If not, the process generates a notification.
    See "Expected Merchant's Transmission" on page 118 for details.

File Recognition

File recognition only validates the physical integrity of an incoming file and does not perform any validation based on business rules. Incoming files are picked up from the path given by the mas.fr_in_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file (for example, *$OSITE_ROOT/data/in*) according to the Scheduler's configured date, time window, and frequency.

The File Recognition processing flow is as follows:



* no duplication or files coming from the same data source or job name have the same totals
** same file name or different file name but the same data

Figure 3 File Recognition—Logical Processing Flow

1. File Logging to Processing Directory

   As soon as processing starts, File Recognitions moves the Input File into the processing directory given by the mas.fr_proc_dir (for example, *$OSITE_ROOT/data/processed*). This action prevents the File Recognition process from processing the same file twice.

2. File Type Recognition

   The File Recognition process compares the incoming file names with file name masks defined during institution setup. If the file name does not match a defined file name mask, the file is ignored and not picked up for processing.

   Files typically received by the File Recognition process are as follows:

   - clearing files—input files containing merchant transactions.
   - reversal files—clearing files that must be reversed either because errors occurred during transaction processing or because an acquirer wishes to reverse already processed transactions.
   - clearing and reversal files with no file duplication check—clearing and reversal files that are not going to be checked for file duplication. This option gives the flexibility of processing a file that has already been rejected during file duplication check.
   - adjustment files—clearing files that contain only adjustment transactions. These are transactions used to adjust payments and fees if the merchant was improperly charged, transactions have been improperly priced, or billing requisitions are generated. The adjustment input file is an automated alternative to generating adjustments manually through the GUI.
     See "Adjustments" on page 92 for more details.
   - remittance files—remittance files contain the details of payments received with invoice reference numbers. Processing of remittance files would apply the remittance to the corresponding invoice and update merchant account.

   To distinguish the file type, the system compares the file name with valid file name masks.
   See "File Name Masks" on page 170 for details on the File Name Mask.

   All clearing files follow the Input File Format, while adjustment files and remittance files follow Adjustment File Format and Remittance File Format, respectively.
   See "Input File" on page 172 for details on the input file format.
   See "Adjustment Input File" on page 177 for details on the adjustment input file format.
   See "Remittance Input File" on page 181 for details on the default remittance input file format. The user can use their own remittance file format.

3. File Duplication Check

   Input Files are checked against the File Log table to verify if they have not already been processed. Duplicate file checking eliminates the possibility of processing the same file twice.

File duplication check verifies for the following:

- Same file name—If this check fails, the new file is rejected.
- Different file name but the same data—If this check fails, the new file is rejected.
- Same totals from the same data source or job name—If this check fails, the new file is processed, and a warning is logged into the debug file.

The file type in the name of the input file can be used to skip the file duplication check. When you name the input file, select a file type that overrides this check. For example, a value of 21 overrides the file duplication check for clearing files. See "File Name Masks" on page 170 for file types currently supported by the system.

4. File Integrity Check

   To determine whether the structure of the file is healthy, the File Recognition process verifies several elements in the Input Files (like record size, record format, and presence of header/trailer records). The process also verifies whether the batch totals and the file total are balanced. If any discrepancies exist, the file is rejected.

   Rejected files are stored in the "reject" subdirectory defined by the mas.fr_reject_dir (for example, *$OSITE_ROOT/data/reject*), and the reason for rejection is contained in the appropriate debug file. If an incoming file is rejected, all transactions within the rejected file are ignored and the system makes no attempt to create suspended items or GL entries.

   Rejected files must be corrected and resubmitted to the system.

5. File Translation

   The content of the Input File is translated from the external message format (XMF) to the internal message format (IMF) and delivered to the Validation process as activity transactions, meaning that the translation process sends the records within the Input File one by one to the Transaction Validation process[1].

Direct Import from Database

Direct import only validates the physical integrity of an batches and does not perform any validation based on business rules. Batches are picked up from the mas_in_batch_log table and its corresponding transaction records are picked from in_mas_trans_log table.

1. The implementation of File Translation uses the Universal Agent, which is configured to receive the information from a file and to send it to a mailbox. This way, the records in the file are individually sent to the next process, the Transaction Validation. For details on the Universal Agent refer to *Universal Agent Guide*.

The Direct import from database processing flow is as follows:

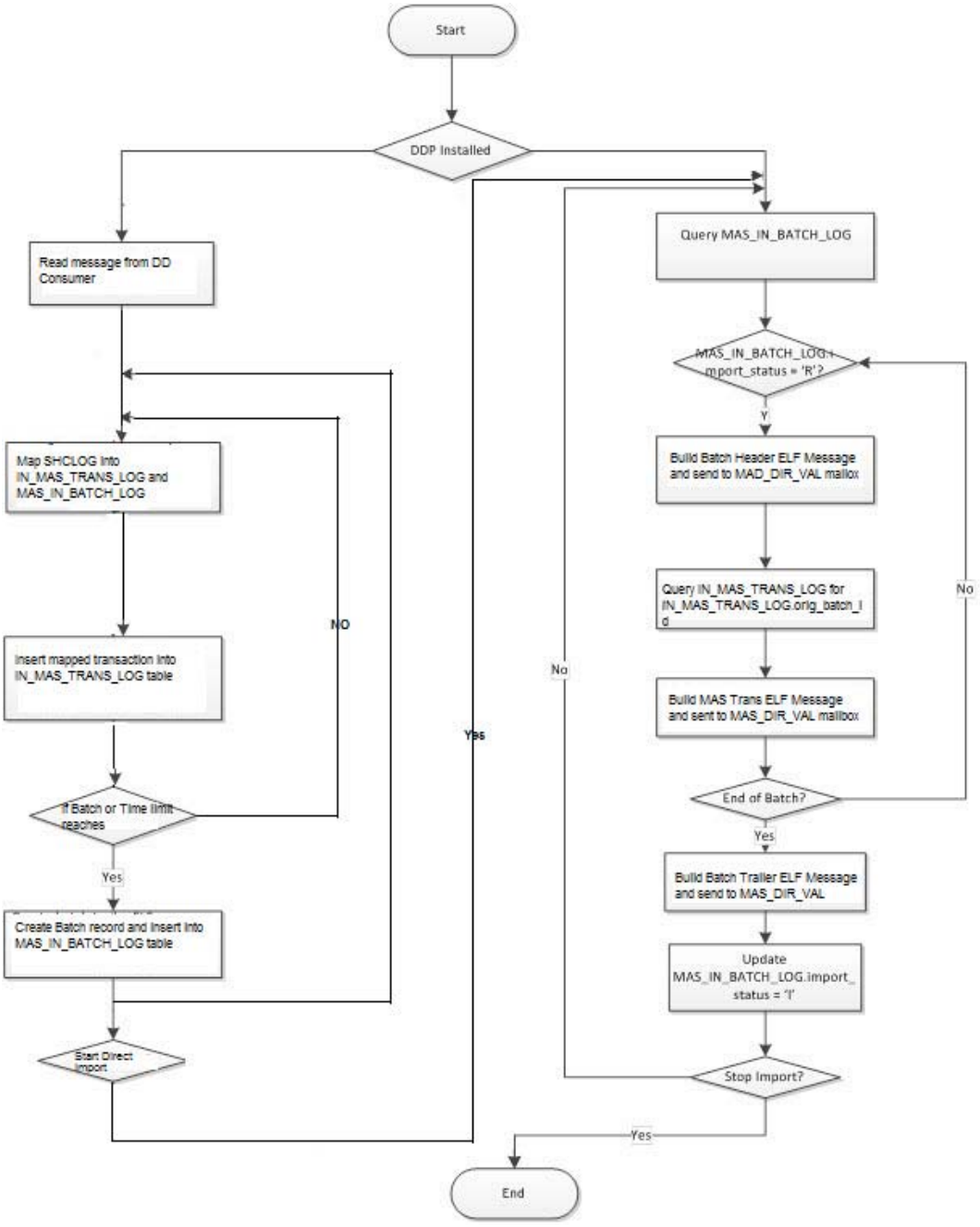


Figure 4 Direct Import from Database

1. Reverse Batch

   Batches that must be reversed either because errors occurred during transaction processing or because an acquirer wishes to reverse already processed transactions. Reversal batches that are not going to be checked for file duplication. This option gives the flexibility of processing a batch and that has already been rejected during batch duplication check.

2. Batch Duplication Check

   Batches imported from clearing are checked against the already process batches from in_mas_batch_log table to verify if they have not already been processed. Duplicate batch checking eliminates the possibility of processing the same batch twice. Batch duplication check verifies for the Same batch—If this check fails, the new batch is rejected.

Transaction Validation

The Validation process performs validations on individual transactions to determine if the transaction contains accepted values. Also, the message is enriched with several pieces of information that is required for further processing (See "Validation Enrichment" on page 209).

The Validation process uses the merchant setup information to verify whether an input record is valid. The following information is retrieved from a record:

- record type—provides the type or record (for example, file header, batch header, and detail record).
- transaction ID (TID)—provides the transaction type in this record (for example, authorization).
  See "Transaction Identifier (TID)" on page 206 for details.
- entity ID—provides the ID of the entity submitting the transactions.
- card scheme—provides the card scheme (for example, Visa or MasterCard).
- MAS code—provides the MAS code.
- MAS code downgrade—provides the MAS code downgrade.
- number of items and total amount for sales—provides the total number of sales transactions accumulated in this record and their total amount.
- number of items and total amount for credits—provides the total number of credit transactions accumulated in this record and their total amount.
- number of items and total amount for cashbacks—provides the total number of cashback transactions accumulated in this record and their total amount.
- invoice number—provides the invoice number of the remittance transaction.

- settlement bank account number—provides the settlement bank account number on which the remittance was applied.

NOTE:	The transaction records in the Input File contain either individual transactions or a summary of transactions of the same kind. The system performs the same processing in either case.

If a transaction passes all validation criteria, it is sent to the Pricing process. If not, the transaction is suspended for subsequent deletion or amendment and reprocessing. Suspended transactions must be resubmitted via the Resubmit Suspended Transactions process.

Based on the information supplied in the header of the Input File, several suspend levels exist: entire file, batch only, or transaction only. If the suspend level is file, the first transaction that does not pass validation suspends the entire file. If the suspend level is batch, an invalid transaction suspends only the parent batch, the rest of the batches are processed. If the suspend level is transaction, only the invalid transaction is suspended.

Processing Flow

The Validation processing flow is as follows (Figure 5 on page 59):

1. Card Scheme Validation

   The transaction's card scheme is validated by comparing the value to the configured values in the Card Scheme table. These values must be set by an administrator during installation and may include card schemes such as MasterCard and Visa.

   If the card scheme is not found, the transaction is suspended.

2. Invoice Number Validation

   The invoice number present in the remittance transaction is verified against the invoices in Invoice table.

3. Settlement Bank Account Number Validation

   The settlement bank account number present in the remittance transaction is verified against the settlement bank account configured in the bank account specified in the invoice.

4. Entity ID Validation

   The entity ID present in the transaction is verified against the values configured in the Acquiring Entity table.

   If the ID is not found, the transaction is suspended. If the ID is found, the transaction is enriched with the Settlement Plan ID and the Calendar ID obtained from the Acquiring Entity table.

   Based on the on-hold merchant indicator, the appropriate Settlement Plan ID is used. If the indicator is not set, the regular Settlement Plan is used. If the indicator is set, the On-hold Settlement Plan is used.

NOTE:	On_Hold_Settl_Plan is only applicable for activity transactions like sales, etc. The Processor will want to charge the merchant with the fee right away even if the transaction amount is on hold.

5. Hot Merchant Check

   Hot Merchant flag is used to enable specific processing, where a global Hot Merchant Settlement Plan is used to provide alternative routes for specific transactions for settlement. A set of transaction types (TIDs) is configured globally and used to direct those transactions to the global Hot Merchant Settlement Plan.

   Please note that this enables payment transactions to be blocked or re-directed but allow other transactions such as fees to be still processed/charged to merchant even if the Hot Flag is set. Once the hot flag is removed, processing resumes based on the regular merchant's Settlement Plan.

The process verifies the Hot Merchant flag for the acquiring entity. If the flag is not set, the regular Settlement Plan is used. If the flag is set, the transaction is enriched with the ID of the Hot Merchant Settlement Plan retrieved from the Field Value Control table.

This plan is common to all merchant that have been declared “hot,” and is used to redirect transactions to a specified account. Together with reserve payment, stop payment, and stop fee, the hot merchant flag manages risks.

6. Settlement Plan Validation

   The transactions’ Institution ID, Settlement Plan ID, Transaction ID and Card Scheme are compared against configured values in the Settlement Plan TID table.

   If the validation fails, the transaction is suspended. If the validation is successful, the transaction is enriched with the following information from the Settlement Plan TID table:

   a. GL Account To.

   b. GL Account From.

   c. Posting Entity ID. Note that if the Posting Entity ID is null in the Settlement Plan TID table, the Entity ID is used as the Posting Entity ID.

   d. Payment Term.

   e. Settlement Flag.

   f. Date To Settle. If the settlement frequency is List and the system has used up all dates in the list, the transaction is suspended (logged in the MAS_TRANS_LOG table). The suspended reason code is given by the RC_NV_SETTLE_DATE parameter in the Field Value Control table.

   g. TID Reserve.

NOTE:	If the On-hold Merchant Indicator is set, this information is retrieved from the On-hold Settlement Plan; in other words, a different posting entity may be configured to handle these special merchants.

   h. Posting Entity ID.

NOTE:	If the Posting Entity ID is null in the Settlement Plan TID table, the Def_posting_Entity in Acq Entity table is used as Posting Entity ID. If Def_Posting_Entity is null, then the incoming Entity ID is used as the Posting Entity ID.

SETTL_PLAN_TID.posting_entity_id	ACQ_Entity.def_posting_entity	How Applied
not null not null	null not null	Use posting_entity id to post. Applies when settlement plan is exclusive
null	not null	Use def_posting_entity to post. Applies when settlement plan is shared.
null	null	Use sending Entity ID to post.

7. Posting Entity Validation

   Based on the Posting Entity ID from the previous step, the system searches the Account Posting Plan for the account to post the particular activity transaction.

   If the value is defined, the transaction is enriched with the Entity Account ID from the Account Posting Plan table; otherwise, the transaction is suspended.

8. Transaction ID in Fee Package

   Each type of transaction processed is identified by a unique code called a Transaction ID (TID), which is stored in the TID table. To validate the TID, the process first looks into the TID table, and any transaction with an unrecognized TID is suspended. If the TID is recognized, the process verifies whether any Fee Package contains the TID as defined in the Fee Package TID table.

9. MAS Code in Fee Package

   MAS Codes are associated with one or more Fee Packages and are used to search for the applicable Fee Package for the merchant entity. If the settlement TID is in the list of wild TIDs defined in the Field Value Control table, the selection of fees includes wildcard matching on the MAS code against the values defined by the WILD_MAS_LIST field in the Field Value Control table. (See "Wildcard MAS Codes" on page 215 for details on wildcard matching.)

   If an entity's Fee Packages do not contain a corresponding MAS Code, the system looks at the entity's parent node in the hierarchy for an appropriate Fee Package and MAS Code. This hierarchical search for information, referred to as roll-up processing, continues until a matching MAS Code is found.

   Successful searches yield an applicable Fee Package. The transaction is enriched with the Fee Package defined in the Acquirer Entity table. Unsuccessful searches result in a query of the Non-qual indicator.

NOTE:	This applies only to those TIDs that have been found in the TID table. When a TID is not found, the input file is rejected.

10. Non-qual Indicator Check

    The Non Qualification indicator (Non-qual) determines the transaction should be enriched with the expected MAS code when an incoming transaction contains a MAS Code other than what is expected for the Entity (merchant). This indicator is ignored if the transaction contains an expected MAS Code.

    For unexpected MAS Codes, further processing depends on the setup value of this indicator. A Y value causes the system to enrich the transaction with the expected MAS Code defined in the Fee Package MAS Code table, while an N value suspends the transaction.

11. MAS Code Downgrade Validation

    The transaction is verified for the presence of a value in the MAS Code Downgrade field.

    If the MAS Code Downgrade is not present, the processing in this step is ignored.

    If the MAS Code Downgrade is present, the entity's Fee Packages are searched for the corresponding MAS Code Downgrade. If a matching code is found, the next step is performed. If a matching code is not found, the Non-qual Indicator is queried. A Y value continues processing, while an N value suspends the transaction.

NOTE:	This step is optional because it applies only if the transaction has to be re-classified. See "MAS Code Upgrade/Downgrade Pricing" on page 73 for details on re-classifying transactions.

12. Batch Summary Creation

    The Batch Summary accumulates information used for batch balancing, which ensures that transactions are not suspended during Validation, Pricing, or Posting without informing the user. The information is logged into the event logging table.

    The information used for batch summary is saved in the BATCH_SUMMARY table and includes the total number of transactions in the batch (Begin Record Count field); and the batch total (Batch Amount field), as the total of the amounts for all transactions in the batch.

    To verify whether a batch is balanced, similar values are accumulated in the Posting process and then compared with the values from Validation.

13. Valid transactions are sent to the Pricing process for further processing.

    Once all records in a file or batches from database are processed, the Input File is moved to a "done" subdirectory, given by the *mas.fr_done_dir* parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file (for example, *$OSITE_ROOT/data/done*), and a notification that the file has been successfully processed is saved to the appropriate log file.

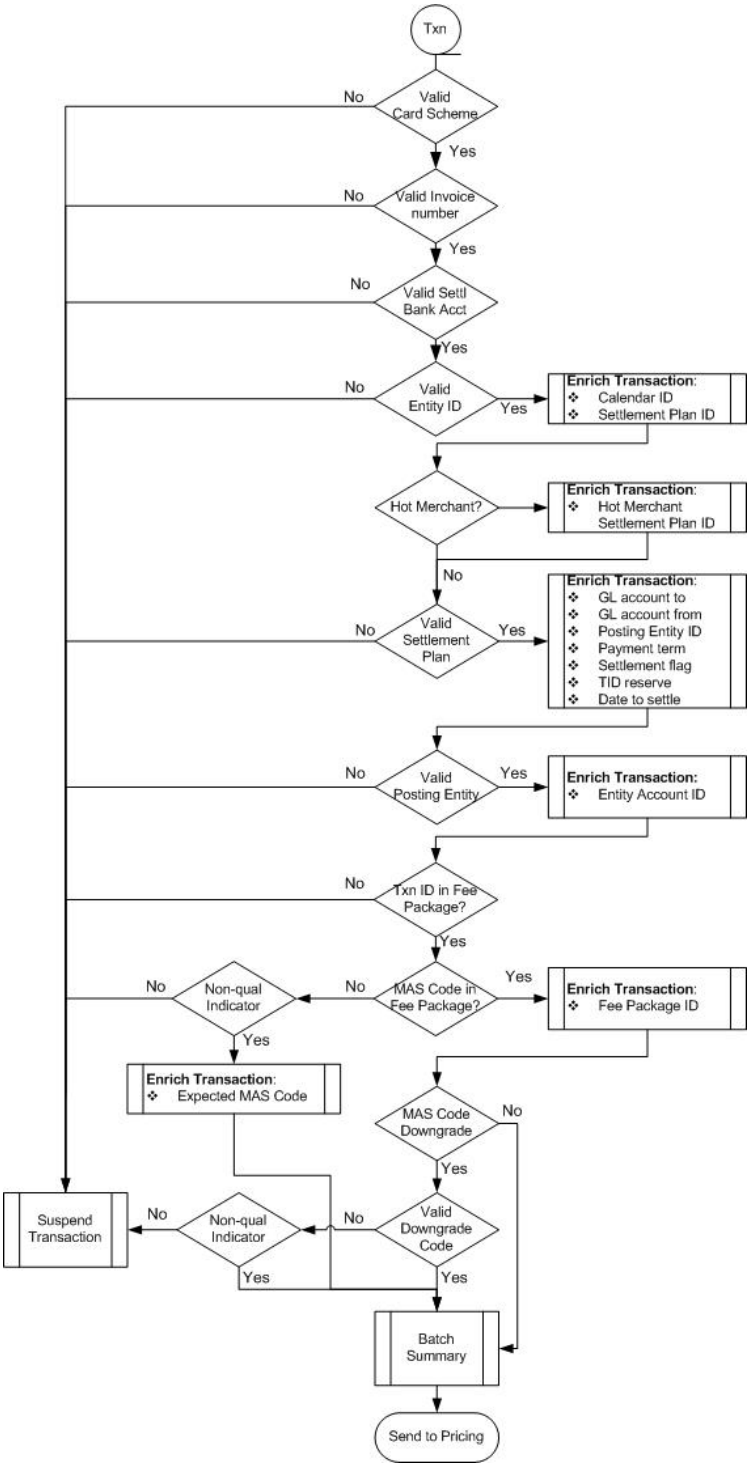


Figure 5 Transaction Validation—Logical Processing Flow

Resubmit Suspended Transactions

If a transaction passes all validation criteria, it is sent to Pricing; otherwise, the transaction is suspended for subsequent deletion or amendment and reprocessing. Based on the information supplied in the header of the Input File, several suspend levels exist: entire file, batch only, or transaction only.

Suspended transactions must be resubmitted via the Resubmit Suspended Transactions process, which is a user-invoked process. Suspended transactions are stored in the MAS Transaction Suspension table (MAS_TRANS_SUSPEND).

The process handles suspended transaction according to the original suspend level provided in the incoming file (and stored in the MAS_FILE_LOG table). Although transactions can be resubmitted by batch, by file, or all, this scheduled task must conform to the original suspend level; otherwise, the *IST/MAS* server does not perform the operation, and the offending command is logged. For example, if the original suspend level is file and the input file contains three batches, you cannot resubmit only one batch.

The suspended transactions are not queued for Pricing, but remain in storage for further intervention. An operator must first manually edit the transactions, or the system setup, in accordance with the suspend reason code (via appropriate GUI form), and then resubmit them to the Validation process for regular transaction processing. For example, if a transaction has been suspended because a MAS code has not been defined, the particular code must be added to the system setup and the transaction resubmitted to Validation.

Suspend reason codes are configurable in the Field Value Control table (done by the administrator) and may include values as invalid settlement plan, invalid fee package, or invalid MAS code.

For the process to obtain the correct GL entry to update, an operator must define two default Suspend Settlement Plans for each TID used by an acquiring institution.

- SUSPEND_SP_IN—used for handling suspended transactions that have been sent to the system for the first time.
- SUSPEND_SP_OUT—used for handling resubmitted suspended transactions.

Suspended transactions that cannot be resubmitted are cleared from the system via the write-off mechanism.
See "Write Off Suspended Transactions" on page 61.

Write Off Suspended Transactions

Write Off Suspended Transactions is a user-invoked process used to clear from the system suspended transactions that cannot be resubmitted.

For example, a transaction may come into the system with the wrong TID. Since the MAS Transaction Suspension table cannot be updated when the TID is not correct (TID is defined in the TID table, but does not exist in the Fee Package), the transaction must be cleared from the system.

To write it off, a suspended transaction must be marked via GUI, and a task must be scheduled via Scheduler. The Write Off Suspended Transactions process updates the appropriate tables (Transaction Log, Transaction Suspended, and Batch Summary) and the GL entry. The written off suspended transaction is populated with the current GL date.

For the process to obtain the correct GL entry to update, an operator must define the SUSPEND_SP_W/O (default Suspend Settlement Plan) for each TID used by an acquiring institution.

Transaction Pricing

After Validation, the activity transaction, enriched with the Fee Package ID and other elements, is delivered to Pricing, which calculates the fees to be charged for processing and generates associated fee transactions.

In *IST/MAS*, fees are generally divided in the following two broad categories:

1. Activity fees—generated by processing incoming activity transaction files or batches from database (authorization files, clearing files, etc.).

2. Non-activity fees—generated by the system at specified dates (recurring fee, ACH fee, etc.). Please note that the system-generated fees may also be related to activity transactions (such as penalty fees if activity transactions fail to reach the threshold volume).
   See "Non-activity Fees" on page 93 for details.

   The Pricing flow is as follows (Figure 6 on page 63):

3. Settlement Date Determination

   For each activity transaction from the Input File or batches from database, the process must establish a settlement date, that is, when merchant entities must pay their fee to the service provider and when payments are posted to the merchant's accounts. Calculations are based on the merchant's cutoff time, next settlement date (determined at the end of each day based on frequency), delay factor, and days to hold.

   The calculated settlement date is adjusted according to the merchant's Calendar. See "Settlement Date" on page 64 for details on settlement date calculations.

4. Pricing Processing

   Based on various flags (reserve payment, debit pricing, non-qual pricing, ISO billing, and MAS Code Upgrade/Downgrade), appropriate processing is performed. Note that various combinations are possible. For example, non-qual pricing and ISO billing pricing may be required. If no flag is set, regular pricing is performed.

   a. Reserve Payment—Reserve payment applies when a merchant must have a reserve account. The payment to the merchant is split between the regular DDA account and the reserve account.
   See "Reserve Payment" on page 68 for details.

   b. Three-Tier Pricing—Simplifies pricing and statement reporting for lower volume merchants by categorizing the interchange qualification levels into three groups: Qualified, Mid-Qualified and Standard.
   See "Three-Tier Pricing" on page 74 for details.

   c. Debit Pricing—Debit Pricing is applied to transactions from a debit network.
   See "Debit Pricing" on page 70 for details.

   d. Non-qual Pricing—Non-qualification pricing applies to transactions of merchants who do not have expected MAS Codes and associated Fee Packages defined, but who are configured with a Non-qual Indicator in the Acquiring Entity table.
   See "Non-Qual Pricing" on page 71 for details.

   e. ISO Billing—ISO billing is applied when the revenue is shared with parent entities. The process, which is repeated up the entity hierarchy structure, calculates the parent's fee and generates a fee transaction.
   See "ISO Billing" on page 72 for details.

   f. MAS Code Upgrade/Downgrade—When a merchant submits an incorrectly qualified transaction, the acquirer recognizes the error, regenerates a new transaction with a revised MAS Code, and submits it for reprocessing.
   See "MAS Code Upgrade/Downgrade Pricing" on page 73 for details.

   g. Regular Pricing—Regular pricing is performed when no specific pricing is required, that is, no special processing flags are set.
   See "Regular Pricing" on page 66 for details.

   h. Tax Calculations—Additional taxable transactions may be sent to Pricing by the Manual Adjustments process, like Billing Requisitions for equipment and supplies sales. These transactions require tax calculations to know how much to charge merchants for the taxes associated with the sales amounts.
   See "Tax Calculations" on page 75 for details.

The following figure presents the Pricing processing flow:

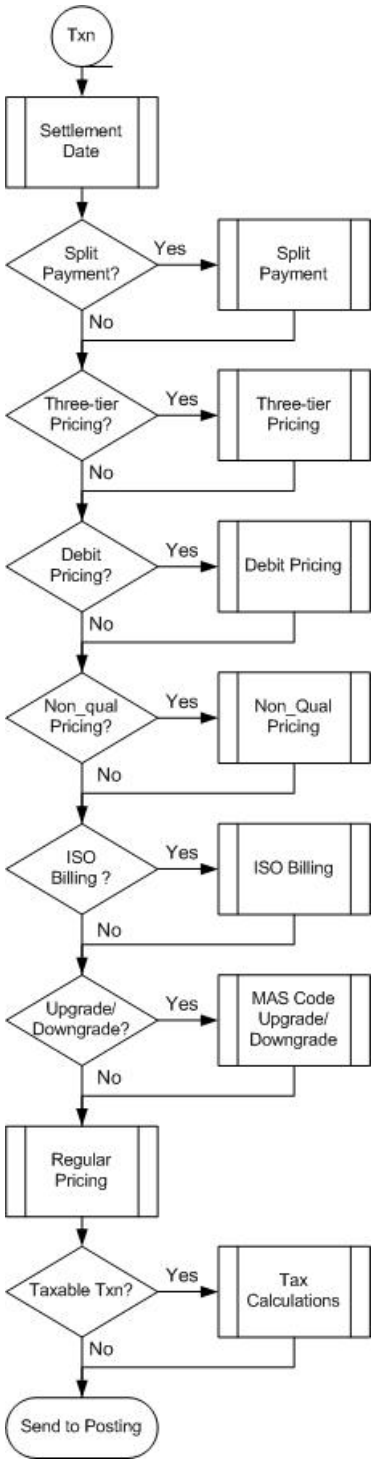


Figure 6 Transaction Pricing—Logical Processing Flow

Although not directly part of the settlement flow, several processes are used to prepare information required by Pricing. These processes are as follows:

- Based on the Item Count Plan settings, the system accumulates transaction amounts (See "Item Accumulation" on page 77). These accumulated totals are used in the Item Count Plan calculation done for the Rate and Fee Adjustment or Non-activity Fees (See "Item Count Plan Calculations" on page 77).
- Pricing is based on the fees and fee groups set up via GUI during general system setup. These fees are made effective either via Fee Group Rate Update (See "Fee Group Rate Update" on page 78) or Future Effective Rate Update (See "Future Effective Rate Update" on page 78).
- Fees are updated via Rate and Fee Adjustment to reflect the merchant's performance over a period of time.
  See "Rate and Fee Adjustment" on page 79 for details.
- Next Settlement Date is used to obtain the next settlement date used by Pricing to determine the transaction's settlement date.
  See "Next Settlement Date" on page 84 for details.
- Merchant Start/End Date is used to define the date of the entity's first clearing transaction sent to the system. This information is required for inquiry purposes.
  See "Merchant Start/End Date" on page 86 for details.

Settlement Date

The settlement date is a target date towards which amounts and counts are accumulated. It is used by the system (together with payment effective date for instance) to determine when the merchant entity must pay its fees to the service provider and when payments are posted to the merchant's accounts.

The settlement date is determined by adjusting an initial settlement date (either the Next Settlement Date or the activity date) for cut-off time, days to hold, and delay factors. When the Settlement date is determined, the following steps are performed:

1. Selection of Initial Settlement Date

   The settlement date is determined by using the Next Settlement Date field in the Settlement Plan TID table. If the Next Settlement Date is not populated (i.e., is zero), the system uses the date in the activity transaction instead.

   For the purpose of this flow, let us assume that the date of the activity transaction is used.

2. Adjustment for Cut-off Time (Adjusted Activity Date)

   The activity date in the incoming transaction is used as the basis for settlement date calculation. If the activity date and time are not populated, the system sets the activity date and time to the current GL date.

Depending on the merchant's cut-off time, which is defined in the Calendar as normal processing hours, the processing date is adjusted if the activity transaction falls after the merchant's cutoff time. In this case, *IST/MAS* adds one day to the activity date to move the transaction to the following day.

The cutoff-time adjustment is based on verifications done during transaction validation. When the transaction is processed in Validation, the CUTOFF_FLAG is set if the transaction came after the normal processing hours defined for the merchant. The flag is used by Transaction Pricing during settlement date determination to adjust the activity date.

3. Adjustment of Payments for Days to Hold (Re-adjusted Activity Date)

   For payments, the adjusted activity date determined in the previous steps is re-adjusted by adding the number of days set up in the Days to Hold field. This global adjustment represents the number of days to delay a payment for an acquiring entity, and therefore affects all accounts defined for the entity.

NOTE:	This step is performed only for payments. It is skipped for fees.

4. Adjustment for Delay Factors (Date to Settle)

   The formula used to determine the date to settle is slightly different for payments and fees. In particular, the date to settle is calculated based on the adjusted activity date for fees, and on the re-adjusted activity date for payments.

   For fees, the settlement date is determined by adding the appropriate Delay Factor to the adjusted activity date. If the adjusted activity date is a holiday, the system also adds the Holiday Delay Factor.

```
date_to_settle (fees) = adjusted_activity_date
    + delay_factor* (counting business days)
    + holiday_delay_factor (if adjusted_activity_date is
holiday)
```

   For payments, the settlement date is determined by adding the appropriate Delay Factor to the re-adjusted activity date. If the re-adjusted activity date is a holiday, the system also adds the Holiday Delay Factor.

```
date_to_settle (fees) = re-adjusted_activity_date
    + delay_factor* (counting business days)
    + holiday_delay_factor (if re-adjusted_activity_date is
      holiday)
```

   The delay factor added depends on the activity date. If the activity date is a Tuesday, the value of the DELAY _FACTOR2 field (i.e., the delay factor for Tuesday is added).

Also, Delay Factors count business days rather than calendar days to determine the settlement date. For example, if the transaction comes Friday and the delay factor for Friday is 1, the settlement date is Monday, since Saturday and Sunday are not business days. Note, however, that this is a generic example. Business days are as specified by the calendar setup.

The settlement date calculation is illustrated in the following examples.

Example—"Early" with activity on a holiday

Let us assume that the delay factors for each day of the week are 1-1-1-1-1-1-1, Fri-Sat-Sun are to be paid on Monday, and the holiday_delay_factor is 0.

For an activity transaction received on Monday, a holiday, before cut-off, the date to settle is as follows:

```
date_to_settle = Mon + 1 + 0 = Tue
```

For an activity transaction received on Monday after the cut-off, the date to settle is as follows:

```
date_to_settle = Tue + 1  = Wed
```

Example—"Regular (default)" with activity on a holiday

Let us assume that the delay factors for each day of the week are 2-1-1-1-1-1-2, Fri-Sat-Sun are to be paid on Tuesday, and the holiday_delay_factor is 1.

For an activity transaction received on Monday, a holiday, before cut-off, the date to settle is as follows:

```
date_to_settle = Mon + 1 + 1 = Wed
```

For an activity transaction received on Monday after the cut-off, the date to settle is as follows:

```
date_to_settle = Tue + 1  = Wed
```

Regular Pricing

NOTE:	Regular Pricing is described first since it is the base of all other pricing methods. Also, note that for your convenience, the example presented includes the steps performed by the system before the actual pricing is done.

Regular pricing is performed when no specific pricing is required, like reserve payment or debit pricing.

To illustrate the processing involved in regular pricing and how fees are calculated, the following example has been considered.

1. A transaction with the TID 304 and the MAS code 1247 has been received for the merchant M1.

MasCd	TID	am	cm	am1	cm1	am2	cm2
1247	305	300	10	100	2	50	1

where -

- *am* is the sales amount and *cm* the sales count
- *am1* is the credit amount and *cm1* the credit count
- *am2* is the cashback amount and *cm2* the cashback count

2. The transaction processing starts in Validation, when the process selects the appropriate Fee Package for the merchant M1 and determines the merchant's Settlement Plan.

   For easier maintenance and application, activity fees are configured in packages. Activity fee packages are set up in the ENTITY_FEE_PKG table, which links acquiring entities to their respective activity fee packages. This organization allows for common fee packages to be configured and applied as groups. Also, multiple acquiring entities can share common fee packages via a pointer, eliminating duplication during activity fee setup.

   Although multiple fee packages exist, only one combination of charge method, TID, and MAS code is allowed. When duplication exists, only one fee is charged, and the active combination is the one in the most current activity fee package.

3. This information is passed on to Pricing, which enriches the incoming transaction with the appropriate fields from the Settlement Plan, and calculates the settlement date.

4. Based on the Fee Package, incoming TID, and MAS code, Pricing selects all current fees charged for the transaction.

5. If the Fee Package is 501, the following information may be retrieved from the MAS Fees table for TID 305 and MAS code 1247:

Fee	%	perItem	am (sales)	cm (sales)	am1 (credit)	cm1 (credit)	am2	cm2
401	0.2	0.05	+	+	-	+		
402	1.5	0.20	+	+		+		
403	0	0.26		+	-	+		
404	0.2	0.08	+	+	+	+		

6. For each fee, Pricing calculates the base transaction amount and count based on the calculation methods (+, -, none) set up for the amounts and counts.

   Based on the settings in the Fee Package 501, the base amounts are calculated as follows:

```
401   sales - credit   300 - 100 = 200
402   sales only                   300
403   credit only                  100
404   sales + credit   300 + 100 = 400
```

   Similarly, the total counts are calculated.

```
401   sales + credit    10 +   2 =  12
```

7. For each fee, Pricing calculates the fee charges by applying the percent rate and rate per item to the base transaction amount and count.

   If percent rate and rate per item are both set to zero, fee transactions are not generated.

   Based on the settings in the Fee Package 501, the charges for Fee 401 are calculated as follows.

```
401   0.2%  x 200  = 0.4   (percent rate)
      0.05  x  12  = 0.6   (rate per item)
```

   Since Fee 401 has both percent rate and per item rate set, the total fee charged is calculated by adding the percent rate charge and the rate per item charge (0.4 + 0.6).

8. For each fee, a new transaction is created.

   Based on the setting s in the Fee Package 501, four new transactions are generated with the TID 401, 402, 403, and 404, respectively, and with the fee amounts calculated in the previous step.

Reserve Payment

IST/MAS offers the ability to split merchant payments between accounts at the payment level. A reserve payment is a partial payment to two accounts. The first payment is directed to the entity's account and the remainder is sent to a reserve account.

NOTE:	A reserve payment is not a pricing method (i.e., not used to calculate amounts), but it is used during posting to redirect already calculated payments.

The reserve payment processing is as follows:

1. Some merchants are set up with a "Reserve-to-be-met" amount, a reserve percentage, and a reserve fund status (initially R).

reserve fund status	may contain three different values, • F =fulfilled, • R =reserve fund, • blank =non-split
reserve to be met	dollar amount used to compare to posted amount in the entity account
reserve percentage	percent of the amount to be put into a reserve fund

2. When a transaction comes in, the reserve fund status is verified. If it is R, the reserve payment is performed.

   Note that the incoming transaction subject to reserve processing needs to be explicitly configured in the Settlement Plan and assigned a Reserve TID, which is used to generate the appropriate reserve transaction for tracking, reporting and settlement to configured Entity Accounts and GL Accounts.

3. For each payment transaction, the “Reserve-to-be-met” amount is evaluated. If the merchant has an amount greater than zero, the reserve amount is calculated by applying the reserve percentage to the merchant’s net payment amount.

   From the original transaction, two new transactions are created: one transaction debits the reserve portion from the account defined by the TO_ACCT field and one transaction credits the reserve account. Consequently, the merchant’s payment is reduced by the reserve amount, so the payment split is between the reserve account and the remaining merchant payment.

   Use the Reserve TID in the Settlement Plan TID table based on the incoming TID to define the credit reserve TID, and then use the Reserve TID in the Settlement Plan TID table based on the credit reserve TID to define the corresponding debit reserve TID.

   For example, if the original transaction (TID 305) is posted to account A, account B is the reserve account, and $50 is the amount to transfer to the reserve account, the following transactions are created:

   - A transaction of $50 to debit account A (TID 705 provided by the TID_RESERVE field in the Settlement Plan of 305). For this purpose, the original transaction (TID 305) has 705 defined as a reserve TID, and the Settlement Plan for TID 705 defines account A as the account to debit.
   - A transaction of $50 to credit account B (TID 505 provided by the TID_RESERVE field in the Settlement Plan of 705). For this purpose, the generated transaction (TID 705) has 505 defined as a reserve TID, and the Settlement Plan for TID 505 defines account B as the account to credit.

4. The reserve payment is applied until the “Reserve-to-be-met” reaches the established amount.

   The payment stops when the accumulated amount in the defined Posting Account (Reserve Account) matches the reserve to be met threshold. This implies that each entity needs to be configured to post reserved funds to one Entity Account dedicated for this purpose. The appropriate DDA to post these funds to is also specified in this Entity Account.

5. The necessary GL entry to reflect a payment split is also generated.

   To provide for the proper GL Account posting and reporting, from the original transaction, two new transactions are created whenever the reserve percentage is applied to an incoming merchant transaction:

   - A partial reversal of the original transaction is applied to the original GL account.
   - A reserve TID transaction is posted to the appropriate GL account to reflect the reserve payment. The GL account impacted by the Reserve TID is configured in the Settlement Plan under Reserve TID transaction type.

   By using a dedicated transaction type for the Reserve TID, reserve transactions can also be settled as configured in the Settlement Plan.

Debit Pricing

NOTE:	It is assumed that the input file contains summarized transactions since all calculations in debit pricing are based on just one summarized transaction that contains totals of sales, credit, and cashback transactions and their volume.

Debit pricing is performed if there are some values found in PER_TRANS_MIN and/or PER_TRANS_MAX columns of the current fee record in the MAS_FEES table, and is applied to transactions coming from a debit network. In other words, applied fees depend on the total transaction amount and volume of transactions.

1. Fee Calculation

   Fees are calculated following regular pricing methods.

2. Average Fee Calculation

   The average fee is calculated by adding together the per item fee and the percent fee and dividing the result by the total number of items.

3. Average Fee Adjustment

   The average fee is compared with PER_TRANS_MIN and PER_TRANS_MAX, if they exist, and the average fee is accordingly adjusted.

The following table shows how the fee is calculated for different settings.

Per Trans Max	Per Trans Min	Average Fee Calculated	Final Fee Calculation
Set	Set	Between Max and Min	Fee = original calculation
Set	Set	Greater than Max	Fee = actual count * Per Trans Max
Set	Set	Less than Min	Fee = actual count * Per Trans Min
Set	Not Set	Less than or equal to Max	Fee = original calculation
Set	Not Set	Greater than Max	Fee = actual count * Per Trans Max
Not Set	Set	Less than MIN	Fee = actual count * Per Trans Min
Not Set	Set	Greater than or equal to MIN	Fee = original calculation

Non-Qual Pricing

When a merchant is set up, several expected MAS codes (one per card scheme) and several excluded MAS codes are defined for a merchant. The expected MAS codes represent usual merchant processing for which *IST/MAS* performs normal pricing. The excluded MAS codes represent unusual merchant processing for which *IST/MAS* still performs normal pricing.

However, the merchant may send transactions with a MAS code that is neither an expected MAS code nor an excluded MAS code. Non-qualification pricing applies to these transactions if the merchant is configured with a Non-qual Indicator set to Y in the Acquiring Entity table. If the indicator is set to N, the transaction is suspended.

Total non-qualification charges are calculated as follows:

1. Fees are calculated using default rates.
2. Additional non-qualification charges are calculated using a non-qual rate that is obtained with the following formula:

   Non-qual rate = default rate for incoming MAS code (actual) – default rate for expected MAS code (expected)

where -

expected	is the expected MAS code defined in the merchant's Fee Package.
actual	is the MAS code of the incoming transaction (which is not an expected MAS code or excluded MAS code).

a. If the calculation is in favor of the merchant, the additional credit is applied only if that merchant is flagged as allowing credits (ALW_NON_QUAL_UPGRD set to Y in the ACQ_ENTITY table). Nevertheless, a surcharge can also be applied to the merchant's transaction. Rates for this surcharge are defined by the non-qual surcharge rate in the merchant's Fee Package.

b. If the calculation results in a charge to the merchant, the additional charge is applied. A surcharge can be applied to the merchant's transaction, and the rates for this surcharge are defined by the non-qual surcharge rate in the merchant's Fee Package.

3. The calculated amount is credited to or debited from merchant's account via a transaction with the TID defined in the MAS Fees table. The amount of this transaction represents the additional charge/credit and the surcharge. In other words, the surcharge decreases the credit amount or increases the debit amount.

ISO Billing

In addition to billing and paying the merchant using the merchant hierarchy structure, *IST/MAS* also supports revenue sharing across the merchant hierarchy via ISO Billing.

For example, Independent Sales Organizations (ISO) may act as agents for smaller merchants, other ISOs, and correspondent affiliate banks. In this role, they negotiate with the Processor (owner of *IST/MAS*) to determine the rates that should be charged for services. Since these organizations find business for the Processor, they are entitled to share with the Processor the revenue obtained from collecting fees.

The revenue is propagated across various entities involved, but no restrictions exist on how to do it. In addition to where to send a bill and where to send the payments, *IST/MAS* must also know where to send the revenue collected.

During Pricing, fees are applied to the original transaction based on the entity fee package. An activity transaction can have one or more fee transactions generated as part of the pricing process. Each of the generated fees is enriched with the settlement plan information.

If it is determined from the Settlement Plan that the current fee has its revenue shared with a parent entity, the ISO billing is applied. The process calculates the parent's share from the fee's revenue and generates a 'profit' fee transaction. The process is repeated up the entity hierarchy structure. Fees are posted to the merchant entity and the calculated 'profit' is posted to the parent entities.

ISO billing is applied as follows:

1. Pricing uses the Parent ISO TID (used as the Activity TID), the MAS Code, and Card Scheme from the original transaction to locate the fee package used by the parent entity. The parent entity is now considered the current entity.
2. The fee package is used to generate profit sharing fee transactions for the parent entity (current entity) in the hierarchy.
3. If the Parent ISO TID field of the current entity's Settlement Plan for the profit sharing TID is populated, the same process is repeated up the hierarchy.
4. The process stops when the Parent ISO TID of the Settlement Plan is empty.

   ISO billing can be combined with Regular Pricing, MAS Code Upgrade/downgrade Pricing, and Non-Qual Pricing.

   During ISO billing processing, Wild MAS code matching is used.

   See "ISO Billing" on page 243 for a detailed example of ISO Billing.

MAS Code Upgrade/Downgrade Pricing

The Upgrade/downgrade pricing occurs when *IST/MAS* has already processed a transaction that the acquirer recognizes as been incorrectly qualified. For *IST/MAS* to reverse the processing of the incorrectly qualified transaction and reprocess the transaction with the correct MAS code, the acquirer must resubmit the transaction with two MAS codes, the original MAS code and the revised MAS code.

The resubmitted transaction will have in the TID field a re-classified TID, not the original TID. The re-classified TID is configured in the system, and is used by the system to retrieve the original transaction TID.

IST/MAS first retrieves the original transaction rate based on the original MAS Code. This rate is used to reverse originally charged amounts. The new MAS Code is used to obtain the new rate and calculate the new charge amount(s).

To calculate the new fee, the fee associated with the original MAS code is subtracted from the fee associated to the revised MAS code:

1. If the difference is positive, two fee transactions are generated for that TID. One fee transaction reverses the originally charged fee and the other fee transaction applies the new fee.

   If there were multiple fees charged to this transaction, all of them would be reversed and new ones generated based on the new rates. The process also performs GL adjustments by posting reversals of old amounts and posting of new amounts to GL accounts specified for the fee transactions in question.
2. If the difference is negative and the merchant setup does not allow upgrades (ALW_RECLASS_UPGRD in ACQ_ENTITY set to N), no additional processing is done, meaning that the original fee is still charged.

3. If the difference is negative and the merchant setup allows upgrades (ALW_RECLASS_UPGRD in ACQ_ENTITY set to Y), two fee transactions are generated for that TID. One fee transaction reverses the originally charged fee and the other fee transaction applies the new fee.

   If there were multiple fees charged to this transaction, all of them would be reversed and new ones generated based on the new rates. The process also performs GL adjustments by posting reversals of old amounts and posting of new amounts to GL accounts specified for the fee transactions in question.

4. If the difference is zero, no transactions are created.

5. Pricing saves in the Outbounder file the associated activity transaction with net zero effect to the merchant's settlement (i.e., the settlement flag is set to N).

   See "Transaction Pricing" on page 61 for a re-classification sample.

Three-Tier Pricing

Many processors are using a Three-tier Pricing Model to simplify the pricing processing, the statement reporting, the fees setup, and the future rates maintenance for lower volume merchants.

The Three-tier Pricing Model categorizes the multiple interchange qualification levels into three groups: Qualified, Mid-Qualified, and Standard. Each group uses one rate, which is applied to all interchange programs within that group. The interchange programs within each set of qualification level categories may vary from merchant to merchant, based on contract arrangements.

For example, a Qualified group may consist of the following interchange programs, and the charged for any of these programs is the same (e.g., 2.175%):

- CPS Retail
- Supermarket Incentive Program
- Merit III
- Corporate Face-to-Face

The MAS code group IDs are defined in the MAS_CODE_GRP table and are maintained in the entity fee package (ENTITY_FEE_PKG) to have control over different fee packages at different points in time.

If a MAS_CODE_GRP_ID is assigned to an entity fee package, the corresponding MAS code is obtained by checking the original qualification MAS code.

To initiate the Three-tier Pricing processing, set up MAS code groups and assign them to entity fee packages via GUI. If a MAS Code group ID is set in the entity fee package, Three-tier Pricing is performed for that entity.

Tax Calculations

IST/MAS must consider the tax implications of transactions processed in the system. For example, billing requisitions for equipment and supplies are sales transactions generated for the equipment and supplies the processor sells to acquiring entities. For these sales transactions, applicable tax fees must be calculated.

However, *IST/MAS* does not calculate the tax amounts. It only generates the applicable tax transactions and uses a third-party tax system[1] to obtain the tax amounts.

To understand how tax amounts are obtained, the processing of the equipment and supplies transactions can be used.

Equipment and supply transactions, which are manually created via GUI or enter the system via the incoming file[2], can be either settled or non-settled based on the existence of an amount in the transaction.

Non-settled transactions are used when there is no amount populated in the transaction, and only the number of sold items is present. (Even if the amount is present, it is ignored and recalculated in Pricing). To calculate the amount to be charged to the merchant, Pricing multiplies the number of items with the full or discounted item price found either in the Fee Package assigned to that merchant (merchant has a contract with discount prices) or in the default general package ('price list') that is assigned to the top level of that merchant hierarchy. The calculated amount is used to generate a new fee transaction.

Settled transactions contain the exact amount to be charged to the merchant so there is no need to generate a new fee transaction.

The new fee transaction and the settled transaction are called taxable transactions, and for each of them a new tax transaction must be generated. The amount for this transaction, however, is not directly calculated by *IST/MAS*, but a message is sent to an outside application to get the tax amount that should be charged to the merchant.

All these transactions are posted to the merchant accounts via regular processing.

Tax on Non-Activity Fee

Tax on Non-activity fees are supported in two ways:

- Configure tax as fee of the non-activity fee in entity's fee package. This is applicable if tax rate differs at entity or fee package level.
- Configure tax as TID_TAXABLE in Field_Value_Ctrl. This is applicable if tax rate differs at TID level.

1. Currently, *IST/MAS* interfaces to Quantum, a tax system from Vertex.
2. Currently, equipment and supplies transactions must be manually entered into the system.

Institution_id	Field_name	Field_value1	Field_value2
1	TID_TAXABLE	010004010000	14
1	TID_TAXABLE	010004020000	6
1	TID_TAXABLE	010004030000	8

The Field_value1 is TID that is taxable, and Field_value2 is the tax rate for that TID. If Field_value2 is null or zero, third party tax calculation application is used to calculate tax via DLL function call MasCalcTax present in libomas_tax library, else Field_value2 will be used as the tax rate.

The tax on activity fees is supported by configuring fee TID as TID_TAXABLE in Field_Value_Ctrl. Tax on activity fees cannot be configured as fee on a fee in entity's fee package as fee on a fee setup is used to setup Non-Qual charge for Non-Qual Pricing.

Secondary Processes

These processes are not directly part of the Pricing processing flow, but are used to prepare information required by Pricing. These processes are as follows:

- Based on the Item Count Plan settings the system accumulates transaction amounts ("Item Accumulation" on page 77). These accumulated totals are used in the Item Count Plan calculation done for the Rate and Fee Adjustment or Non-activity Fees ("Item Count Plan Calculations" on page 77).
- Pricing is based on the fees and fee groups set up via GUI during general system setup. These fees are made effective either via Fee Group Rate Update ("Fee Group Rate Update" on page 78) or Future Effective Rate Update ("Future Effective Rate Update" on page 78).
- Fees are updated via Rate and Fee Adjustment to reflect the merchant's performance over a period of time.
  See "Rate and Fee Adjustment" on page 79 for details.
- Next Settlement Date calculates the next settlement date used by Pricing to determine the transaction's settlement date.
  See "Next Settlement Date" on page 84 for details.
- Next Generate Date calculates the next generate date used by Non-activity fees to determine the next time when the non-activity fee is generated.
  See "Next Generate Date" on page 85 for details.
- Merchant Start/End Date defines the date of the entity's first clearing transaction sent to the system. This information is required for inquiry purposes.
  See "Merchant Start/End Date" on page 86 for details.

NOTE:	These processes are run at the end of the day, except for item accumulation, which is performed during transaction processing.

Item Accumulation

Each activity transaction that enters the system has three pairs of count and amount fields: one pair for sales, one for credit, and one for cashback. These fields may contain values for either individual records, meaning that the counts and amounts relate to a single transaction, or summary records, meaning that the counts and amounts relate to many transactions with the same MAS code.

Based on the Item Count Plans that have been set up in the system, the system accumulates counts and amounts from each processed transaction. This means that not all transactions are accumulated, but only those specified in the Item Count Plan.

These accumulated totals are part of the Item Count Plan calculation done for Rate and Fee Adjustment or Non-activity Fees processes.

Item Count Plan Calculations

Whenever total count (volume) or amount is required by a process (for example, Rate and Fee Adjustment or Non-activity Fees), the Item Count Plan is used to calculate the totals based on the transactions defined and the operators set up.

To calculate totals, the system performs the following:

1. Gets the accumulated totals for all transactions specified in the Item Count Plan.
2. Applies the operators in the amount group codes or the count group codes, whichever is applicable.

   An amount method group consists of a combination of operators defined in Amount Method, Additional Amt1 Method, and Additional Amt2 Method. Each of the three method fields can have one of three operators:

   - “+” => add the amount
   - “-“=> subtract the amount
   - “blank” => ignore the amount

   A count method group consists of a combination of operators defined in Count Method, Additional Cnt1 Method, and Additional Cnt2 Method. Each of the three method fields can have one of three operators:

   - “+” => add the count
   - “-“=> subtract the count
   - “blank” => ignore the count

3. Applies the operator in the Count Sign.

   There are three operators defined in the Count Sign:

   - "+" => add the result after applying the Amount or Count method group to the accumulated transaction totals.
   - "-"=> subtract the result after applying the Amount or Count method group to the accumulated transaction totals.
   - "%" => use the result after applying the Amount or Count Method to add to the total then calculate the percentage of the transaction against the total count or amount.

   See "Regular Pricing" on page 66 for an example on how operators are used by the system to perform fee calculations. Item Count Plan calculations are similar; however, the grand total (amount or count) is obtained by applying the Count Sign to the amount and count totals.

Future Effective Rate Update

The system allows for new rates to be entered into the system, rates which are not yet effective but have an effective start date. The effective start date may be set to any date, but duplicate entries for a specific MAS code or group of codes (wildcards) are not allowed.

Fee Group Rate Update

Fee Packages consist of fee rates. These can be rates for the dollar value per item, a percentage of dollar value, or both. Certain rates that are attached to a Fee Package may be bundled into groups referred to as rate groups.

Fee Group Rate Update is a stand alone process completed at the end of each day. Rates are updated using rates that are set for each subgroup in the fee group. The updated fees are applied throughout an entire group to minimize the number of updates required for each individual merchant or entity.

To update fee the following are performed:

1. The process first scans all fee groups assigned to a fee package for duplicate entries.

   If duplicate entries are found, an exception record is logged to the FEE_UPDT_EXCEPTION table, and no updates are carried out. This log can be interrogated using the Fee Update Exception form, and this way the operator can fix discrepancies in the fee group setup.

2. Based on the value of the MAS_FEE_ACTION field (I = insert, U = update, D = delete), the process selects rate records from the Fee Group table and updates them in the MAS Fees table.

   If the action is insert (I), the FEE_PKG_TID, FEE_PKG_MASCODE, and MAS_FEES tables are updated with the new fee package entry.

The action taken is preserved in the ACTION_TAKEN field (that is, one of I, U, D), and the process sets the MAS_FEE_ACTION to N, meaning that no action is required for the record.

NOTE:	Current, already effective rates are not changed, so the process can run at the same time as regular transaction processing.

Rate and Fee Adjustment

Fees and rates can be adjusted in one of the following ways:

- Rate re-assessment—rates are adjusted to reflect the processing volume of a billing period (i.e., transaction volume, average transaction amount, total amount). See "Rate Re-assessment" on page 79 for details.
- Variable pricing—fees are adjusted to reflect a user-initiated rate change for a given billing period.
  See "Variable Pricing" on page 81 for details.
- Sliding-scale pricing—fees are adjusted to reflect the processing volume of a billing period (i.e., transaction volume, average transaction amount, total amount).
  See "Sliding-scale Pricing" on page 82 for details.

Rate Re-assessment

Rate Re-assessment is used to update the per-item rates and the percent rates of the fee package in the MAS Fees to reflect the processing volume (i.e., transaction volume, average amount, total transaction amount) of the transactions defines in the Item Count Plan. The re-assessed rates allow acquirers to offer preferred rates to merchants with a higher volume of processing (transaction volume, average transaction amount, total amount).

IST/MAS calculates new rates according to configured next assessment dates, Item Count Plan IDs, and tier IDs stored in the Fee table. Rate Re-assessment replaces current rates with the new rates to be used until the next assessment date.

The re-assessment is done based on the defined assessment frequency and can be one of the following:

- Daily—every day.
- Weekly—every week.
- Bi-weekly—every two weeks.
- Monthly—every month, on the day of the month indicated in the field Day Of Month.
- Quarterly—every three months on the day of the month indicated in the field Day Of Month.

- Semi-annually—every six months on the day of the month indicated in the field Day Of Month.
- Annually— every 12 months on the day of the month indicated in the field Day Of Month.
- List—every date indicated in the list.

To perform rate re-assessment, the following setup is required:

- Set up the tier ID and the corresponding tier rates.
- Set the Item Count Plan to accumulate count and amount for the activity transactions in the billing period and their respective fees.

Once the setup is in place, the rate re-assessment flow is followed:

1. All rates to be re-assessed for the current business day are selected from the MAS Fees table (TIER_NXT_ASSM_DATE from the MAS_FEES table <= today).
2. Once rates are selected, the Item Count Plan (used to determine how transactions are accumulated for rate assessment) is retrieved, and the tier is checked.

   The Entity Fee Package table is employed to find the Entity ID that uses the Fee Package. If the Include Child Entity flag of the Item Count Plan ID is set, the processing volume includes that of the child entities. The Entity Fee Attrib table is also employed to find the Entity Group ID the Entity ID belongs to. If the Include Flag is set to Y and Child Flag is set to N, the processing volume includes that of the Entity ID. If the Include Flag is set to Y and Child Flag is set to Y, the processing volume includes that of the Entity ID and that of the child entities. If the Include Flag is set to N and Child Flag is set to N, the processing volume excludes that of the Entity ID. If the Include Flag is set to N and Child Flag is set to Y, the processing volume excludes that of the Entity ID and that of the child entities.
   See "Item Count Plan Calculations" on page 77 for details.

   There can be four different tier accumulation methods, as set in the TIER_ACCUM_METHOD from the TIER table:

   - V - based on transaction volume
   - A - based on the transaction average amount (i.e., amount/volume)
   - D - based on the total transaction amount
   - C - based on transaction volume and total transaction amount

   For TIER_ACCUM_METHOD V, A, and D, the values of TIER_RANGE_LOW and TIER_RANGE_HIGH would be count, amount, or percent based on TIER_ACCUM_METHOD.

   If TIER_ACCUM_METHOD is 'C', TIER_RANGE_LOW and TIER_RANGE_HIGH contains the transaction count, while TIER_RANGE1_LOW and TIER_RANGE1_HIGH contain the transaction amount.

3. Using defined tier tables and the total volume (or average amount or total amount) of the previous period, a new rate is determined.

   The new rates are applicable to all entities that use the particular Fee Package ID.

4. The new rates are stored in the MAS Fee table as “current rate,” and the previous rates are stored as “old rate.”

Variable Pricing

Variable pricing allows you to change rates within the billing period. At the end of the billing period, charged/calculated rates are adjusted based on the last valid rate within the billing period. This type of pricing allows you to apply adjustments to all entities, and their children, that use the same fee package.

Rates can be manually changed (or automatically changed via rate re-assessment) any time within a billing period. Fees are calculated first with the initial rate set by the user for daily calculations. When the rate changes, the calculated fee are adjusted. The latest rate within the billing period is used as the final rate and applied to the entire billing period up to and including the last processing day of the billing period.

To perform variable pricing, the following setup is required:

- Configure MAS_FEES to price incoming activity transactions. To change the rate, always create a new record in the MAS_FEES table.
- Set VARIABLE_PRICE_FLG to Y for all rates in the Fee Package that require variable processing.

Once the setup is in place, the Variable Pricing flow is followed:

1. During daily processing, activity transactions are priced using the current rates in the fee package (MAS_FEES table).
2. At the end of the billing period, Variable Pricing performs the following if the VARIABLE_PRICE_FLG is set:

   a. Calculates the initial fee—Variable Pricing accumulates (into separate accumulators in the ITEM_ACCUM table) the total count and the total amount of all activity transactions in the billing period affected by the changed fee. Also, it accumulates the total fee amount corresponding to these transactions. The fee is based on the initial rates.

   b. Calculates the actual fee—Variable Pricing applies the new rates to the accumulated totals (count and amounts) for the activity transactions in the billing period. The calculations are based on the formula:

c. actual_fee = activity_txn_count x new_per_item_rate +
activity_txn_amount x new_percent_rate/100

d. Calculates the adjustment fee—The difference between the actual fee and the initial fee is the adjustment fee.

e. Generates a fee adjustment transaction—The calculated fee-adjustment amount is the amount of the transaction.

Sliding-scale Pricing

For a billing period, fees can be calculated based on sliding-scale rates, that is, rates dependent on the processing volume. Several accumulation methods may be used through the Item Count Plan to calculate the processing volume: transaction volume, average transaction amount, total amount. This type of pricing allows you to apply adjustments to all entities, and their children, that use the same fee package.

Every rate in the fee package can be defined as a sliding-scale rate.

Fees are calculated first with an initial rate set by the user for daily calculations. At the end of the billing period, however, a new rate is determined based on the processing volume, and the calculated fees are adjusted.

NOTE:	The new rates are not updated in the fee package; they are used for fee adjustment only.

To perform sliding-scale pricing, the following setup is required:

- Set up the TIER table with the accumulation method and the corresponding tier rates. For example, set the accumulation method to D (average ticket size), and provide the appropriate rates for the defined average ticket-size ranges.
- Set the Item Count Plan to accumulate count and amount for the activity transactions in the billing period and their respective fees.
- Set the initial fees in the MAS_FEES table.
- Set SLIDING_PRICE_FLG to Y; otherwise, a regular rate re-assessment process is performed.

Once the setup is in place, the sliding-scale flow is followed:

1. During daily processing, activity transactions are priced using the initial rates in the fee package, and at the end of each day, daily counts and amounts are calculated.

   Counts and amounts for all activity transactions in the day are accumulated into separate accumulators in the ITEM_ACCUM table. Also, the total amount for their corresponding fee transactions are accumulated. These fees are based on the initial rates set in MAS_FEES.

2. At the end of the billing period, Sliding-scale Pricing performs the following if the SLIDING_PRICE_FLAG is set:

   a. Calculates the initial fee—Based on the previously accumulated daily totals, it calculates the fee amount corresponding to the activity transactions in the fee package for the entire billing period. Remember that this fee is based on the initial rates set in MAS_FEES.

   b. Determines the new rates—Using the accumulated activity totals for the billing period, it determines a tier range (from the Tier table) for the processing volume (i.e., average ticket size, transaction volume, transaction amount). The rates corresponding to this tier range is used as the new rates.

   c. Calculates the actual fee—It applies the new rates to the accumulated totals (count and amounts) for the activity transactions in the billing period. The calculations are based on the formula:

```
actual_fee = activity_txn_count x new_per_item_rate +
       activity_txn_amount x new_percent_rate/100
```

   d. Calculates the adjustment fee—The difference between the actual fee and the initial fee is the adjustment fee.

   e. Generates a fee adjustment transaction—The calculated fee-adjustment amount is the amount of the transaction.

Valid Pricing Setup Summary

As seen, several fee and rate adjustment scenarios are possible. Note, however, that only the following set-ups are valid:

Pricing Scenarios/ MAS Fees Fields	Rate Re-assessment and Variable Pricing	Rate Re-assessment	Variable Pricing	Sliding-scale Pricing
ITEM_CNT_PLAN_ID	P1	P1		P1
TIER_ID	T1	T1		T1
VARIABLE_PRICE_FLG	Y		Y	
SLIDING_PRICE_FLG				Y

NOTE:	For the Rate Re-assessment and Variable Pricing scenario, the new rate is calculated by rate re-assessment. For the Variable Pricing scenario, the rate is provided by the user.

Next Settlement Date

IST/MAS updates the Next Settlement Date according to the Payment Cycle field in the Entity Account and the Settlement Frequency flag set in the Settlement Plan TID.

If the Payment Cycle field is set to all/ALL, the Next Settlement Date is updated for all entity account entries. If the field provides a payment cycle, the Next Settlement Date is updated only for the entity account entries in the payment cycle.

Settlement Frequency flags include the following:

D	Daily
W	Weekly
M	Monthly
B	Bi-weekly
Q	Quarterly
S	Semi-annually
A	Annually
L	LIST (pre-determined list of settlement dates specified by merchant)

For fees and payments that have the Settlement Frequency set to Daily, the Next Settlement Date is the next business day.

For fees and payments that have the Settlement Frequency set to other than Daily, the next settlement date is calculated based on the current settlement date, frequency, and day of the month (if the frequency is monthly). The following table shows how the Next Settlement Date is calculated:

Settlement Frequency	Next Settlement Date
Weekly	current settlement date + 7 days
Bi-weekly	current settlement date + 14 days
Monthly	current settlement date + 1 month If the calculated day 31 and the month is February, the date is moved to February 28 or 29.
Quarterly	current settlement date + 3 months

Settlement Frequency	Next Settlement Date
Semi-annually	current settlement date + 6 months
Annually	current settlement date + 12 months
List	next date in the list

Next Generate Date

Next Generate Date is used to calculate the next generate date used by Non-activity fees to determine the date of the next time when the non-activity fee is generated.

IST/MAS updates the Next Generate Date according to the Generate Frequency flag set in the NON_ACTIVITY_FEES table.

Generate Frequency flags include the following values:

D	Daily
W	Weekly
M	Monthly
B	Bi-weekly
Q	Quarterly
S	Semi-annually
A	Annually
L	LIST (pre-determined list of generate dates specified by merchant)

For non-activity fees that have the Generate frequency set to Daily, the Next Generate Date is the next business day.

For non-activity fees that have the Generate Frequency set to other than Daily, the next generate date is calculated based on the current generate date present in the Next Generate Date field, generate frequency, and day of the month (if the frequency is monthly). The following table shows how the Next Generate Date is calculated:

Generate Frequency	Next Generate Date
Weekly	current generate date + 7 days
Bi-weekly	current generate date + 14 days
Monthly	current generate date + 1 month If the calculated day 31 and the month is February, the date is moved to February 28 or 29.

Generate Frequency	Next Generate Date
Quarterly	current generate date + 3 months
Semi-annually	current generate date + 6 months
Annually	current generate date + 12 months
List	next date in the list

Merchant Start/End Date

For inquiry purposes, the system records the Actual Start Date and the Last Clearing Date for a merchant. The Actual Start Date defines the date when the entity's first clearing transaction was sent to the system. This information is updated only once. The Last Clearing Date defines the date when the entity's last transaction was sent to the system. This information is updated every time an activity transaction is processed.

To update the Actual Start Date of each entity, the system performs the following:

1. Selects all merchants without a start date and verifies whether there are any transactions for this merchant in the MAS_TRANS_LOG table.
2. Updates the merchant's actual start date with the oldest ACTIVITY_DATE_TIME found among transactions for that merchant.
3. If there is no transactions for a particular merchant, meaning merchant did not start working yet, the actual start date is left empty.

Cascade Fee Split

This feature applies to non-activity fees and activity fees.

The field SPLIT_FEE_TYPE in SETTL_PLAN_TID table supports splitting of fees among the immediate child entities level. The following are the valid values of SPLIT_FEE_TYPE field that displays based on the requirement:

- E - Evenly split
- V - Volume
- A - Amount

Set the SETTL_PLAN_TID.SPLIT_FEE_TYPE of the child entity to E, V or A to split the fee down the hierarchy where, the hierarchy may reach the leaf child.

This feature traverse recursively down the hierarchy based on settl_plan_tid.split_fee_type.

For example:

In the hierarchy below, the system will first consider the split_fee_type of the originating posting entity (2000000000). Since the split_fee_type is 'E', it will split the fee evenly among it's immediate children (2100000000 and 2200000000), excluding it. Then, it will traverse down the immediate children and look at their split_fee_type. If the entities 2100000000 and 2200000000 split fee types are also 'E', then the fee should further evenly split to their child entities 2110000000 and 2210000000.

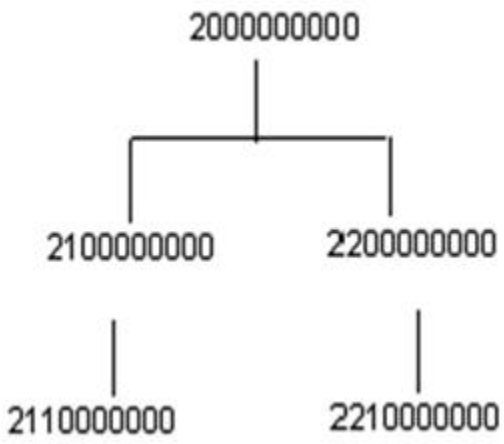


Figure 7 Cascade Fee Split

The following figure presents the Cascade Fee Split processing flow:

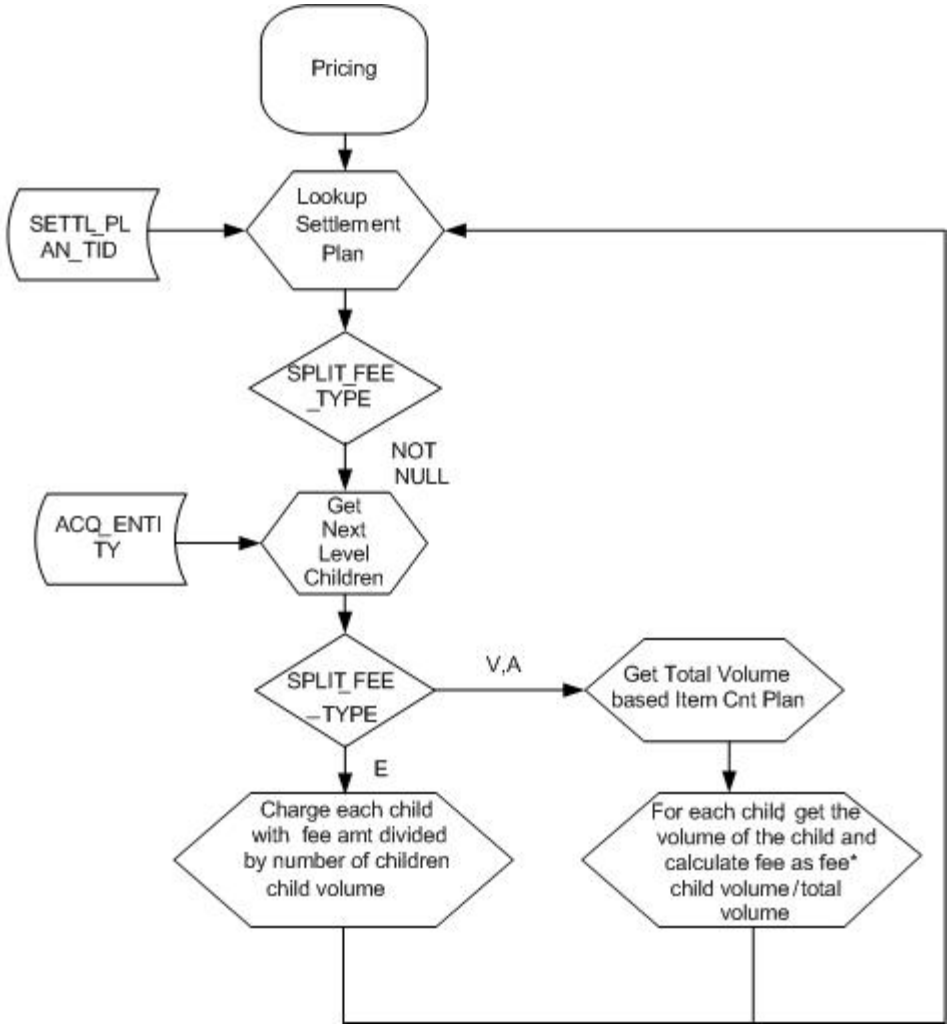


Figure 8 Cascade Fee Split Processing Flow

Fee Cap

The feature of charging a fee to a cap amount is applied to both non-activity fees and activity fees.

The following fields are added to ENT_FEE_REMAIN_PRD:

- INSTITUTION_ID
- ENTITY_ID
- NON_ACT_FEE_PKG_ID

- CHARGE_METHOD
- TID
- MAS_CODE
- NBR_REMAIN_PERIOD
- INC_CHILD_ENT_FLAG
- ENTITY_GROUP_ID
- CAP_TO_BE_MET
- CAP_ACCUM_AMT
- TOTAL_ACCUM_AMT
- CAP_FREQ
- CAP_DATE_LIST_ID
- CAP_DAY_OF_MONTH
- LAST_CAP_DATE
- NEXT_CAP_DATE

The Non-Activity Fee process reads the ENT_FEE_REMAIN_PRD and charge the fee until CAP_TO_BE_MET is met.

The table ENTITY_FEE_ATTRIB is added to hold the activity fee information for each entity. The fields are:

- INSTITUTION_ID
- ENTITY_ID
- FEE_PKG_ID
- TID_ACTIVITY
- MAS_CODE
- TID_FEE
- INC_CHILD_ENT_FLAG
- ENTITY_GROUP_ID
- CAP_TO_BE_MET
- CAP_ACCUM_AMT
- TOTAL_ACCUM_AMT

- CAP_FREQ
- CAP_DATE_LIST_ID
- CAP_DAY_OF_MONTH
- LAST_CAP_DATE
- NEXT_CAP_DATE

The Pricing Server reads ENTITY_FEE_ATTRIB and charge the fee until CAP_TO_BE_MET is met.

After reaching the CAP, the fee will need to be charged again in the next period. This is achieved by adding LAST_CAP_DATE and NEXT_CAP_DATE to allow user to setup the starting date of the CAP period. If NEXT_CAP_DATE is less than or equal to GL_DATE, Pricing server or Non-activity fee process will reset CAP_ACCUM_AMT and TOTAL_ACCUM_AMT to zero and set NEXT_CAP_DATE to the start of the next cap period.

Even if CAP_TO_BE_MET is already met, Pricing server and Non-activity fee process will continue to accumulate fee to TOTAL_ACCUM_AMT. TOTAL_ACCUM_AMT is used for checking whether or not user can still update CAP_TO_BE_MET in GUI.

NOTE:	a) If Cap_freq is NULL then next_cap_date will be incremented by 1 day. b) If Last_cap_date and next_date_date are NULL before processing Non-activity fees because the user didn't setup the dates then the system will assign last_cap_date with the current GL_Date and next_cap_date with GL_Date plus 1 day.

Requirements in GUI

If TOTAL_ACCUM_AMT is greater than or equal to CAP_TO_BE_MET (in this case CAP_ACCUM_AMT is equal to CAP_TO_BE_MET), GUI should disable update on CAP_TO_BE_MET. This is to prevent inconsistency of when the discrepancy will be charged. For example, if the user changes CAP_TO_BE_MET to be less than or greater than CAP_ACCUM_AMT, any reimbursement or any charges can only be done on the next fee charge. If there is no fee charge before the end of the period, the discrepancy will have to be calculated on the next period. By restricting the user to update CAP_TO_BE_MET when cap is already met will ensure that fees are not carried forward to the next period and the fees charge for the period does not exceed the cap.

After user had changed CAP_TO_BE_MET, GUI should verify that the changed amount is greater than or equal to CAP_ACCUM_AMT. If the changed amount is less than CAP_ACCUM_AMT, GUI should display a message indicating that cap to be met can only be changed to value greater than or equal to CAP_ACCUM_AMT.

The following figure presents the Non-Activity Fee Cap processing flow:

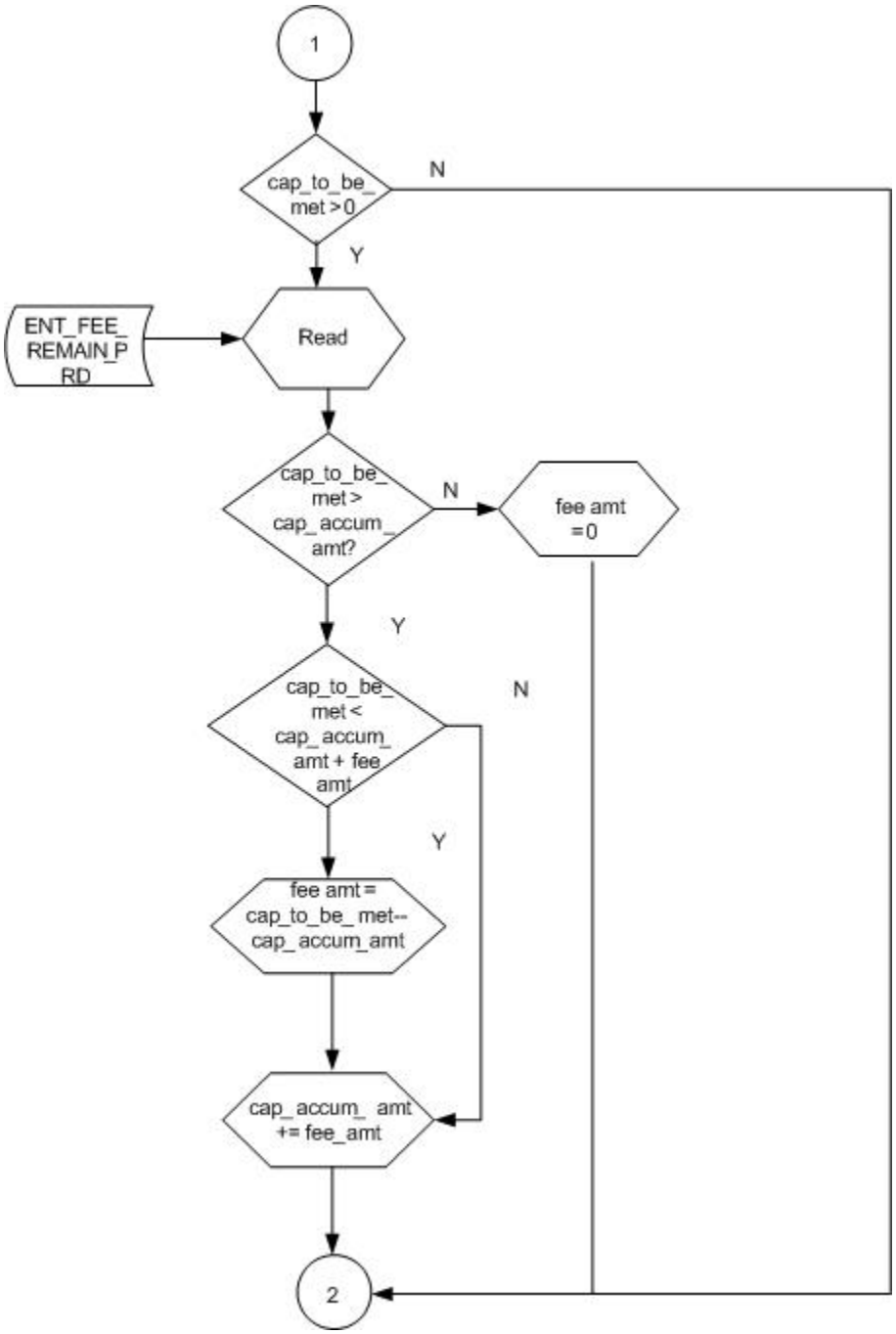


Figure 9 Non-Activity Fee Cap Processing Flow

After reaching the CAP, the fee will need to be charged again in the next period. This is achieved by adding LAST_CAP_DATE and NEXT_CAP_DATE to allow user to setup the starting date of the CAP period. If NEXT_CAP_DATE is less than or equal to GL_DATE, Pricing server or Non-activity fee process will reset CAP_ACCUM_AMT and TOTAL_ACCUM_AMT to zero and set NEXT_CAP_DATE to the start of the next cap period.

Even if CAP_TO_BE_MET is already met, Pricing server and Non-activity fee process will continue to accumulate fee to TOTAL_ACCUM_AMT. The TOTAL_ACCUM_AMT is used for checking whether or not user can still update CAP_TO_BE_MET in GUI.

Adjustments

Transactions may be adjusted to debit or credit the recipient's account. Adjustments may be made to Fees, Payments, or Billing Requisitions for equipment and supplies. Adjustments generate activity transactions that are sent to Validation. From there they follow the regular processing path (i.e, sent to Pricing, which generates the corresponding fee and tax transactions).
See "Tax Calculations" on page 75 for an example on how tax transactions are handled.

Adjustment Transactions can be either generated through the GUI or imported from an external system via a file. The file format is similar to the standard MAS format supported for activity transactions with the exception that each transaction can be associated with a free text description.
See "Adjustment Input File" on page 177 for details.

Adjustments are typically assigned their own transaction class (TID class = 05) and require configuration similar to other activity transactions. This is a common setup for adjustment transactions submitted though both GUI and files.

- Each adjustment type can be assigned a TID.
- TID should be configured as allowed for the merchant.
- Settlement Plan, Account Posting Plan must specify settlement/posting options for each adjustment transaction.

The following are sample adjustments:

- GL Adjustment—Adjustment transactions could be non-settlement transactions that only perform GL account adjustment credit/debit.
- Volume Adjustment—Adjustments can also be non-settlement transactions that are used only for volume adjustment.
- Merchant Payment Adjustment—Traditional adjustment transactions can be configured to credit or debit merchant account. The appropriate reason/free text description can be supplied.

- Merchant Fee Adjustment—Charge a miscellaneous fee or adjust a fee already charged.

Resubmit Error Transactions

Individual transactions may be suspended due to pricing and posting errors (like invalid settlement plan for fees, invalid account posting plan and tax calculation failures) and saved to the MAS_TRANS_ERROR table. Through the GUI, these errors must be manually corrected, the transactions enriched with the information Validation is normally providing ("Validation Enrichment" on page 209), and resubmitted to Posting. These transactions are processed at the end of each day.

If an ISO parent TID exists (that is, ISO billing is applied), the transaction is always sent to Pricing, which calculates the fees applicable to the ISO parent; otherwise, the transaction is sent to Posting.

If too many transactions need to be corrected, the following procedure can be applied:

1. Submit the entire Input File as a reversal file. All transactions from the Input File are reversed in the system.
2. Correct the data setup.
3. Resubmit the whole Input File.

Non-activity Fees

Each merchant entity can be associated with many non-activity fees for such items as terminal and PIN-pad rental. Non-activity Fees are periodically applied to merchant entities based on preferences of date, time window and frequency. The Scheduler starts the whole process at the end of the day, but dates for charging fees are defined during non-activity fee setup.

Non-activity fee types (charge methods) include the following:

Non-activity Fee	Description
Recurring Fees	A fee charged periodically. See "Recurring Fees" on page 104.
Fixed Number of Installments	A fee charged in equal amounts over a number of periods. See "Fixed Number of Installments" on page 104.
Seasonal Non-activity Fees	A fee charged only during a pre-defined period in the year. See "Seasonal Charges" on page 105.
Minimum Charges	A fee charged as a minimum amount for a fee normally billed through regular processing. See "Minimum Charges" on page 105.

Non-activity Fee	Description
Penalty Charges	A fee charged on failure to reach a threshold value. See "Penalty Charges" on page 107.
Tiered Pricing	A fee charged on a tier structure. See "Tier Pricing with Minimum Charges" on page 108.
Differential Tier Pricing	A fee charged on a tier structure and based on variable volume rates. See "Differential Tier Pricing" on page 109
Each Transaction	A fee charged as a per item rate for each transaction that exceeds the threshold value. See "Each Transaction" on page 109.
Flat Fees	A fixed fee charged on reaching a threshold value. See "Flat Fees" on page 110.
ACH Fees	A fee charged for the number of payments made via ACH. See "ACH Fees" on page 111.
Wire Fees	A fee charged for the number of payments made via Wire transfer. See "Wire Fees" on page 111.
Batch Header Fees	A fee charged for the number of deposits (deposit batch headers) received from the entity within a specified period. See "Batch Header Fees" on page 111.

The frequency with which to calculate these fees is defined by the Generate Frequency as follows:

Daily	Charge this fee every day.
Weekly	Charge this fee every week.
Bi-weekly	Charge this fee every 2 weeks.
Monthly	Charge this fee every month, on the day of the month indicated in the field Day Of Month.
Quarterly	Charge this fee every 3 months on the day of the month indicated in the field Day Of Month.
Semi-annually	Charge this fee every 6 months on the day of the month indicated in the field Day Of Month.
Annually	Charge this fee every 12 months on the day of the month indicated in the field Day Of Month.
List	Charge this fee every date indicated in the list.

For easier maintenance and application, non-activity fees are configured in packages. This organization allows for common fee packages to be configured and applied as groups. Also, multiple acquiring entities can share common fee packages via a pointer, eliminating duplication during non-activity fee setup.

Although multiple fee packages exist, only one combination of charge method, TID, and MAS code is allowed. When duplication exists, only one fee is charged, and the active combination is the one in the most current non-activity fee package.

Non-activity fee packages are set up in the ENT_NONACT_FEE_PKG table, which links acquiring entities to their respective non-activity fee packages. Each unique NON_ACT_FEE_PKG record contains one or more NON_ACTIVITY_FEES records to create a non-activity fee package.

To add a new type of non-activity fee, a fee TID is configured in the TID table and a charge type associated with the fee. If there is no need for a new charge type, no coding is required. When a new charge type is required, the coding effort is significantly reduced if existing building blocks are used: Item Count Plans, Tier Tables, etc.

Processing Flow

All non-activity fees follow the same processing flow. They differ only in the way the fee amount is calculated. That is, they use different source elements and formulas to calculate the fee amount.

To apply non-activity fees, the process performs the following:

The following figure presents the Non-Activity Fee Package Inheritance processing flow:

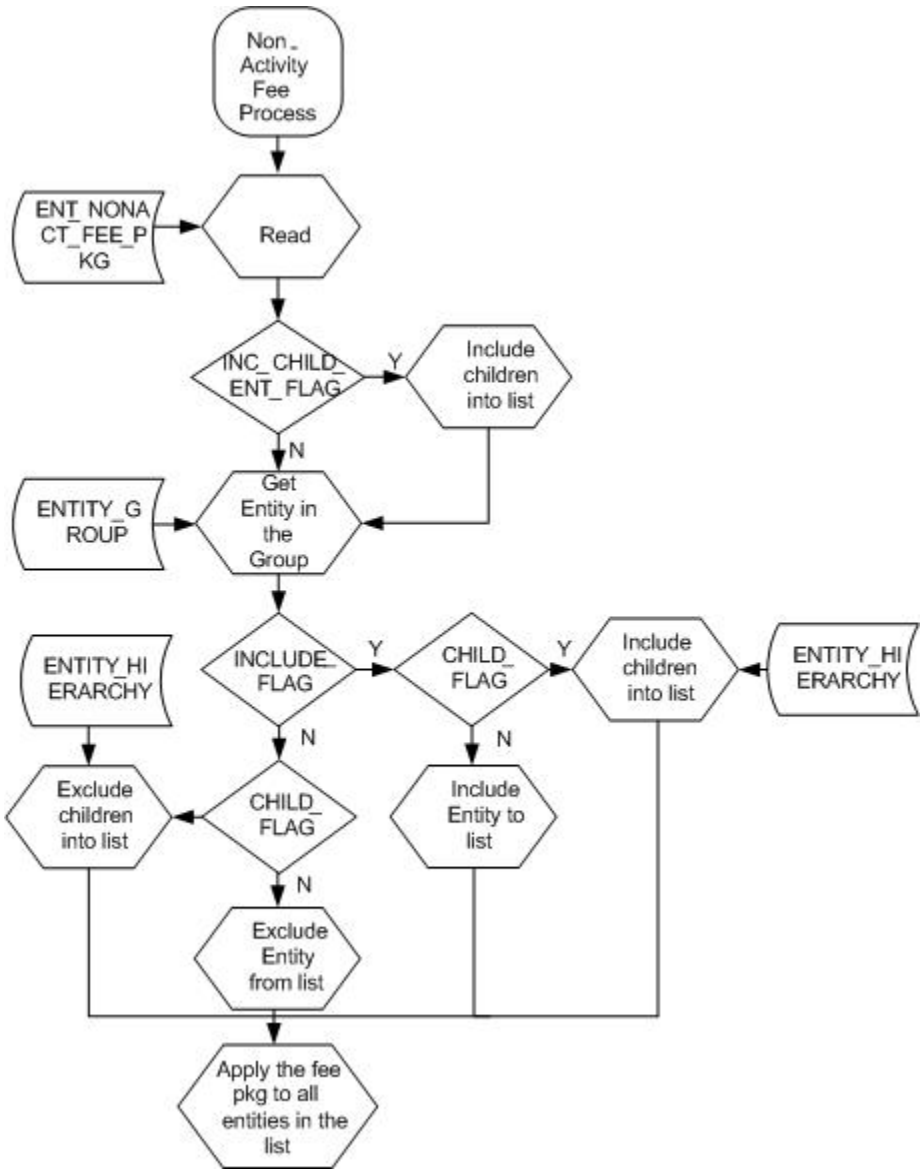


Figure 10 Non-Activity Fee Package Inheritance Processing Flow

ENTITY_GROUP should be assigned to each ENT_NONACT_FEE_PKG. To apply the non-activity fee package to all child entities except one entity, the entity to be excluded will be added to ENTITY_GROUP with INCLUDE_FLAG set to 'N'. To exclude the

children entities, set CHILD_FLAG to 'Y'. To apply the fee package to only 3 entities out of all child entities, the 3 entities to be included will be added to ENTITY_GROUP with INCLUDE_FLAG set to 'Y'. To include the children entities, set CHILD_FLAG to 'Y'.

In GUI, it allows ENTITY_GROUP_ID to be assigned to entities belonging to the group.

1. From ENT_NON_ACT_FEE_PKG, select all effective non-activity fee packages of each entity. The effective fee package is obtained by comparing the next generate date with the system date. If the next generate date is smaller or equal to the system date the fee package is considered in effect.

2. For each record in the ENT_NONACT_FEE_PKG that matches the selected fee package ID, check the activity flag in the Non-activity Fees table.

   If the Check Activity flag is set to N in the Non-activity Fees table, the fee transaction is automatically generated.

   If the Check Activity flag is set to Y, the non-activity fee is generated only if the date of the last processed transaction, as recorded in the Acquiring Entity table, is between the last generate date and the next generate date. This flag allows, within a non-activity fee package, some non-activity fees to be charged unconditionally, while others to be charged only if activity exists.

   LAST_GENERATE_DATE < ACQ_ENTITY.LAST_TRANS_CL_DATE <= NEXT_GENERATE_DATE.

   Since the system allows multiple fee package, keep in mind that if two non-activity fee packages—one “in season” and one “out of season”— have overlapping effective dates, both fee packages are effective. If the same non-activity fee exists in both fee packages, the one with the latest start date is used.

   For example,

```
ENT_NONACT_FEE_PKG
ENTITY_ID         NON_ACT_FEE_PKG_ID  START_DATE    END_DATE
20001             NAFP1               12/1/2002     5/30/2003
20001             NAFP2               5/1/2003      11/30/2003

NON_ACTIVITY_FEES
NON_ACT_FEE_PKG_ID      CHARGE_METHOD       FEE_AMT
NAFP1                   R                   100
NAFP1                   M                    50
NAFP2                   R                   200
```

   On processing day of 5/30/2003, both NAFP1 and NAFP2 are effective, and the following non-activity fees are charged:

```
NON_ACT_FEE_PKG_ID CHARGE_METHOD     FEE_AMT
NAFP2              R                 200
NAFP1              M                 50
```

During initial setup, if the LAST_GENERATE_DATE is not supplied, the system calculates the last generate date based on next generate date and generate frequency.

For example,

```
ENT_NONACT_FEE_PKG
ENTITY_ID  NON_ACT_FEE_PKG_ID   START_DATE    END_DATE
20001      NAFP1                4/20/2003     10/30/2003

NON_ACTIVITY_FEES
FEE_PKG_ID LAST_GEN_DATE  NEXT_GEN_DATE  G_FREQ CHECK_ACTIVITY
NAFP1                    4/30/2003       M      Y
```

The system calculates the last generate date to be 3/30/2003, and if the LAST_TRANS_CL_DATE is 4/14/2003, the NAFP1 will still be charged although the START_DATE is 4/20/2002.

3. For Minimum Charge, Penalty Charge, Tier Pricing, Differential Pricing, Each Transaction, and Flat Fee charge methods, calculate the Tier Period using the following fields:
   - TIER_FREQ
   - TIER_DATE_LIST_ID
   - TIER_ASSM_DAY
   - TIER_LASTASSM_DATE
   - TIER_NXT_ASSM_DATE

   The TIER_NXT_ASSM_DATE less number of days depending on TIER_FREQ defines the start date of the tier period. The start date of the period plus number of days depending on TIER_FREQ defines the end date of the tier period.

   If TIER_LASTASSM_DATE is null or 01-JAN-1980, it is calculated as TIER_NXT_ASSM_DATE less number of days depending on TIER_FREQ. If TIER_FREQ is not set, GENERATE_FREQ is used as TIER_FREQ.

   If TIER_ASSM_DAY is set to a different value than the day of month of TIER_LASTASSM_DATE and TIER_NXT_ASSM_DATE, the tier start date and end date is adjusted according to TIER_ASSM_DAY.

   If the TIER_ASSM_DAY differs from the day in tier date range, then the TIER_LASTASSM_DATE and TIER_NXT_ASSM_DATE should be updated according to TIER_ASSM_DAY.

   The TIER_NXT_ASSM_DATE can either be set to a value (Set Tier Period) or set as NULL (Dynamic Tier Period).

   In the cases explained below, the Tier_assm_day used is as same as the day of month of Tier_nxt_assm_date and Tier_lastassm_date.

Set Tier Period

The Set Tier Period is configured by setting TIER_NXT_ASSM_DATE to date greater than 01-JAN-1980.

The second row in the cases below are the result after running non-activity fee process.

1. Generate_Freq same frequent as Tier_Freq

Case A1: Next Generate Date is greater than Tier Last Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	M	12/31/2007	1/31/2008
M	2/29/2008	3/31/2008	M	1/31/2008	2/29/2008

Tier Start Date: 12/31/2007

Tier End Date: 1/31/2008

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/15/2008	2/15/2008	M	1/31/2008	2/29/2008
M	2/15/2008	3/15/2008	M	2/29/2008	3/31/2008

Tier Start Date: 1/31/2008

Tier End Date: 2/29/2008

Case A2: Next Generate Date is less than or equal to Tier LastAssm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	M	2/29/2008	3/31/2008
M	2/29/2008	3/31/2008	M	2/29/2008	3/31/2008

Tier Start Date: 2/29/2008

Tier End Date: 3/31/2008

2. Generate_Freq more frequent than Tier_Freq

Case B1: Last Generate Date is greater than or equal to Tier NxtAssm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	A	12/31/2006	12/31/2007
M	2/29/2008	3/31/2008	A	12/31/2006	12/31/2007

Tier Start Date: 12/31/2006

Tier End Date: 12/31/2007

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	A	2/28/2006	2/28/2007
M	2/29/2008	3/31/2008	A	2/28/2007	2/29/2008

Tier Start Date: 2/28/2006

Tier End Date: 2/28/2007

Case B2: Next Generate Date is within Tier Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/15/2008	2/15/2008	A	2/28/2007	2/29/2008
M	2/15/2008	3/15/2008	A	2/28/2007	2/29/2008

Tier Start Date: 2/28/2007

Tier End Date: 2/29/2008

Case B3: Next Generate Date is greater or equal to Tier Nxt Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	2/15/2008	3/15/2008	A	2/28/2007	2/29/2008
M	3/15/2008	4/15/2008	A	2/29/2008	2/29/2009

Tier Start Date: 2/28/2007

Tier End Date: 2/29/2008

Case B4: Next Generate Date is less than or equal to Tier Last Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	A	2/29/2008	2/28/2009
M	2/29/2008	3/31/2008	A	2/29/2008	2/28/2009

Tier Start Date: 2/29/2008

Tier End Date: 2/28/2009

3. Generate_Freq less frequent than Tier_Freq

Case C1: Next Generate Date is greater than Tier Last Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
A	2/28/2007	2/29/2008	M	1/31/2007	2/28/2007
A	2/29/2008	2/28/2009	M	2/28/2007	2/29/2008

Tier Start Date: 1/31/2007

Tier End Date: 2/28/2007

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
A	2/15/2007	2/15/2008	M	1/31/2008	2/29/2008
A	2/15/2008	2/15/2009	M	2/29/2008	2/28/2009

Tier Start Date: 1/31/2008

Tier End Date: 2/29/2008

Case C2: Next Generate Date is less than or equal to Tier Last Assm Date:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
A	2/28/2007	2/29/2008	M	2/29/2008	3/31/2008
A	2/29/2008	2/28/2009	M	2/29/2008	3/31/2008

Tier Start Date: 2/29/2008

Tier End Date: 3/31/2008

Dynamic Tier Period

The dynamic Period volume is configured by setting TIER_NXT_ASSM_DATE to NULL or 01-JAN-1980. In this case, the tier period is dynamic which changes according to the billing period and includes the volume of current billing period. The tier period is dynamic when TIER_NXT_ASSM_DATE is NULL and NEXT_GENERATE_DATE is used as the tier end date. Hence, the tier period changes as the billing period changes. The start date of the tier period is NEXT_GENERATE_DATE less the day based on TIER_FREQ, and end date of the tier period is NEXT_GENERATE_DATE.

1. Generate_Freq same frequent as Tier_Freq:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	M		
M	2/29/2008	3/31/2008	M	2/29/2008	1/1/1980

Tier Start Date: 1/31/2008

Tier End Date: 2/29/2008

2. Generate_Freq more frequent than Tier_Freq:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
M	1/31/2008	2/29/2008	A		
M	2/29/2008	3/31/2008	A	2/29/2008	1/1/1980

Tier Start Date: 2/28/2007

Tier End Date: 2/29/2008

3. Generate_Freq less frequent than Tier_Freq:

Generate Freq	Last Generate Date	Next Generate Date	Tier Freq	Tier LastAssm Date	Tier NxtAssm Date
A	2/28/2007	2/29/2008	M		
A	2/29/2008	2/28/2009	M	2/29/2008	1/1/1980

Tier Start Date: 1/31/2008

Tier End Date: 2/29/2008

4. Creates a fee transaction using data from the Entity Non-activity Fee Package, Non-activity Fee Packages, Non-activity Fees, and Settlement Plan TID tables.

   The fee amount is calculated based on the non-activity fee applied.
   See "Recurring Fees" on page 104.
   See "Fixed Number of Installments" on page 104.
   See "Seasonal Charges" on page 105.
   See "Minimum Charges" on page 105.
   See "Penalty Charges" on page 107.
   See "Tier Pricing with Minimum Charges" on page 108.
   See "Differential Tier Pricing" on page 109
   See "Each Transaction" on page 109.
   See "Flat Fees" on page 110.
   See "ACH Fees" on page 111.
   See "Wire Fees" on page 111.

5. Calculates the next generate date and updates the Non Activity Fees table. See "Next Generate Date" on page 85 for details on how the Next Generate Date is calculated.

6. Sends the non-activity fee transaction to Pricing.

Recurring Fees

Use this charge method to charge a fee on a periodic basis. For example, 'XYZ Processor' charges a terminal rental fee of $300 semi-annually to 'New Store'. Charge period is one of the available non-activity generation frequencies.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- fee amount

The value of the fee amount element is used as the amount of the fee transaction.

Fixed Number of Installments

This charge method is similar to the Recurring Fee type, but it is used when a fee is charged in equal amounts over a fixed number of periods. For example, the merchant must pay the fee in five monthly installments. After each monthly installment is paid, the number of remaining installments is decreased. When the installment counter reaches zero, the charge is stopped.

The number of outstanding installments is maintained in the NBR_REMAIN_PERIOD field from the ENT_FEE_REMAIN_PRD table. The initial value for this field is given by the NBR_INSTALLMENTS field in the NON_ACTIVITY_FEES table.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- number of remaining periods
- fee amount

The value of the fee amount element is used as the amount of the fee transaction.

Seasonal Charges

This charge method is similar to the Recurring Fee type, but it is charged only during a season defined by a start and end date. For example,

- May 1 to October 5
- November 20 to February 5

These fees may be applied to selected merchants based on their seasonal-driven business, so that they are changed only during a pre-defined period.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- start day and month
- end day and month
- fee amount

The value of the fee amount element is used as the amount of the fee transaction.

NOTE:	The calculated generate date is adjusted to take into account the season's start and end dates. If the calculated generate date is greater than end day and month of the current year, then the generate date is determined for the next season using start day and month and the current year plus one.

Minimum Charges

Minimum monthly charges may be applicable to certain merchants for specific charge categories (for example, activity-related charges, supplies or one-time charges).

Use this charge method to charge a minimum amount for a fee normally billed through regular processing. This is based on the accumulation of fees already billed. The minimum charge can also be based on accumulation of multiple fees. You only need to specify each one in the Item Count Plan you setup for the minimum charge record.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- Item Count Plan ID
- fee amount

To calculate the fee amount to be charged, the process retrieves previously charged fees that are accumulated in the former period (using the Item Count Plan ID) and gets the total amount from the Item Accumulator table.
See "Item Count Plan Calculations" on page 77 for details on how the accumulation is done.

If the Include Child Entity flag is set in the Item Count Plan, all child entities for the parent entity are located and their count or amount is included in the parent's total count. The Entity Fee Remain Prd table is employed to find the Entity Group ID the Entity ID belongs to. If the Include Flag is set to Y and Child Flag is set to N, the processing volume includes that of the Entity ID. If the Include Flag is set to Y and Child Flag is set to Y, the processing volume includes that of the Entity ID and that of the child entities. If the Include Flag is set to N and Child Flag is set to N, the processing volume excludes that of the Entity ID. If the Include Flag is set to N and Child Flag is set to Y, the processing volume excludes that of the Entity ID and that of the child entities.

If this value is greater than the fee amount from the Non- activity Fees table, the system does nothing; otherwise, it calculates the difference between the fee amount and the previously charged amount. This differences the amount to be charged.

The way in which the minimum charge is applied depends on how the Settlement Plan, Account Posting Plan, and Non-activity Fees are set up. The simplest way to apply minimum charges is to setup the same Generate Frequency for the TIDs you are using to accumulate totals in the Settlement Plan and the Fee TID in Item Count Plan.

For example, if you want to charge a minimum amount for fee 401 (fee you charge based on activity) using a monthly frequency, perform the following settings. For this discussion, fee 401 is the regular fee and fee 471 is the minimum charge fee associated with the accumulation of 401s during a period.

- Set up the 401 TID in the entity's Settlement Plan to have the Settlement Frequency "Monthly".
- Set up the non-activity fee (for example, 471) to have the same settlement Frequency in the Settlement Plan and to have the same Generate Frequency in the Non-Activity Fee.
- Ensure that the Next Generate Date and the day of the month for the 471 in the Non-activity Fee record is the same as the one set in the Settlement Plan for the 401 transaction.
- Both 401 and 471 entries should be assigned the same posting entity so that both transactions are posted to the same Entity Account.

This way the two fees, 401 and 471, appear in the entity's account on the same date.

Penalty Charges

Use this charge method to charge a fee based on failure to reach a threshold value. For example, a fee can be charged to an entity if an agreed on transaction volume is not attained over a specified period.

For example, if you want to charge an entity a penalty fee when the volume of sales transactions is less than 100,000 per month you would setup a Non-activity fee record of type “Penalty” and assign to the fee record an Item Count Plan containing the definition of the sales transaction to be counted. Then, you would specify a penalty amount, a threshold value (100,000), and a monthly generate frequency.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- Item Count Plan ID
- fee amount
- threshold value

To calculate the fee amount, the process:

1. Determines the transaction volume for the defined period.
   See “Item Count Plan Calculations” on page 77 for details on how the accumulation is done. If the Include Child Entity flag is set in the Item Count Plan, all child entities for the parent entity are located and their count or amount is included in the parent’s total count. The Entity Fee Remain Prd table is employed to find the Entity Group ID the Entity ID belongs to. If the Include Flag is set to Y and Child Flag is set to N, the processing volume includes that of the Entity ID. If the Include Flag is set to Y and Child Flag is set to Y, the processing volume includes that of the Entity ID and that of the child entities. If the Include Flag is set to N and Child Flag is set to N, the processing volume excludes that of the Entity ID. If the Include Flag is set to N and Child Flag is set to Y, the processing volume excludes that of the Entity ID and that of the child entities.
2. Compares the actual volume with the threshold count. If the volume is smaller than the threshold count, *IST/MAS* charges the penalty fee; otherwise, it does nothing.

Tier Pricing with Minimum Charges

Use this charge method to charge a fee, within the current period, based on a tier structure. The tier is applied within the Generate Frequency period, or the period defined in a Generate Date List.

When the Non-activity Fee process is used to generate fees for activity transactions, the accumulation period varies depending on the Generate Frequency or Generate Date List selected. For example, if you select “Daily” frequency, then the tier is applied on the daily accumulation of totals. If you are using Generate Date list, it is applied based on the period between generate dates in the list.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- Item Count Plan ID
- tier ID
- threshold value (minimum fee amount, can be 0 if fee not based on minimum amount)

To calculate the fee amount, the process:

1. Calculates the transaction volume and transaction amount for the defined period using accumulation rules in the Item Count Plan.
   See “Item Count Plan Calculations” on page 77 for details on how the accumulation is done. If the Include Child Entity flag is set in the Item Count Plan, all child entities for the parent entity are located and their count or amount is included in the parent’s total count. The Entity Fee Remain Prd table is employed to find the Entity Group ID the Entity ID belongs to. If the Include Flag is set to Y and Child Flag is set to N, the processing volume includes that of the Entity ID. If the Include Flag is set to Y and Child Flag is set to Y, the processing volume includes that of the Entity ID and that of the child entities. If the Include Flag is set to N and Child Flag is set to N, the processing volume excludes that of the Entity ID. If the Include Flag is set to N and Child Flag is set to Y, the processing volume excludes that of the Entity ID and that of the child entities.
2. Uses the calculated volume or amount or average ticket to determine from the Tier Table the rate to be charged.
3. Calculates fees to be charged.

   ```
   ((amount * corresp_rate_%) + (volume * corresp_rate_per_item))
   ```

4. Compares this amount with the threshold amount, if set in the Non Activity Fees table, and charges the greater of the two amounts.

For example, you can bill for a given TID (e.g., 305) as follows: if the volume is in the range 1-1000 items, the fee charged is transaction volume times $0.07; if the volume is over 1000 transactions, the fee charged is transaction volume times $0.03.

Differential Tier Pricing

This charge method is similar to the Tier Pricing type, but it is used to charge a fee based on a variable grid. For example, you can bill for a given TID (e.g., 305) the first 1000 transaction at $0.07, the next 5000 transactions at $0.05, and anything over 6000 at $0.03.

The only difference between the two pricing methods is in the way the charged fee is calculated. That is, the differential tier pricing calculates fees based on ranges defined in the tier table, as described in the example.

Each Transaction

This is a fee charged on per item rate for each transaction that exceeds the threshold value. For example, you would use this charge method to charge a per item fee for excessive chargeback transactions.

The system uses the Item Count Plan assigned to the fee record to get the percentage of the total number of certain transaction types. If this percentage is greater than the threshold percentage then the Fee Amount is charged for each transaction above the threshold percentage.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- Item Count Plan ID
- fee amount
- threshold value

For example, monthly fees are charged for excessive chargebacks and there is a $10 fee charged for each chargeback transaction above 8% of the total number of transactions. To accomplish this setup, follow the steps:

1. Set up a “Each Trans” non-activity transaction.
2. Set the fee amount to $10.

   Set the Item Count Plan to be used to determine the total number of transactions. See “Item Count Plan Calculations” on page 77 for details on how the accumulation is done. If the Include Child Entity flag is set in the Item Count Plan, all child entities for the parent entity are located and their count or amount is included in the parent’s total count. The Entity Fee Remain Prd table is employed to find the Entity

Group ID the Entity ID belongs to. If the Include Flag is set to Y and Child Flag is set to N, the processing volume includes that of the Entity ID. If the Include Flag is set to Y and Child Flag is set to Y, the processing volume includes that of the Entity ID and that of the child entities. If the Include Flag is set to N and Child Flag is set to N, the processing volume excludes that of the Entity ID. If the Include Flag is set to N and Child Flag is set to Y, the processing volume excludes that of the Entity ID and that of the child entities.

3. Set the Threshold to 5% => 5% of total transactions determined by the Item Count Plan.

4. If the number of transactions in Mas file is 6, then the fee charged would be:

```
ptcnt- (threshold/100*tcnt)) * fee_amt
ie, (6--(5/100)*6)*10 =>(6-(0.05*6))*10 =>(6-0.3)*10=> 5.7*10 =>
57
ptcnt -> Percentage Count and
tcnt -> Transaction Count ( number of transactions in Mas file)
```

Flat Fees

Use this charge method to charge a fixed fee on reaching a threshold value.

For example, if you want to charge a fixed amount whenever an entity's percentage chargeback exceeds an allowed percentage of the total number of transactions within a specified period.

This charge method is similar to the Each Transaction charge method, except that the amount charged is not a per item fee. In this case, the Fee Amount is charged as a fixed fee once the threshold percentage is exceeded.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- Item Count Plan ID
- fee amount
- threshold value

The value of the fee amount element is used as the amount of the fee transaction.

ACH Fees

Use this charge method to charge a fee for the number of payments made via ACH within a specified period. If a date is not supplied, the previous period of the current GL date is used as the payment counting period.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- fee amount

The amount of the fee transaction is calculated as the number of ACH payments multiplied by the fee amount.

Wire Fees

Use this charge method to charge a fee for the number of payments made via Wire transfer within a specified period. If a date is not supplied, the previous period of the current GL date is used as the payment counting period.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- fee amount

The amount of the fee transaction is calculated as the number of Wire payments multiplied by the fee amount.

Batch Header Fees

This charge method is used to charge a fee for the number of deposits (deposit batch headers) received from the entity within a specified period. Batch Headers are counted on the incoming transaction files.

These are the elements used to apply this charge type:

- elements common to all fees: charge method, frequency, start value for the Next Generate Date, entity ID, TID, MAS code, fee currency
- fee amount (per Batch Header)

The amount of the fee transaction is calculated as the number of Batch Headers multiplied by the fee amount per Batch Header.

Transaction Posting

In Transaction Posting, the priced activity transaction and fee transactions are posted to the various merchant entity accounts.

The posting flow is as follows (Figure 13 on page 117):

1. Settlement Amount Determination

   When a transaction such as an activity or fee is received for posting, the process first determines if the transaction has any amounts to be settled by checking the settlement flag in the Settlement Plan.

   - If the settlement flag is set to Yes, the process posts the sales amount minus the credit amount.
   - If the settlement flag is set to N, the process posts an amount of zero.

2. Posting Account Determination

   An Account Posting Plan is created by entering a main record in ACCT_POST_PLAN_MASTER and the transaction level detail in ACCT_POSTING_PLAN.

   Account posting plan must be assigned to the ACQ_ENTITY record in the field ACQ_ENTITY.acct_ppplan_id. This allows an account posting to be shared across entities within the same institution. When shared, entity account post is done using an account posting type. An account posting type is used to reference a particular category of entity account setup for a merchant, and is an attribute of the record set-up in ACCT_POSTING_PLAN.

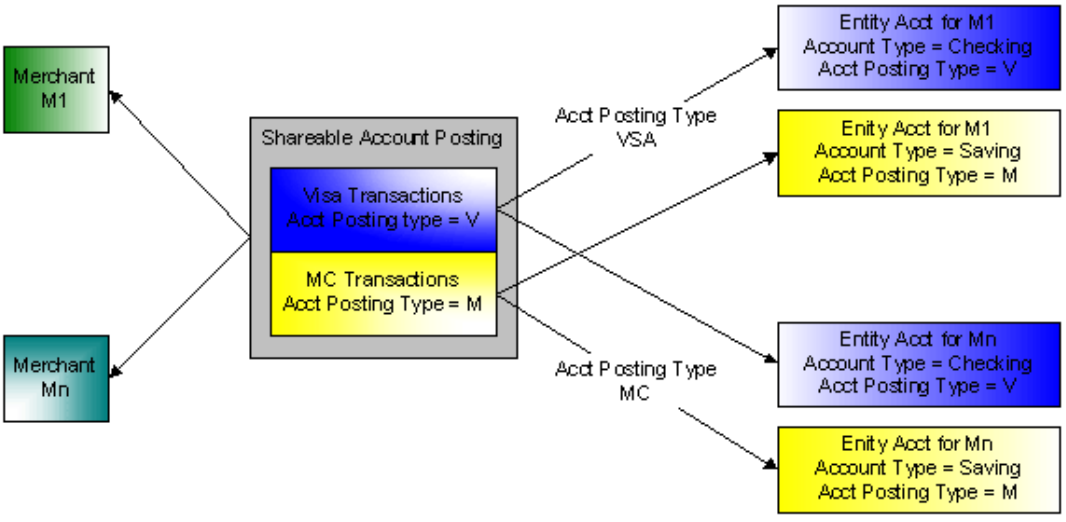


Figure 11 Account Posting Plan Business Model

In the Account Posting Plan Business Model example illustrated above:

Several merchants use the same Account Posting Plan since:

- Visa transactions will be posted to an entity account of type V (Visa). The actual bank account number is specific to the individual merchant.
- MasterCard transactions will be posted to an entity account of type M (MasterCard). the actual bank account number is specific to the individual merchant.

Account Posting Plan has the following features:

a. An assigned account posting plan type
   - General:- can be assigned to one or more merchant entity.
   - Exclusive:- can be assigned only to one merchant entity.

b. Setup of posting either to specific entity account or a user defined account posting type.
   - Specific entity account applies only to account posting plan of type, “exclusive”.
   - Account posting type applies only to account posting plan of type, “general”.

c. Assignment of PAYMENT_METHOD_CD, ACCT_ACCUM_METHOD, PAYMENT_CYCLE per transaction record. This now used to determine the entity account to post to and allows accumulation of multiple payment methods per entity account. This feature removes the necessity to setup an entity account per payment method.

NOTE:	Account Posting Plan determination of Activity Transactions is performed in the Validation component while Account Posting Plan determination of fees is performed in the Transaction Posting component.

Posting uses the Account Posting Plan to determine the Entity Account to which the transaction is to be posted. If an Account Posting Plan is not found, then the process sends the transaction to the Transaction Error Log for manual data correction and enrichment and requires the operator to resubmit the transaction to the Transaction Posting.
See “Resubmit Error Transactions” on page 93 for details.

Each merchant entity can be associated with one or more accounts depending on how settlement is required for each transaction type and currency. There are three types of accounts:

- Deposit—a Direct Deposit Account (DDA) to which payment is deposited.
- Reserve—a DDA to which amounts held as a reserve against the merchant entity is deposited.
- AR—an account receivable account, against which invoices are generated.

The Account Posting Plan allows transactions to be settled in various ways. The following lists some examples:

- transactions of different currencies settled to a single account
- transactions of different types to same or different account. (e.g., payments to one account and fees to a different account.
- any combination of transactions and currency to one account

Each account has a payment method, which determines the way the funds are transferred for settlement (e.g., wire, invoice). Each entity account must have a valid payment method, otherwise the format of the output file is unknown.

3. Settlement Currency Determination.

   The entity account is used to determine the currency in which the transaction amount is to be settled. If the transaction currency and the entity account currency is not the same, the amount is converted to the entity account's currency using the Currency Conversion Plan.
   See "Currency Conversion" on page 161 for details.

4. Settlement Amounts Accumulation

   During Transaction Posting, transactions are posted either as payment or fee transactions to the entity accounts. This step performs the actual transaction posting, when transactions are actually linked to an account accumulator, which represents a bucket for the entity account.

   The transaction amount is accumulated in account accumulators (ACCT_ACCUM_DET table) based on the settlement date, and if the transaction is to be settled immediately, posting is done to the Account Accumulators having the settlement date equal to the current date (activity date).

   Although the accumulation is always done per settlement date, individual accumulators can be defined for an entity account via accumulation methods (ACCT_ACCUM_METHOD field in the ENTITY_ACCT table). The following account accumulation methods are supported:

D	Accumulate per day all transactions and generate one entry in the payment/fee file.
B	accumulate per batch all transactions within a batch to a single detail and generate one entry per batch in the payment/fee file.
T	accumulate per TID all transactions within a batch to separate detail and generate one entry per batch per TID to the payment/fee file.
C	accumulate per card scheme all transactions within a batch to separate detail and generate one entry per card scheme per batch.
Z	accumulate per batch + TID.
P	accumulate per payment term.

S	accumulate per single transaction.
F	accumulate per file.
O	accumulate per Original File ID Clearing system generates new File ID for resubmit suspend clearing transactions. This accumulation method allows to generate one payment for the same Original File ID, thus preserving the file integrity of the submitted file.
R	accumulate per Original Batch ID. Clearing system generates new Batch ID for resubmit suspend clearing transactions. This accumulation method allows to generate one payment for the same Original Batch ID, thus preserving the batch integrity of the submitted batch.
E	accumulate per Entity ID.
X	accumulate per file and Entity ID.
Y	accumulate per batch and Entity ID

To use various accumulation methods, define multiple entity accounts, one for each accumulation method you plan to use. All accounts should have the same DDA information, but each one would be used for one accumulation method.

For example, to post each transaction (each chargeback or each adjustment) as a separate record in the ACH file, define an entity account with the same DDA information as the other accounts, but with the accumulation method S (accumulate per single transaction).

In the example presented in the following figure, the entity has several accounts defined, and the account accumulators are defined per transaction ID.

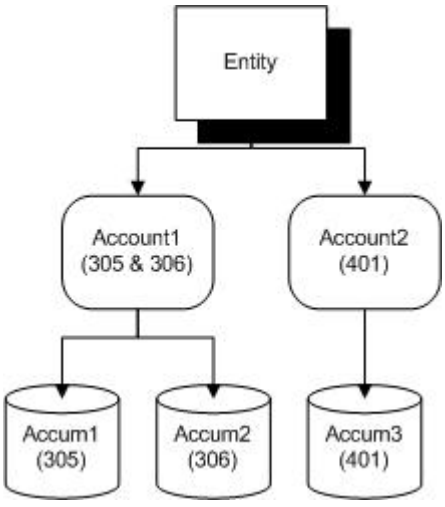


Figure 12 Accumulators

Accumulators are read by the Payment process for new accumulated amounts through the course of daily processing to create payment files, and each accumulator represents a record in the payment file.

5. GL Account Posting

   For revenue recognition and other accounting purposes, amounts are posted to the GL account accumulator for the current date (that is, the activity date from the incoming file). The GL account to which the account entries should be passed are contained in the appropriate Settlement Plan record.

NOTE:	If the settlement flag in the Settlement Plan record is set to No, no GL posting is performed. However, if the TID settlement method indicates that the transaction is a credit, GL posting is still performed.

6. Batch Summary Update

   A batch summary record is created for each batch of settled transactions. The Batch Summary accumulates information used for batch balancing, which ensures that transactions are not suspended during Validation, Pricing, or Posting without informing the user (i.e., logging to the event log table). The information used for the batch summary is saved in the BATCH_SUMMARY table and includes the total number of transactions in the batch (End Record Count field) and the batch total (Posted GL Amount field).

   To verify whether a batch is balanced, these values are compared with the similar values accumulated by the Validation process. If the values match, the batch is balanced.

   A change in the Batch ID indicates the end of one batch and the start of another, thereby triggering a batch balancing process. The Batch ID is created during Validation and is a system-generated unique number to identify a batch.

7. Logging to the Transaction Log

   The Transaction Log contains all activity transactions and fee transactions generated through Pricing. Within the on-line processing, Posting writes to the Transaction Log, exclusively. Reports may be generated based on the Transaction Log.

The following figure presents the Posting processing flow:

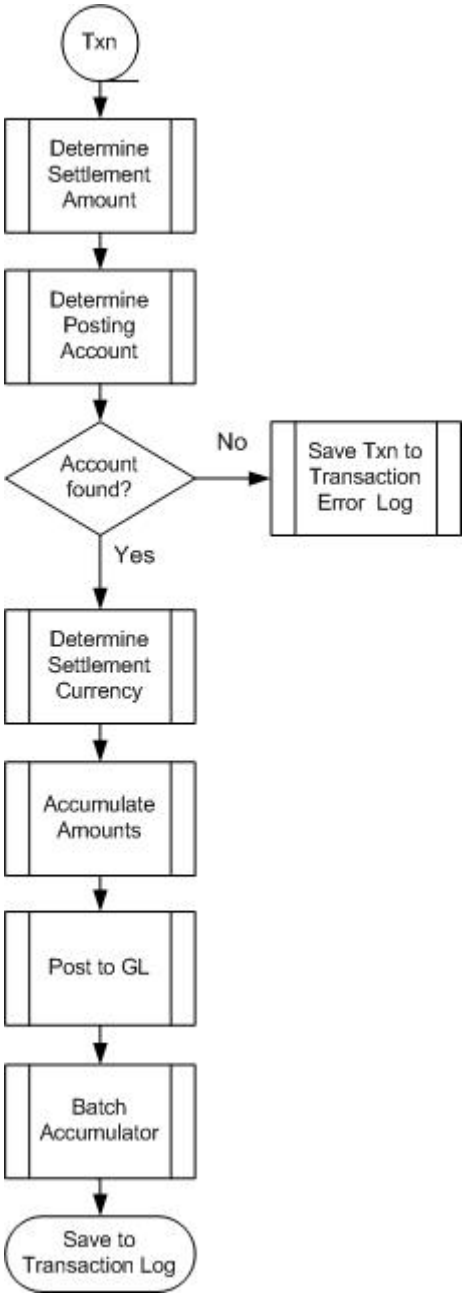


Figure 13 Transaction Posting—Logical Processing Flow

Expected Merchant's Transmission

The Expected Merchant's Transmission is a list of file types that the acquirer expects for a particular date-time. *IST/MAS* scans the File Log to see if the file has already arrived. If the file has arrived, processing continues as normal. If the file is not found, an error message displays the number of files received out of the number of expected files.

Scaling MAS Daily Processes

The following figure presents the Daily process with OVM Distributor:

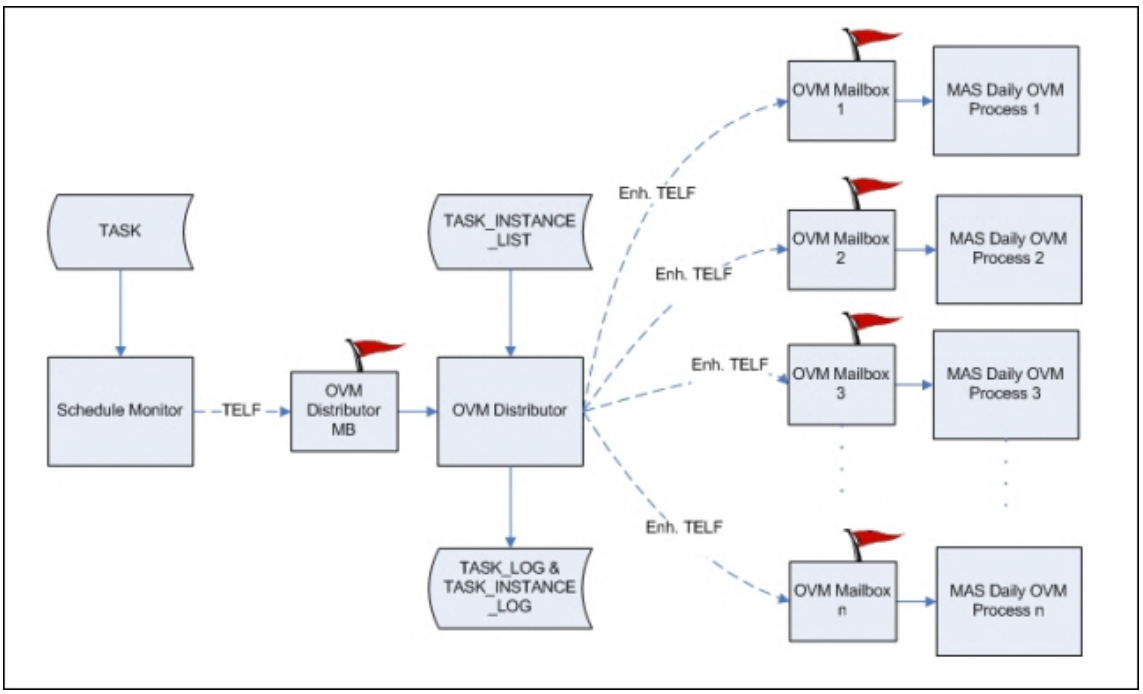


Figure 14 Daily process with OVM Distributor

The following figure presents the Daily process with OVM Distributor and Control OVM:

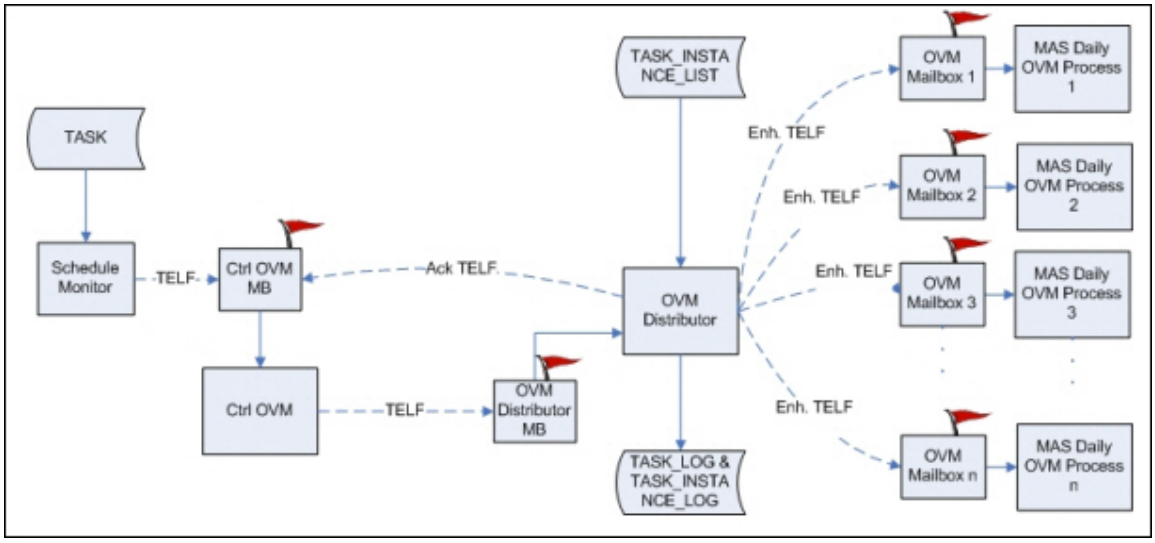


Figure 15 Daily process with OVM Distributor and Control OVM

Modules

Schedule Monitor

The schedule monitor (sch_mon) is the utility that is used in IST for task scheduling. The entries are configured in the TASK table of the database and the sch_mon polls table continuously. When a task has to be executed, a TELF message is created and forwarded to a Mailbox.

Control OVM

The control OVM provides the infrastructure for execution of complex tasks, which will execute based on some business rules. It acts by controlling the execution of tasks from scheduler to the OVM distributor. This is a normal IST OVM process run by the application manager. The code is written in TML (Transaction Modeling Language) and hence gives flexibility to customize the functionality of the control OVM.

OVM Distributor

The OVM distributor splits a single task into multiple sub-tasks and forwards them to different OVM instances. It monitors the overall status and sends feedback to the control OVM or updates status in the TASK_LOG table.

Application Manager

The Application Manager (APM) manages the run-levels (initialization, execution, and termination sequences) of all IST/Foundation application processes.

Daily Process Modules

The IST/MAS daily processes like build item to accumulate, transaction counting, Non-activity fees, Rate Reassessment, Payment, Out bounder, MAS Summary are required to process higher number of records or transactions in the current processing window. This has been achieved by splitting each process so that they process a different dataset, working in parallel they can achieve the required throughput.

The following process modules in IST/MAS are scaled. For each of these processes, the bulk of Transactions/Records to be processed are split and fed into individual instances of these processes based on a key field which is termed as dataset splitting key.

Build Item to Accum

Dataset splitting key : Item_cnt_plan_id

Transaction Counting

Dataset splitting key : Entity_Id

The build item to accumulate should have completed successfully for this task to start.

Non-activity fees

Dataset splitting key: Non_act_fee_pkg_id

The transaction counting should have completed successfully for this task to start.

Rate Reassessment

Dataset splitting key: Fee_pkg_id

The transaction counting should have completed successfully for this task to start.

Payment

Dataset splitting key: Entity_Acct_Id

The cycle balancing should have completed successfully for this task to start.

Outbounder

Dataset splitting key: Entity_Id

MAS Summary

Dataset splitting key: Entity_Id

OVM Distributor

Implementation

The OVM Distributor is a module that is used for splitting and distributing tasks scheduled to multiple instances of the OVM. This is used to scale the processing by parallel processing of a single task by multiple OVM processes. This module is plugged between the scheduler and OVM processes in question.

As many OVM processes as supported by the available resources (CPU, memory etc) can be instantiated by changing the apm.src file. Each individual command/task can be split into several instances by configuration in the TASK_INSTANCE_LOG. Various parameters like the callback function required for splitting the task and number of instances can be configured. This configuration can be done after benchmarking the hardware and depending on the load (number records/transactions) for each task.

The existing scheduler is configured (changing istparam.cfg) to send the TASK messages to the OVM distributor mailbox, rather than the OVM mailbox. The OVM distributor creates as many copies of the task message as configured for that command number. It then enhances the command parameter field of the TELF message with new values based on the dataset splitting API, which is again configured. These new enhanced TELF messages are sent to the next idle OVM process.

The OVM distributor ensures that updating the TASK_LOG table with the coordinated status does not impact the existing logging. Individual sub-task status is updated in TASK_INSTANCE_LOG.

Task Scheduling using OVM Distributor

The following figure presents the Task Scheduling using OVM Distributor:

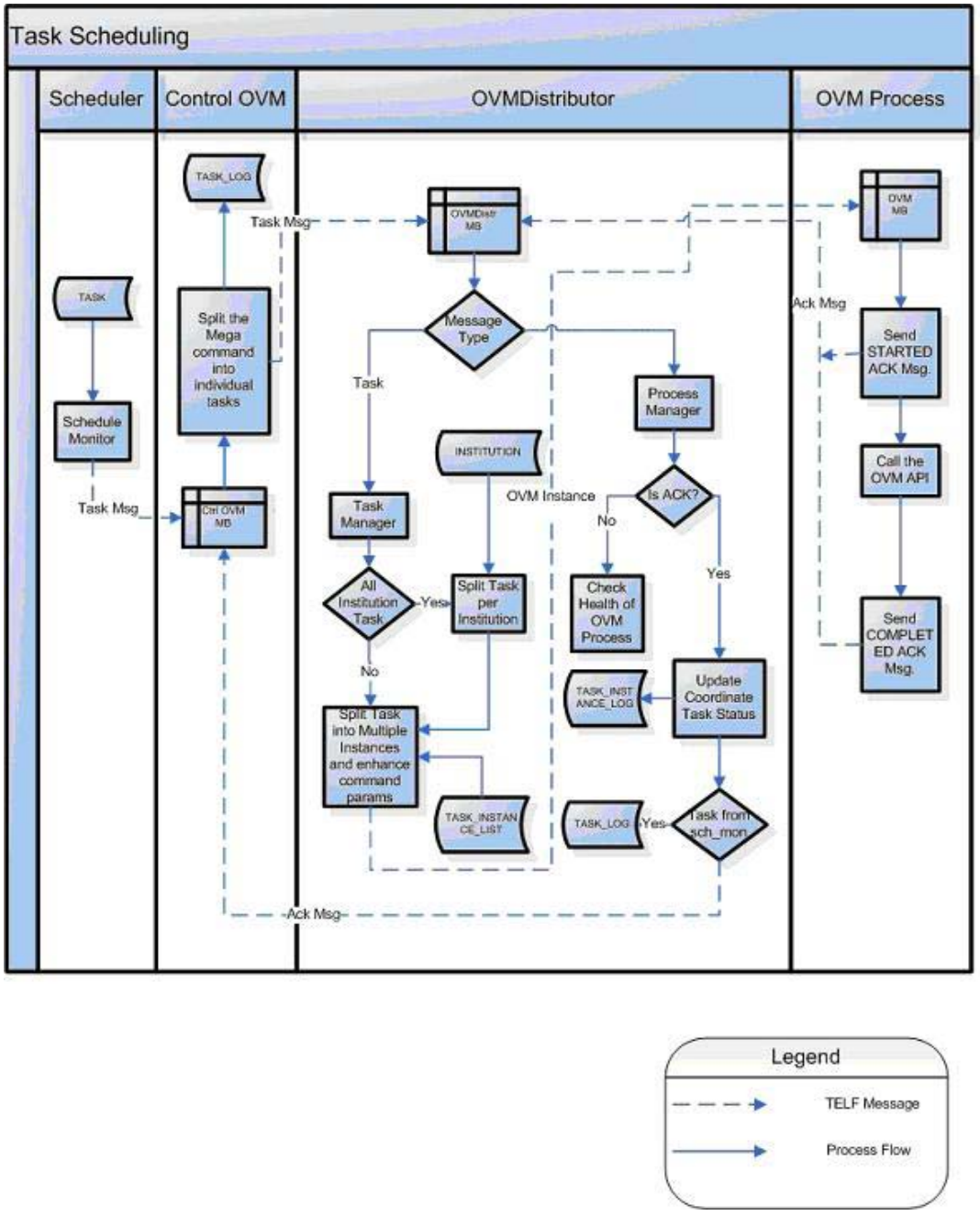


Figure 16 Task Scheduling

OVM TML Script

The MAS Daily OVM process interprets a script written in TML (Transaction Markup Language) for the business processing. This script is modified (mas_daily.scr) to use the OVM Distributor. The following is the list of changes

Acknowledgement Messages

The MAS daily processes must send acknowledgements messages to the OVM distributor mailbox for the process flow to be completed.

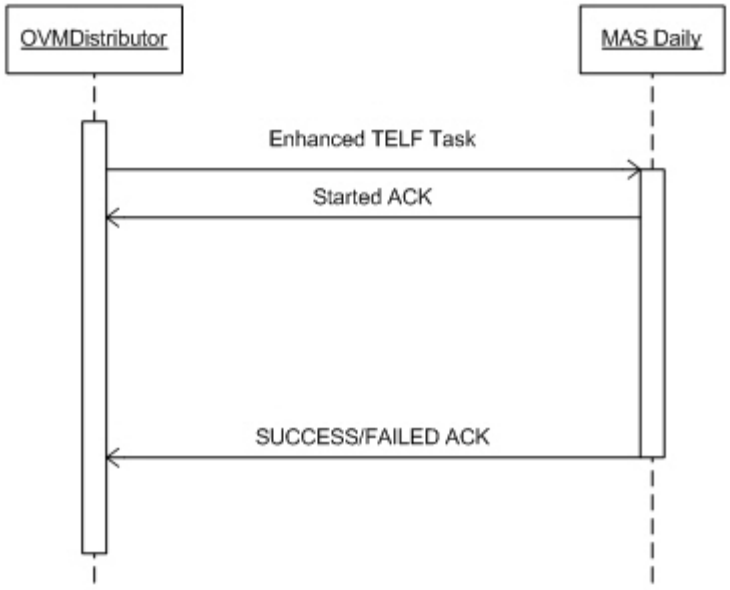


Figure 17 Acknowledgement Messages

Below is a sample from mas_daily.scr:

```
...} else if ((cmd_str == "107") || (cmd_str == "157")) {
daily.send_ack("STARTED");
prn.println("Transaction Counting Request");
int nRet = daily.transaction_counting(nAdm);
if( nRet == 0){
daily.send_ack("SUCCESS");
} else {
daily.send_ack("FAILED");
 }
} else if …
```

Control OVM

TML script and the corresponding C++ shared libraries must be created for the control OVM. There should be a provision for storing the incoming messages in an internal data structure because with this OVM mailbox will receive both task messages from schedule monitor and acknowledgement messages from the OVM distributor. Acknowledgement messages should be handled in a suitable manner.

Scheduler routes the task execution request based on mas.cmd_mailbox parameter configured in istparam.cfg. To route task execution requests to MAS Control OVM, mas.cmd_mailbox should be set to MAS_CNTRL_OVM. Control OVM will in turn route task execution requests to OVMDistMB. OVM Distributor will distribute the tasks to its target mailbox as configured in OVMDistr.target_mbox.

Below are the entries in istparam.cfg:

Entry	Description	Value
mas.cmd_mailbox	Command mailbox for the MAS usage is MAS_CNTRL_OVM. The scheduler will route the entire task execution request to this mailbox.	MAS_CNTRL_OVM
OVMDistr.target_mbox	OVMDistributor receives task command messages from scheduler and it routes the messages to the set of OVM processes that are running for one distributor mailbox.	MAS_DAILY (By default, CRD_CORE)
mas.ovmdistr_hash_func	Hash function is used by OVM Distributor for data splitting key.	**Valid values**: ora_hash mod

Control OVM is implemented as an OVM process. The following mas control OVM entry are added to apm.src:

OVM Process Name	Other Entries							OVM config file name	Distribution mailbox name		No. of OVM processes to start (Insatnces)	
mas_cntrl_ovm	test	Localhost	Null	oasisvm	Null	null	OSITE_ROOT	mas_cntrl_ovm.cfg	MAS_CNTRL_OVM	null	1	0

Business Logic:

```
…
if (cmd_str == '301')
{
    daily.stop_fr(nAdm.getChild(SCH_DT).getLong());
    daily.waitQueueEmpty();

      //create cmd for cycle_balancing and send to OVMDistMB
nAdm.getChild(SCH_DATA).putString(nAdm.getChild(SCH_INSTITUTION).
getString());
    nAdm.getChild(SCH_CYCLE).putString("N");
    nAdm.getChild(SCH_CMD_NBR).putString("118");
    msg.sendMessage("OVMDistMB", "MAS_CNTRL_OVM");
    status = get_ack(ctx, prn);
   if (status != "SUCCESSFUL")
   {
         prn.println("Cycle Balancing failed");
   }

      //create cmd for eod_bal_report and send to OVMDistMB
nAdm.getChild(SCH_CYCLE).putString("Y");
    nAdm.getChild(SCH_CMD_NBR).putString("130");
    msg.sendMessage("OVMDistMB", "MAS_CNTRL_OVM");
    status = get_ack(ctx, prn);
   if (status != "SUCCESSFUL")
   {
         prn.println("EOD Balance Report failed");
   }

    if (pay_send_bal(ctx, msg, nAdm, prn, daily, "ach", "A", "Y", "00",
"Y") == 0)
   {
         prn.println("Payment File Generation failed");
   }
    daily.start_fr(nAdm.getChild(SCH_DT).getLong());

}
…
```

Payment Processing

In transaction posting, amounts to be settled through the Payment process are accumulated in the merchant's account. This eliminates the need to accumulate payment for priced transactions from the main transaction store, which contains a very large number of transactions.

The payment cycle used to deliver payment information responds to scheduled (time-driven) or command invocation. Command invocation reschedules or overrides scheduled payment cycles.

Payment Processing Flow

The Payment processing flow is as follows (Figure 18 on page 131).

1. Account Selection

   For Payment processing, only amounts from account accumulators having the Settlement Date less than or equal to the next day. For the ACH payment method, the next day is the next processing date that is a business day. For the Wire and Manual Wire payment methods, the next day is the next GL date that is a business day.

2. Payment Effective Date

   Payment is controlled by the bank's calendar so that payment processing is sensitive to the day of week and the business calendar. The Calendar is used to determine if a given date, which may be either the GL date or the processing date, is a valid business day.

   If the payment processing date is a business day, the Payment Effective date is set to be the payment processing date. If the payment processing date is not a business day, the Payment Effective date is set to be the next business day.

   See "Payment Effective Dates" on page 133 for details on ACH payment effective dating.

3. Merchant Status Verification

   Before a record is created in the payment file, the system verifies in the Acquiring Entity table the status of the merchant. The Stop Payment and Stop Fee flags are verified.

   If the Stop Payment flag is set to Y, the payment is not processed. If the Stop Fee flag is set to Y, the fee is not processed. If both are set to Y, no payment record is created in the payment file.

   To re-process the skipped payments and fees, the corresponding flag has to be set to N via GUI.

4. Payment File Creation

   IST/MAS produces payment files with each payment method. Payment files include the original activity transactions sent to the system from various source systems and manual adjustments entered via GUI. If fees are deducted directly from payments, the payment files include fee transactions also.

The following payment files are produced based on the payment method configured for the account.

- ACH—Automatically generated ACH payment files.
  See "ACH Payment" on page 132 for details.
- Wire—Automatically generated Wire payment files.
  See "Wire Payment" on page 135 for details.
- Manual Wire—Automatically generated Manual Wire payment files.
  See "Manual Wire Payment" on page 137 for details.

NOTE:	Another payment option displayed by the system is Invoice, which is not a payment method in itself but a fee invoicing method. When this option is selected, the system creates Invoice files from which invoices are generated. See "Account Receivable (AR) Processing" on page 137 for details.

In addition to the pre-defined payment files ACH, Wire, and Manual Wire, IST/MAS allows defining any custom payment files by setting up MAS Payment Method and MAS File Type tables, and defining the payment file layout using Universal Agent (UA) configuration files.

Scaling on Payment Process

To achieve performance in payment process, the payment (any payment) task can be split up by increasing the task instance. Since task instance is increased, it should be supported by equal MAS Daily process. So based on dataset splitting key, the payment records will be split & fed to different processes which results in multiple files being created for the single payment task execution. Since payment records are split and processed in each MAS Daily process in parallel, the Universal Agent (UA) should also be increased to match the MAS Daily process to generate the payment files in parallel. This ensures a overall performance improvement.

NOTE:	To do task split up, configuration has to be done in task_instance_list table to specify number of task instances and apm.src to specify number of Daily process and UA.

MAS Payment Method table is setup as follows:

Pymt Type	Pymt Method CD	File Type	Roll Up Flag	No Neg	Create Inv	Output Rollup Rec
A	A	52	N	N	N	N
W	W	53	Y	Y	N	N
M	M	56	Y	Y	N	N

One payment type can have several payment method codes. This allows variation of payment effective date calculation or bank calendar for the same payment type.

Valid values for Roll Up Flag are Y, N, O.

- 'N' indicates generate payment record according to the defined accumulation method.
- 'Y' indicates that payment records having the same payment effective date will be combine into one payment record. Payment will still be according to accumulation method.
- 'O' (owner) indicates that payment records having the same Entity Acct ID and Payment Effective Date will be combine into one payment record. Payment will not be according to the defined accumulation method, instead payment will be one payment per entity acct id. This is useful for store level rounding that allows the sum of amounts reported to each merchant location matches the amount in the payment file. This is done by setting accumulation method to 'E' (Entity ID), and set roll-up flag to 'O'.

No Neg is a 'Y' or 'N' flag that indicates whether or not to allow negative payments.

Create Inv is a 'Y' or 'N' flag that indicates whether or not an invoice will be generated.

Valid values for Output Rollup Rec are 'Y', 'N', 'D', 'C'.

- 'Y' means output roll-up payment records as specifying transaction records to payment file.
- 'N' means do not output roll-up payment records to payment file.
  'D' means output roll-up payment records if one of the payment records is a debit payment.
- 'C' means output roll-up payment records if all records are credit payment.

MAS File Type table is setup as follows. Each file type is mapped to a UA adapter which contains the file layout description of the payment file.

File Type	UA Adapter
52	a4
54	a5
56	a8

5. Logging to Payment Log

   The Payment Log provides an audit trail for the Payment Process. Once a payment is generated to a payment file, a Payment Log record is created.

   From the Payment Log table, a Payment Log report file is generated to contain all payment records. The Payment Log file is saved in the directory given by the mas.58_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file. See "File Formats" on page 171 for details on the file name and format.

6. Outbounder File Creation

   Based on the Payment Log records, the process takes the appropriate detail transactions from the MAS Transaction Log and creates the Outbounder Report file. The file is saved in the directory given by the mas.54_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
   See "File Formats" on page 171 for details on the file name and format.

7. GL Posting

   During Payment Processing, exported payment amounts are also accounted for in terms of GL accounting. Each payment amount is accumulated within specific GL accounts. The GL accounts to which the account entries are passed are contained in the respective Entity Account record.

The following figure presents the Payment processing flow:

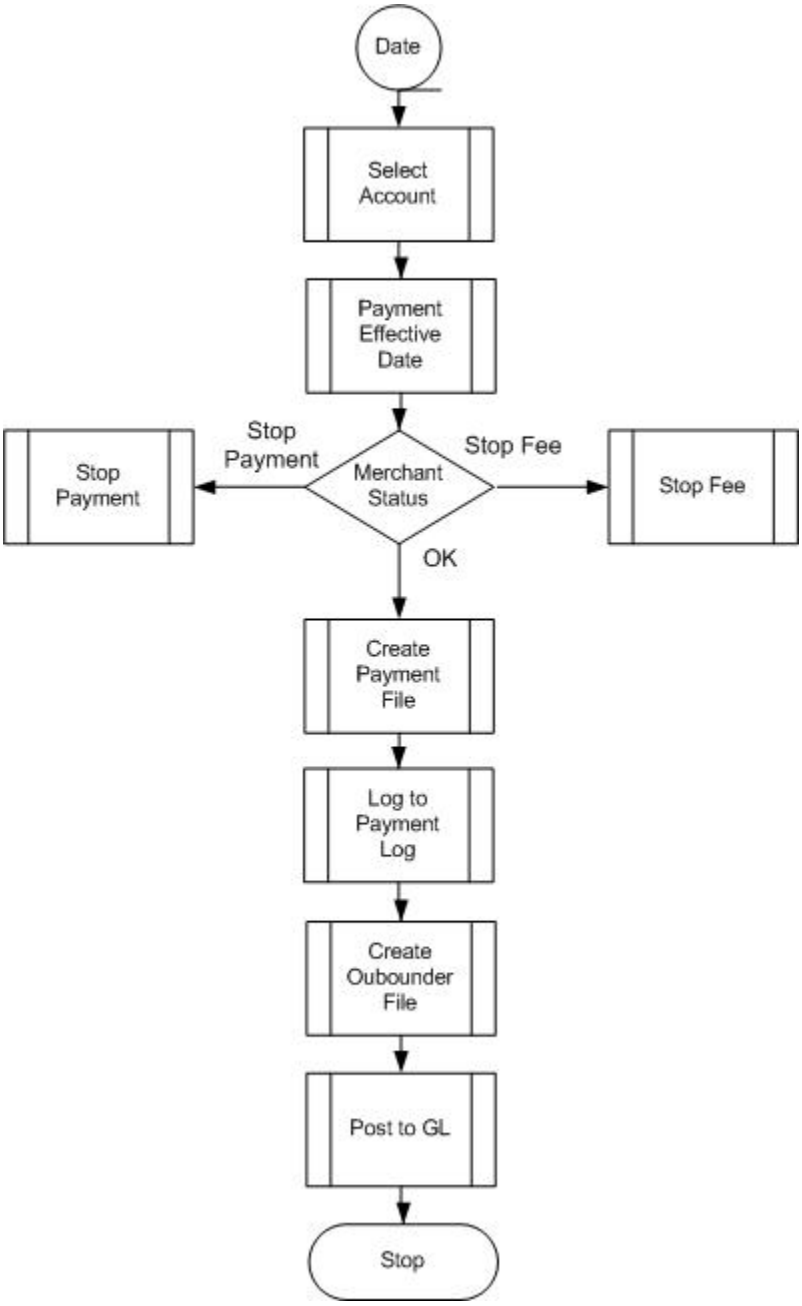


Figure 18 Payment—Logical Processing Flow

ACH Payment

ACH payment files contain payment records generated for all accounts of all entities, and payments records are calculated totals of a single account accumulator.

Multiple account accumulators result in multiple ACH Payments as shown in the following diagram:

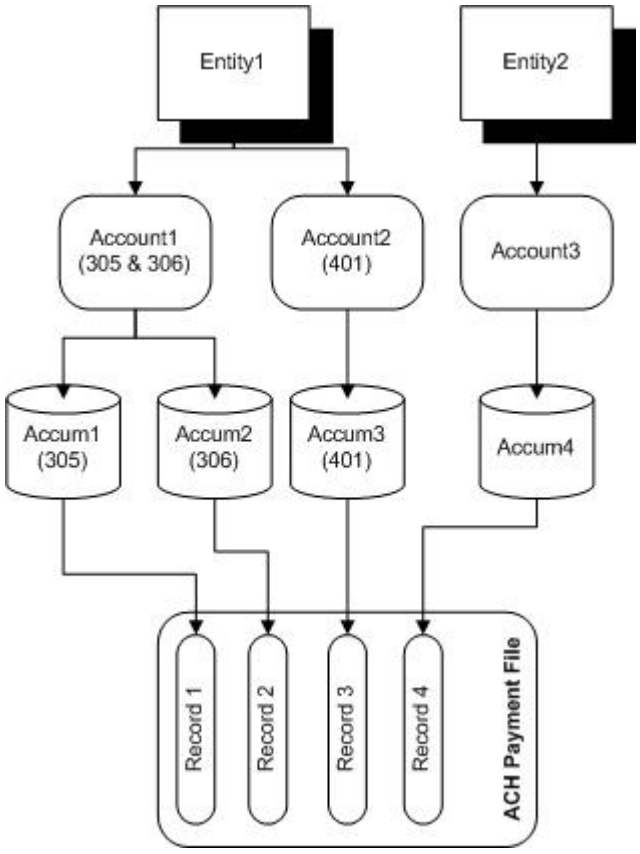


Figure 19 ACH Payment—Payment File Generation

ACH payment values may be positive, indicating a credit to a merchant account, or negative, indicating a debit from the merchant's account. Payments may be negative when fees are deducted directly from payments or when payment adjustments are required.

In the detail record, the type of account is considered. Thus, the system distinguishes between debit and credit amounts posted to savings or checking accounts, for example. The type of account is saved in the ACCT_TYPE field from the ENTITY_ACCT table.

ACH payment records are saved in the ACH payment file. Note that records are organized in batches, and any ACH file contains records from one and only one batch of records.

The ACH payment file is saved in the directory given by the mas.52_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

Payment Effective Dates

Based on the delay_factor in the PAYMENT_METHOD_CD field from the ACCT_POSTING_PLAN table, different payment effective dates on the ACH file can be supported. These are values currently supported for delay factor:

-1	settlement date in the ACCT_ACCUM_DET table.
>= 0	add delay_factor as business days to selection date.

The selection date can have one of the following values based on the PAY_USE_SYS and PAY_TODAY fields in the FIELD_VALUE_CTRL table:

PAY_USE_SYS	PAY_TODAY	Selection date[1] (earliest business day of)
Y	Y	System date
Y		System date + 1
	Y	GL date
		GL date + 1

1. The selection date depends on the calendar used for the payment method. Via MAS_PAYMENT_METHOD, payment methods can be set up to use different calendars, or they can be set up to share the same calendar.

Splitting ACH Files

ACH file generation can be customized to split ACH files by Payment Cycle.
See "Payment Cycle" on page 134.

Furthermore, the ACH file generation can be customized to group payment records in one file based on defined Trans Routing Numbers and a default payment cycle. This way, the activity of certain acquiring entities can be saved in one ACH file (e.g., all on-us Trans Routing Numbers).
See "Transit Routing Number" on page 134.

Payment Cycle

The system generates separate ACH files based on the PAYMENT_TERM field in the ACCT_ACCUM_DET table, information which is set by the Settlement Plan.

For example, *IST/MAS* can generate two ACH payment files for the same GL date. One payment file can be for daily deposits and discounts and one for monthly (or cycle) fees.

To generate two separate ACH files, set up two entity accounts for the same entity. The two accounts may have the same DDA information, but they must be set with different payment cycles, so that they are processed in two different ACH files during the day. For this entity, all daily deposit and discount transactions use one account, while fee transactions use the other account. Posting to the appropriate entity account is accomplished through the Account Posting Plan.

Transit Routing Number

If required, *IST/MAS* groups payment records in one file based on defined Trans Routing Numbers and a default payment cycle. The grouping does not affect the ACH file splitting based on payment cycle. It only defines which payment cycle (i.e., file) will contain the payment records of selected acquiring entities.

The selection is done by associating their Transit Routing Numbers with a defined default payment cycle. All Transit Routing Numbers assigned to the default payment cycle are saved to the corresponding ACH file. For example, all on-us Trans Routing Numbers can be consolidated into one file.

All payment records that may have assigned the default payment cycle but their Transit Routing Number is not associated to the default payment cycle are saved to another ACH file. The file is defined by associating an alternative payment cycle to the default cycle.

NOTE:	The payment effective dates within the two files is not necessarily the same.

To configure this feature, create the following records in the FIELD_VALUE_CTRL table:

1. Set ON the feature.

   Create one TRN_PC_SETUP_REQ record and set it to Y to indicate that the feature is desired.

   The record is as follows:

FIELD_NAME	TRN_PC_SETUP_REQ
FIELD_VALUE	Y/N

2. Define the default cycle and its Transit Routing Numbers.

   Create as many TRN_PAY_DEF_CYCLE records as required to indicate all the transit routing numbers associated with the default payment cycle.

   Each record is as follows:

FIELD_NAME	TRN_PAY_DEF_CYCLE
FIELD_VALUE1	Default payment cycle (e.g., 0)
FIELD_VALUE2	routing number (e.g., 668)

3. Define the alternative payment cycle for the default payment cycle.

   Only one record is required to define the alternative payment cycle.

   The record is as follows:

FIELD_NAME	TRN_PAY_ALT_CYCLE
FIELD_VALUE1	Default payment cycle (e.g., 0)
FIELD_VALUE2	Alternative payment cycle (e.g., 1)

Wire Payment

Wire payment files contain payment records generated for all accounts of all entities, and payments records are calculated totals of all account accumulator associated with an entity. Since payment amounts can be positive or negative and negative wire payments are not permitted, all accounts for an entity are accumulated into one payment record.

If the net payable total of all account accumulators is still negative, payment is skipped for the account. Negative amounts are accumulated in the following payment cycle and are re-evaluated to determine if a positive payment can be made.

Multiple account accumulators result in a single Wire payment as shown in the following diagram.

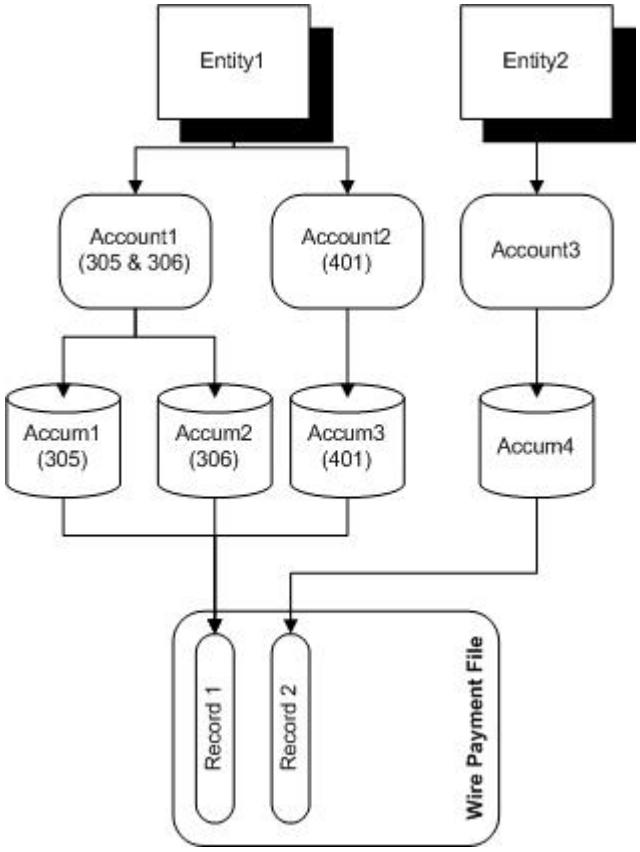


Figure 20 Wire & Manual Wire Payment—Payment File Generation

In this case, account accumulators and the corresponding payment records are linked through the actual payment sequence number, and the LINK_PAY_SEQ_FLAG is set. Because one payment record is created from more than one account accumulator (each with its own payment sequence number that links this accumulator with the detail transactions), all account accumulators used for the payment record are updated with the actual payment sequence number defined in the payment record.

If the Stop payment flag is set for a merchant but fees need to be charged, the Wire payment methods cannot be used to charge automatically the fees since negative amounts are not permitted for this payment methods. To charge fees, the payment method of the merchant's account must be changed to Invoice and the fees are charged through the normal invoicing process.
See "Account Receivable (AR) Processing" on page 137 for details on invoicing.

Wire payment records are recorded in the Wire payment file, which is saved in the directory given by the mas.53_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

Manual Wire Payment

Manual Wire Payment is identical to Wire Payment, except that input of additional information, like DDA account, may be required to help the financial institution process the wire payment. (Figure 20 on page 136)

If the Stop payment flag is set for a merchant but fees need to be charged, the Manual Wire payment method cannot be used to charge automatically the fees since negative amounts are not permitted for this payment method. To charge fees, the payment method of the merchant's account must be changed to Invoice and the fees are charged through the normal invoicing process.
See "Account Receivable (AR) Processing" on page 137 for details on invoicing.

Manual Wire payment records are recorded in the Manual Wire payment file, which is saved in the directory given by the mas.56_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

Account Receivable (AR) Processing

Invoice is considered a 'payment' method used exclusively for fees. To use invoice to charge for fees, an entity must have an entity account for the payment method 'I' (invoice), and fees are routed to the account specified by the Account Posting Plan.

Amounts from fee transactions that should be invoiced are posted by the Posting process to the entity accounts with the payment method Invoice. Invoiced fees are fees that are not settled directly to the merchant's DDA account, meaning that the fees are not subtracted directly from the payment amount.

Account Receivable Processing Flow

The following processing flow is followed during AR processing:

1. Invoices are generated to charge merchants for the processing fees incurred.

   The invoiced amount is the exact amount from the account accumulators, and it can be either credit or debit. There are two ways of generating invoices:

   - Debit and credit amounts generate invoices; however, if the invoice amount is negative (credit), the invoice status is set to open, and the invoice must be manually closed via GUI.
   - Debit amounts are charged by invoice, and credit amounts are deducted from the subsequent invoices as unapplied amounts.
     See "Invoice Generation (Unapplied Amounts Method)" on page 139 for details.

The Invoice file is saved in the directory given by the mas.55_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

NOTE:	Before a record is created in the invoice file, the system verifies in the Acquiring Entity table the status of the merchant. The Stop Payment and Stop Fee flags are verified. If they are set, payments and fees are not processed.

2. When merchants remit cheques to pay for the invoiced amounts, remittance transactions are created via GUI to clear the invoiced amounts from the system.
   See "Remittance Transactions" on page 145 for details.
3. Invoices are aged. This allows for automatic tracking of invoices and eventual write-off if the invoice is not paid within a specified amount of time. This way these amounts are no longer receivable, but a debt.
   See "Invoice Aging" on page 143 for details.

Invoice Generation (Unapplied Amounts Method)

The scheduler, at pre-defined date-times, initiates an invoice generation process, which has the following flow. Note that the most common path has been highlighted.

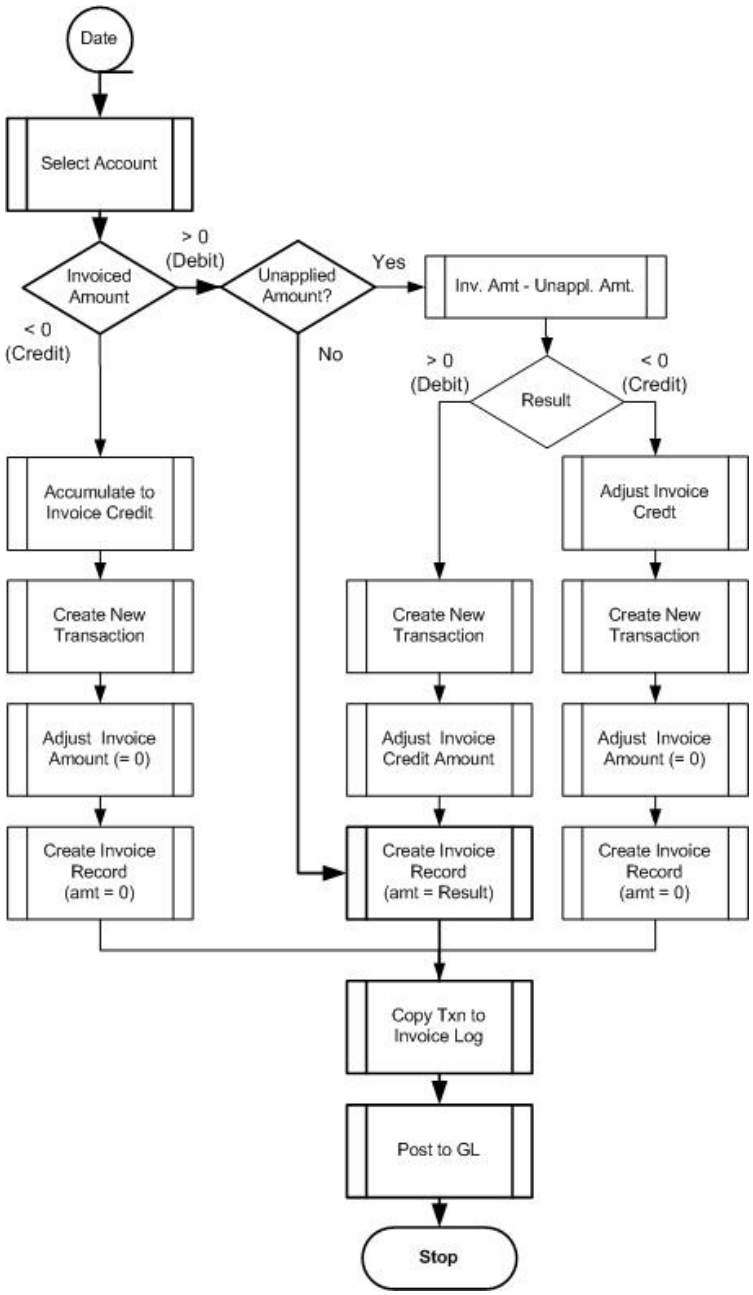


Figure 21 Invoice Generation—Logical Processing Flow

1. Account Selection

   Fees to be invoiced are accumulated in account accumulators per settlement date and payment term. *A* payment term is defined as part of the transaction Settlement Plan and can be any number of business days.

   The Invoice Generation process goes through the account accumulators and selects records with an Invoice payment method. To ensure that invoices are generated by payment terms, administrators must set up at configuration time the accumulation method to be accumulation by payment term (ACCT_ACCUM_METHOD field).

   All transactions to be included in an invoice are found and tagged with the invoice number.

   After invoices are created, new and open invoices are outputted to an Invoice File. By default, only new and overdue invoices are outputted to the Invoice File. User can specify not to generate overdue invoices in the command parameter so that only new invoices are outputted to Invoice File. To generate all open invoices including old invoices that are not yet overdue, INC_ALL_OPEN_INV in Field_value_ctrl should be set to Y.

NOTE:	By default, transaction details are not included in Invoice File. To include transaction details, INCLUDE_INV_DETAIL in Field_value_ctrl should be set to Y.

2. Amount Verification

   Invoice records are created in the invoice file for each accumulator record, regardless of the record's positive (debit), negative value (credit). However, based on the amount value, the processing is different.

3. Debit Flow (no unapplied amount)

   If the amount in the account accumulator is positive, meaning that the merchant should be charged, the unapplied invoice amount is verified. If no unapplied invoice amount exists, an invoice record is created with the value of the invoice amount.

4. Debit Flow (unapplied amount exists and Result >0)

   If there is some unapplied cash from the previous cheque payment (i.e. the merchant overpaid the last invoice) or as a result of fee calculation, the amount is deducted from the invoice amount. The Result of this operation, dictates the following processing.

a. Create New Transaction—A new transaction is created with the amount equal to the unapplied amount. The transaction ID (e.g., 502) is given by the apply_credit_inv_amt_TID parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file. This new transaction is linked to the current invoice.

b. Adjust Invoice Credit—The Invoice Credit becomes zero.

c. Create Invoice Record— A new invoice record is created with the invoice amount equal to the Result amount.

5. Debit Flow (unapplied amount exists and Result < 0)

   a. Create New Transaction—A new transaction is created with the amount equal to the unapplied amount. The transaction ID (e.g., 502) is given by the apply_credit_inv_amt_TID parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file. This new transaction is linked to the current invoice.

   b. Adjust Invoice Amount—Since negative invoices are not permitted, the invoice amount is adjusted to zero and the unapplied amount that was not covered by the Invoice Amount (i.e., the Result) will stay to be applied to the next invoice.

   c. Create Invoice Record— A new invoice record is created with the invoice amount equal to zero. In this case, a configurable status is assigned to the invoice. Since the invoice has a zero amount and can be closed, the status can be used to trace the invoice via GUI to close it. The status is configured through the *inv_zero_stat* parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file.

6. Credit Flow

   If the amount in the Account Accumulators table is negative (meaning that the merchant should be credited), the amount is accumulated to the Invoice Credit so that it can be used against the merchant's future invoices.

   a. Accumulate to Invoice Credit

   b. Create New Transaction— The amount of the transaction is equal to the amount of the account accumulator but with a positive sign, and the transaction ID (e.g., 501) is given by the adjust_credit_inv_amt_TID parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file. This new transaction is linked to the current invoice.

   c. Adjust Invoice Amount—Since negative invoices are not permitted, the invoice amount is adjusted to zero.

   d. Create Invoice Record— A new invoice record is created with the invoice amount equal to zero. In this case, a configurable status is assigned to the invoice. Since the invoice has a zero amount and can be closed, the status can be used to trace the invoice via GUI to close it. The status is configured through the *inv_zero_stat* parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file.

7. Copy Transaction Record to Transaction Invoice Table

   Using the payment sequence number from the Account Accumulators table, all individual transactions are located in the Transaction Log table, updated with the corresponding invoice number, and copied to the MAS Transaction Invoice table for inquiry purposes.

NOTE:	At the end of invoice processing, newly-generated transactions (e.g., 501 and 502) can be found in both Transaction Log and Invoice Log.

8. GL posting

   During invoice generation, exported fee amounts are also accounted for in terms of GL accounting.

Invoice Aging

Invoices are aged to track their payment and to write off unpaid invoices that are not collectable. The following figure presents the Invoice Aging processing flow according to the aging sample values provided.

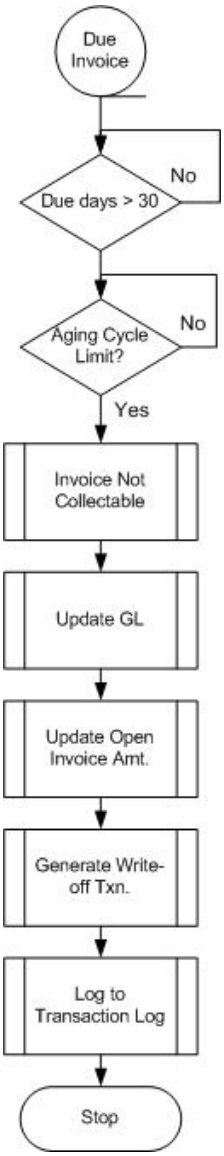


Figure 22 Invoice Aging—Logical Processing Flow

1. Aging Cycle Verification

   The Invoice Aging process is triggered by the invoice due date determined by the payment terms. Note that the system supports any number of business days offset to be configured as a payment term, these are only examples:

   a. Immediate (payment on receipt)

   b. Net 30, 45, 60 and 90 days

   c. Cash on Delivery (COD)

   If the invoice due date passes, and the difference between the current date and the due date is less than 30 days, the aging cycle is set to 1 (meaning first aging cycle)

0 – 30 days	overdue invoice	aging cycle set to 1
30 – 60 days	overdue 30	aging cycle set to 2
60 – 90 days	overdue 60	aging cycle set to 3
….		

   Another cycle of 30 days forces the aging cycle to 2, and so on, until the aging cycle value reaches the defined limit.

   The number of days in an aging cycle is given by days_in_cycle defined in Field Value Ctrl. Field_value1 is set to the aging cycle and field_value2 is set to the number of days in the aging cycle. Different number of days can be configured for aging cycle 1, 2, 3, and so on.

2. Aging Cycle Limit Verification

   If the aging cycle limit is reached, the system starts the procedure to write off the invoice automatically. The maximum number of aging cycle is configured in max_aging_cycles defined in Field Value Ctrl.

3. Invoice Not Collectable

   If the aging cycle limit is reached, the invoice is declared not collectable. The process scans for all open invoices, partially paid invoices or reopened invoices and changes their statuses to not collectable.

4. GL Posting

   During write-off processing, invoiced amounts are also accounted for in terms of GL accounting. Each amount is accumulated within specific GL accounts defined in the respective Entity Account record.

5. Open Invoice Amount Update

   The entity account's open invoice amount is updated by deducting the invoice's unpaid amount from the open invoice amount.

6. Write-off Transaction Generation

   The system automatically generates a write-off transaction with a configured TID. This transaction indicates that the write-off procedure was performed. The write-off TID is given by the write_off_TID parameter in the *$OSITE_ROOT/cfg/istparam.cfg* configuration file.

7. Transaction Logging

   The transaction is logged in the Transaction Log and the processing is concluded.

Remittance Transactions

Remittance transactions are created from the remittance amounts received for invoices. These transactions are entered using the Remittance Entry screen in the GUI and are used to clear invoices after the invoice amounts have been fully paid by merchants. One remittance transaction must be generated for each payment (cheque) received.

NOTE:	Currency restriction! Only invoices with the same currency as the cheque currency can be paid with that cheque.

A single cheque may partially pay one invoice, completely pay one invoice, or even pay multiple invoices. However, each remittance transaction generated is always linked with the appropriate cheques and invoices.

The Remittance transactions are posted through the Transaction Posting component, but they do not follow standard posting processing. Remittance amounts are not accumulated in the account accumulators. Instead, the balance of the account is updated every time a remittance transaction is posted. When amounts are posted, the invoice status on all invoices that are paid with that cheque are changed to ‘closed’ (or ‘billed’ for partially paid invoices).
See “Invoice Status Indicators” on page 210 for details on the invoice status.

If a cheque is returned, the remittance transaction must be reversed. However, the original remittance transaction is not purged from the system, but a new transaction is created to undo the processing. Three scenarios are possible for returned cheques:

- Amount of the cheque is exactly as the invoiced amount—In this case, the status of the invoice paid with this cheque is set to Open.
- Amount of the cheque is smaller than the invoiced amount—In this case, the status of the invoice paid with this cheque is set to Partially Paid.
- Amount of the cheque is greater than the invoiced amount—In this case, the status of all invoices paid with this cheque is set to Open or Partially Paid. If the amount only covers one invoice and the difference was used as unapplied cash, the invoice status is set to Open and the unapplied cash is updated.

The remittance activity is also registered in the GL accounts.

Automated Remittance Processing

Automated Remittance processing is now done by the MAS system. Earlier, the remittance was received manually through the GUI. Now this process is replaced by accepting the remittance files directly. Here is how this is done:

- A UA adaptor will convert incoming each remittance in the file into an internal message format, and send it to a validation process.
- The validation server process each remittance transaction that has a match in INVOICE, and enrich the transaction with GL accounts from Settlement Plan TID.
- The pricing server will calculate the corresponding fees if there is any. Fees can be optionally charge to returned cheque transaction depending on the business contract.
- The posting server will post the remittance transaction to the entity account, invoice, and GL account, and log it MAS_TRANS_LOG.
- Unmatched remittances will be suspended and written to suspend table, MAS_TRANS_SUSPEND, for manual work/resubmission through the GUI.

Unmatched Remittance Processing

An incoming remittance may be suspended due to either incorrect invoicer's account number (field 2), or incorrect reference number (field 6), or both. Hence, this remittance should be suspended and logged for manual rework/submission.

There will be two ways of handling unmatched references through the:

- Remittance Form

  If unmatched remittance are handled through the Remittance Form, a new remittance transaction will be created in MAS_TRANS_REMITT, and will be picked-up by remittance process and send to validation server. The suspended remittance transactions in MAS_TRANS_SUSPEND will need to be written off by running Transaction Write-Off process.

- Suspend Remittance Form (Suspend Remittance Inquiry and Reprocess Suspend Remittance Form)

  If unmatched remittance are handled through the Suspend Remittance Form, the form will correct the invalid values in MAS_TRANS_SUSPEND, and runResubmit Suspend process to submit the edited suspended transactions to validation server for re-processing.

These unmatched remittances will be logged in MAS_TRANS_SUSPEND with predefined suspend reason code.

General Ledger (GL) Processing

NOTE:	Accounting is performed in the currency of the institution designated as the *IST/MAS* user.

Posting to the General Ledger is done in transaction posting, payment processing, and invoice processing to account, at the processing point, for all transactions that carry an amount.
See "GL Accumulation" on page 148 for details.

General Ledger Processing Flow

The GL processing is as follows:

Figure 23 General Ledger—Logical Processing Flow

1. Cycle Balance

   Transaction amounts are continuously accumulated in the GL accounts. Several times a day (cycles), the transaction processing is interrupted and a snapshot of the GL accounts is taken and balance processing is performed to assess the cash movement within the defined cycle.

   When the cycle balancing is finished, the transaction processing can resume.
   See "Cycle Balancing" on page 150 for details.

2. End of Day (EOD) Balance

   At the end of each day, the EOD balance is performed to assess the cash movement for the current day.
   See "End of Day Balancing" on page 152 for details.

3. GL Export

   At the end of each day or period, amounts in the GL account accumulators are exported in a GL File by the GL Export process.
   See "GL Export" on page 153 for details.

GL Accumulation

GL accumulation is performed during posting, payment, and account receivable processing. At the end of each day or period, amounts in the GL account accumulators are exported in a GL File by the GL Export process.

The GL accounts are defined in the GL Chart of Accounts table and assigned for accumulation through the Settlement Plan, the Account Plan, Field Value Control table, and GUI screens.

During GL accumulation, all transaction amounts are converted to the GL currency, which is defined in the Institution table.

GL Accounts

Based on the transaction type and the process handling the transaction, GL accounts are specified in the following way:

- The Settlement Plan specifies the GL accounts to use during posting for activity and fee transactions. One GL account pair (credit and debit) is defined for each transaction type.
- GUI screens are used to specify GL accounts to use during posting for adjustment and remittance transactions entered directly via GUI. One GL account pair (credit and debit) is defined for each transaction type.
- Different sets of accounts are pre-populated based on the set-up of the Entity Account GL Account List (EA_GL_ACCT_LIST), which allows definition of GL groupings. This way, different GL accounts can be defined for different scenarios (e.g., Reserve, ISO). The default GL group is automatically assigned when no other sets of GL accounts are set up.
- For suspended transactions, the suspend-in settlement plan is retrieved from Field Value Ctrl table (SUSPEND_SP_IN). The GL accounts are retrieved from the suspend-in settlement plan.
- For transactions released from suspension, the suspend-out settlement plan is from Field Value Ctrl table (SUSPEND_SP_OUT). The GL accounts are retrieved from the suspend-out settlement plan.

NOTE:	One GL account may be used for all merchants. For example, all sales transactions are accumulated into one account. However, credit and debit GL accounts must be properly configured for the GL processing to be correctly performed.

Setup Example

As per Settlement Plan configuration, transaction posting to GL is configured as follows:

Purchase Transaction

```
Debit GL Account: OUTGOING-CLEARING-SUSP[1]
Credit GL Account: MER-DEP-SETT-SUSP
```

Discount Transaction

```
Debit GL Account: MER-FEE-SETT-SUSP
Credit GL Account: DISCOUNT-REVENUE
```

Fee Transactions

```
Debit GL Account: MER-FEE-SETT-SUSP
Credit GL Account: FEE-REVENUE
```

An incoming merchant purchase transaction ($250) is charged a discount ($4.13) and an additional fee ($0.27). This would result in the following GL Chart Of Account for the institution (ID = 1000001):

gl_acct	acct_desc	last_export_dt	gl_date	amt_debit	amt_credit	master_gl_acct
discount-revenue	Discount Revenue Account	11/6/02	11/7/02	0.00	4.13*	xyzmstacct1
fee-revenue	Fee Revenue Account	11/6/02	11/7/02	0.00	0.27*	xyzmstacct1
mer-dep-sett-susp	Merchant Deposit Settlement Suspense	11/6/02	11/7/02	0.00	250.00	xyzmstacct1
mer-fee-sett-susp	Merchant Fee Settlement Suspense	11/6/02	11/7/02	4.40*	0.00	xyzmstacct1

1. Note that GL account numbers are used as descriptive as opposed to numeric values. This is for easier tracking of the logic. Also, since posting of each transaction type is to configurable pair of GL accounts (debit/credit), there is no limitation imposed to specific GL accounts or types of accounts shown in this example.

gl_acct	acct_desc	last_export_dt	gl_date	amt_debit	amt_credit	master_gl_acct
outgoing-clearing-susp	Outgoing Clearing Suspense	11/6/02	11/7/02	250.00	0.00	xyzmstacct1
ach suspense	ACH Suspense	11/6/02	11/7/02	0.00	0.00	xyzmstacct1
ach-mer-fee-susp	ACH Merchant Fee Suspense	11/6/02	11/7/02	0.00	0.00	xyzmstacct1

The accumulation step is performed during the Posting phase. That means, as the incoming transaction file is processed, all incoming transactions and their corresponding fees are accumulated in the appropriate GL accounts in the above table. Also, GL accounts debited and credited are saved with each transaction (including fees) in the transaction log.

Further, when the settlement occurs, say ACH, GL account entries such as mer_dep_sett_susp and mer-fee-sett-susp are offset by the actual payments and fees settled to the merchant by configuring those as Debit GL accounts in the Ea_gl_acct_list configuration. The same Ea_gl_acct_list configuration contains appropriate ACH Suspense accounts as Credit GL Accounts.

As part of end of day processing, the GL Chart of Account table is used to export transactions in the default GL file format. At that time, GL date is advanced and the accumulation snapshot saved in the GL Chart of Account Summary table for future reference. All accumulation totals are cleared in the GL Chart of Account table to prepare for next day's processing.

Cycle Balancing

Cycle balancing is performed to assess the cash movement within a defined cycle, which is a period of time defined through the Scheduler process.

Transaction amounts are continuously accumulated in these GL accounts, and several times a day, the transaction processing is interrupted and the Scheduler initiates the Cycle Balancing process. For each GL date there are many balancing cycles, and each cycle begins after all payments are made.

Thie Total MAS_TRANS_LOG (Y/N) parameter indicates whether to turn on the balancing between GL and MAS_TRANS_LOG. Totalling on MAS_TRANS_LOG takes quite a while to finish when the size of the table becomes big, so some users turns this feature off by setting this parameter to N.

The cycle balancing processing flow is as follows:

1. Cycle Balance Table Updates

   At the end of a cycle, all transaction processing must stop so that amounts accumulated in the GL accounts can be copied to the Cycle Balance table. The beginning balance of the first cycle in a day is always assumed zero.

2. Cycle Balance Summary

   From the Cycle Balance table, the process creates several totals based on the cycle balance type and saves them in the Cycle Balance Summary table. The cycle balancing types are as follows:

   - Payment Suspense (P)—Pending payments to be delivered to the merchant.
   - Accounts Receivable Suspense (A)—Fees pending collection
   - Accounts Receivable Invoice (I)—Open invoices
   - MAS Edit Suspense (S)—Suspended transactions

   The totals accumulated by the process are as follows:

   - Transactions In
   - Transactions Out
   - System Total Amount
   - GL Amount

   Based on the balance type, different transaction amounts are accumulated to generate these totals.
   See "Cycle Balance Calculations" on page 211 for details.

3. End of Cycle Report Creation

   After the End of Cycle Balancing is performed, the End of Cycle Balancing report file is created. The GL file is saved in the directory given by the mas.57_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
   See "File Formats" on page 171 for details on the file name and format.

   To determine whether the cycle is balanced, two scenarios are possible:

   - End of Cycle Balancing Report—This report contains the balancing of the system, which includes the entire processing in the current cycle, from merchant's transmission to merchant settlement and payment. The report documents the GL posting and indicates whether all incoming transactions have been processed for the current cycle.
     See "Scheduler" on page 154.
   - Manual Calculation—For the cycle to be called balanced, the following calculation must be true: trans_in -trans_out = GL amount. However, the system does not automatically perform this calculation. GUI inquiries can be used by a system operator to review the amounts and to determine whether the system is balanced.

4. Resume Transaction Processing

   When cycle balancing is finished, transaction processing is resumed.

   The Total MAS_TRANS_LOG (Y/N) parameter indicates whether to turn on the balancing between GL and MAS_TRANS_LOG. Totalling on MAS_TRANS_LOG takes quite a while to finish when the size of the table becomes big, so some users turns this feature off by setting this parameter to N.

End of Day Balancing

End of Day Balancing is basically a summary of all cycle balances for the day. As for cycle balancing, the same totals are calculated; however, the source of information comes from different places:

- Transactions In—Sum of 'Transactions In' totals calculated in the cycle balance summary for all cycles belonging to this GL date. Each cycle balance type has one EOD total calculated this way.
- Transactions Out—Sum of 'Transactions Out' totals calculated in the cycle balance summary for all cycles belonging to this GL date. Each cycle balance type has one EOD total calculated this way.
- System Total Amount—These amounts are calculated based on the cycle balance type the same way as for cycle balancing, i.e. directly from the Transaction Log table, but for the entire day.
- GL Amount—Sum of 'GL Amount' totals calculated in the cycle balance summary for all cycles belonging to this GL date. Each cycle balance type has one EOD total calculated this way.

Several additional total amounts are calculated for the EOD balancing, but these totals are for information only:

- Manual Adjustments
- Equipment/Supplies Adjustments
- Retained Deposits
- Reserved Deposits
- Non Settled Card Type Payments
- Wire Payments
- Manual Wire Payments
- ACH Payments

Based on the balance type, different transaction amounts are accumulated to generate these totals.
See "EOD Cycle Balance Calculations" on page 212 for details.

After the EOD Balancing is performed, the EOD Balancing report file is created. The file is saved in the directory given by the mas.57_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

To determine whether the cycle is balanced, two scenarios are possible:

- EOD Balancing Report—contains the balancing of the system, which includes the entire processing in the current GL date, from merchant's transmission to merchant settlement and payment. The report documents the GL posting, and indicates whether all incoming transactions have been processed for this GL date.
  See "Scheduler" on page 154.
- Manual Calculation—For the cycle to be called balanced, the following calculation must be true: trans_in -trans_out = GL amount. However, the system does not automatically perform this calculation. GUI inquiries can be used by a system operator to review the amounts and to determine whether the system is balanced.

GL Export

The GL Export process exports to an outgoing file all amounts accumulated during the GL date in the GL accounts. The GL export must be run daily, before the end of the GL date.

As soon as the export is performed, records are created in the GL_ACCT_ACCUM table with the new GL export date. When this is done, the GL accounts are set to zero in GL_chart_of_acct,and the GL date is changed to the next day in Field_value_ctrl table.

The GL file is saved in the directory given by the mas.51_dir parameter in the *$OSITE_ROOT/cfg/istparam.cfg* file.
See "File Formats" on page 171 for details on the file name and format.

Supporting Processing

As identified by the architecture section (“IST/MAS Server Components” on page 14), several components are used to assist the main processing components. These supporting components are used to schedule system tasks, log system messages, or convert amounts to the appropriate currency.

This section presents a brief overview on the duties of the supporting components:

- Scheduler
  See “Scheduler” on page 154.
- Logging
  See “Logging” on page 160.
- Currency Conversion
  See “Currency Conversion” on page 161.
- Reporting
  See “Reporting” on page 164.

Scheduler

The Scheduler module is an independent server providing the capability of launching tasks at pre-defined times of the day, days of the week or month to any identified business application. Through this module you are able to schedule, manually or automatically, all the tasks required by a business application, like on-line file processing or daily operation and maintenance.

The Scheduler reads the tasks which are set by system administrators at the time of system configuration and uses the information to create a task log. You may manually override the configured tasks and append new tasks through the GUI. Refer to the *IST/MAS Business Operations Guide* for details on how to use the GUI to create new tasks.

The Scheduler is executed via task manager of the underlying IST/Foundation system. In other words, the Scheduler must have a task entry defined in istparam.cfg.

The Scheduler continuously peeks into the Scheduler database and identifies tasks that are scheduled to be run. (See “Database Setup” on page 157 for details.) Once a task has been identified as execution ready, the Scheduler creates an ELF message with a command number and related parameters. This message is sent to the appropriate application server for actual execution.

Daily Operations

Each individual business application employing the Scheduler must have a server running that accepts task-launching ELF messages and launches the tasks accordingly. Each business application has its own specific configuration to ensure that the server is running. For *IST/MAS*, the mas_daily server accepts the ELF messages, and the server is automatically initialized via apm.src.

To receive scheduled commands, each of the business applications employing the Scheduler have to set up its individual application server (Figure 24 on page 156). To identify these servers to the Scheduler server, the application must specify the servers' mailbox. The mailbox name is given via a configuration parameter in istparam.cfg with the following syntax:

```
<usage>.cmd_mailbox <mb_name>
```

usage	published application domain name of the business-application's server.
mb_name	mailbox name of the business-application's server.

The following configuration identifies that the task server of *IST/MAS* is called MAS_DAILY.

```
mas.cmd_mailbox MAS_DAILY
```

When the Scheduler prepares and tries to launch a task, it needs to find out the mailbox name of the server that should receive the task. This is done by mapping the <usage>.cmd_mailbox to the configured mailbox name of the application's task server. Following the above example for *IST/MAS*, The Scheduler sends all task launch messages belonging to *IST/MAS* to the mailbox MAS_DAILY.

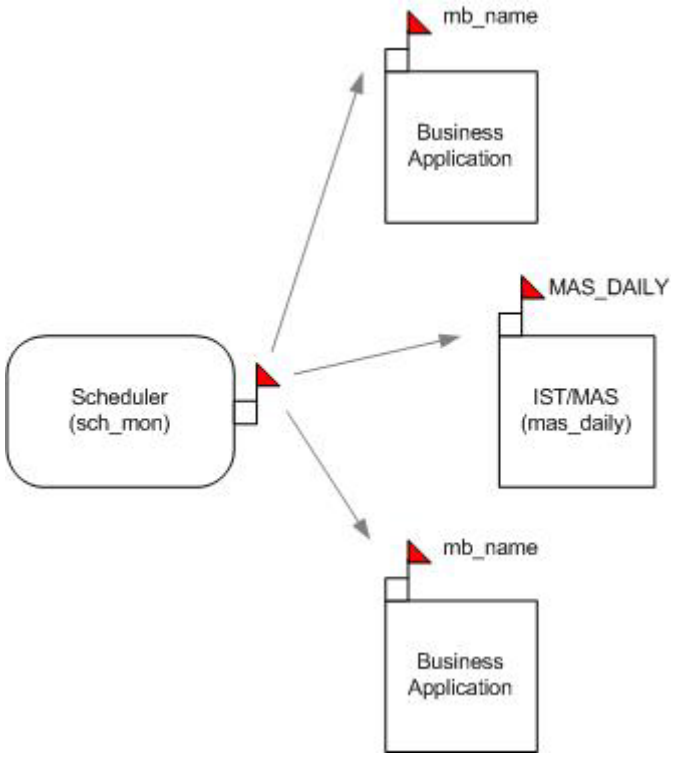


Figure 24 Scheduler

On-line Processing Windows

In many business applications, apart from scheduled command launching, there is also an element of on-line file processing. This processing consists of constantly monitoring the file system, and if a certain types of file is detected, the business application automatically starts the file processing.

If a business application provides on-line file processing, two configuration options are available:

- On-line file processing always active once the business application is up and running.
- On-line file processing active only within certain processing windows.

If the on-line file processing is always active, an incoming file is processed automatically, and the Scheduler module is not required.

If an on-line file processing window is desired, the Scheduler provides a way to have such processing windows defined and activated. This is done through a command toggling mechanism with special commands.

For, the special commands are 100 (toggle File Recognition on) and command number 200 (toggle File Recognition off).

The business application must provide calling information on its file recognition method via the following istparam.cfg parameters:

```
<usage>.fr_dll                  <callback_library>.DLL
<usage>.fr_func                 <callback_function>
```

usage	published application domain name of the business-application's server.
callback_library	dynamic linked library containing the file-recognition callback function.
callback_function	callback function providing the on-line file recognition processing.

For *IST/MAS* the following entries should be present in istparam.cfg:

```
mas.fr_dll                      libomas.DLL
mas.fr_func                     process_incoming
```

If such parameters are defined, the Scheduler maintains in memory a file recognition status for the business application. When a command number 100 (toggle FR on) is scheduled, the Scheduler change the recognition status to ON, while a command number 200 (toggle FR off) will change the status to OFF. Upon restart, the initial file recognition status is always set to OFF.

If the file recognition status is set to ON, the Scheduler calls the corresponding callback function as defined in istparam.cfg at 60 second intervals. It is up to the business application to implement the actual file recognition logic in the supplied function. The commands 100 and 200 (stop and start File Recognition) are the only two commands interpreted by the monitoring module of Scheduler, and they are not send to the corresponding application servers.

Database Setup

Scheduling tasks requires GUI input into the database, where the result of scheduling is also reported. These tables are used by the Scheduler:

- TASK_CMD_LIST stores all available commands ready for scheduling. A tasks is identified by command number, usage, and command parameters. The Usage column identifies the business application. The same command number between different business applications may perform very different tasks. It is up to the individual business application to publish its command numbers, parameters and usages.
- TASKS stores all tasks, that is, all scheduled commands and their execution times. A tasks is uniquely identified by a Task Number, and a tasks number is not related to a command number. Many server tasks may execute the same command.

- TASK_LOG logs the execution details of scheduled tasks. It records the scheduled date-time of the launched task, the start and end date-time of the launched task, and the preliminary execution results.

Due to its scope, the Scheduler can only decide whether a task launch message has been successfully sent to the destination application server. The Scheduler records in the TASK_LOG table the end date time as it finishes sending the task launch message to the application server's mailbox; the date and time do not reflect the end of the actual task. Also, the Result column in TASK_LOG is marked by the Scheduler as *executed* once the Scheduler successfully sent the message to the application server. This also does not guarantee that the task has been successfully executed.

The business application is responsible for providing the actual start and end date-time of the executed task and its result. The business application must report back to Scheduler this information via an API function.

Supported *IST/MAS* Scheduler Commands

The *IST/MAS* commands supported by the Scheduler include, but are not limited to, the following:

- Start File Recognition—Starts File Recognition.
- Stop File Recognition—Stops File Recognition
- ACH Payment—Prepares and generates the ACH payment records.
- Wire—Prepares and generates the ACH payment records.
- Manual Wire—Prepares and generates the Manual Wire payment records.
- Invoice Generation—Prepares and generates the Invoice records.
- GL Export—Exports GL amounts and resets GL accounts to zero.
- Transaction Counting—Counts and consolidates transactions based on Item Count Plan.
- Rate Re-assessment—Reassesses fees based on the tier setup.
- Next Settlement Date—Prepares next settlement dates for settlement plans.
- Non-Activity Fees—Generates non-activity fees.
- Remittance—Generates non-activity fees.
- Aging—Selects and ages pending invoices; generates write-off transactions.
- Adjustment—Sends adjustment transactions to the Posting server for processing.
- Fee Group Rate Update—Update fees based on fee group settings.

- Mas Fee Future Effective Date—Update fees based on scheduled effective dates.
- Resubmit Suspended Batch—Sends suspend transactions to the Validation server for re-tries.
- Expected Merchant Transmission—Based on expected date and time, determine whether expected files have arrived.
- Cycle Balancing—Accumulates GL totals based on GL types and generates cycle balancing.
- Payment Count—Counts payment to consolidate monthly totals.
- Reprocess Lost Transactions—Accepts processed file and instructs File Recognition to re-process. File recognition determines and reprocesses the transactions that are not present in MAS_TRANS_LOG, MAS_TRANS_SUSPEND or MAS_TRANS_ERROR tables.
- MAS Outbounder File—Generates the outbounder file.
- Invoice File—Generates the Invoice file.
- EOD Balancing—Generates End-of-Cycle and End-of-Day report data.
- Monthly Summary—Consolidates monthly totals.
- Submit Mas_trans_error—Sends error transactions to the Fee server for re-processing.
- Re-generate ACH—Re-generates ACH payment files.
- Re-generate Wire—Re-generates Wire payment files.
- Re-generate Manual Wire—Re-generates Manual Wire payment files.
- Generate Payment Log File—Generates the detail Payment Log file.
- Cycle Balancing Report—Accumulates GL totals based on GL types and generates cycle balancing records.
- EOD Balancing Report—Accumulates GL totals based on GL types and generates EOD balancing records
- System Call—Executes shell commands.

The *IST/MAS* commands fall within several groups of processes, as shown in Figure 25 on page 160

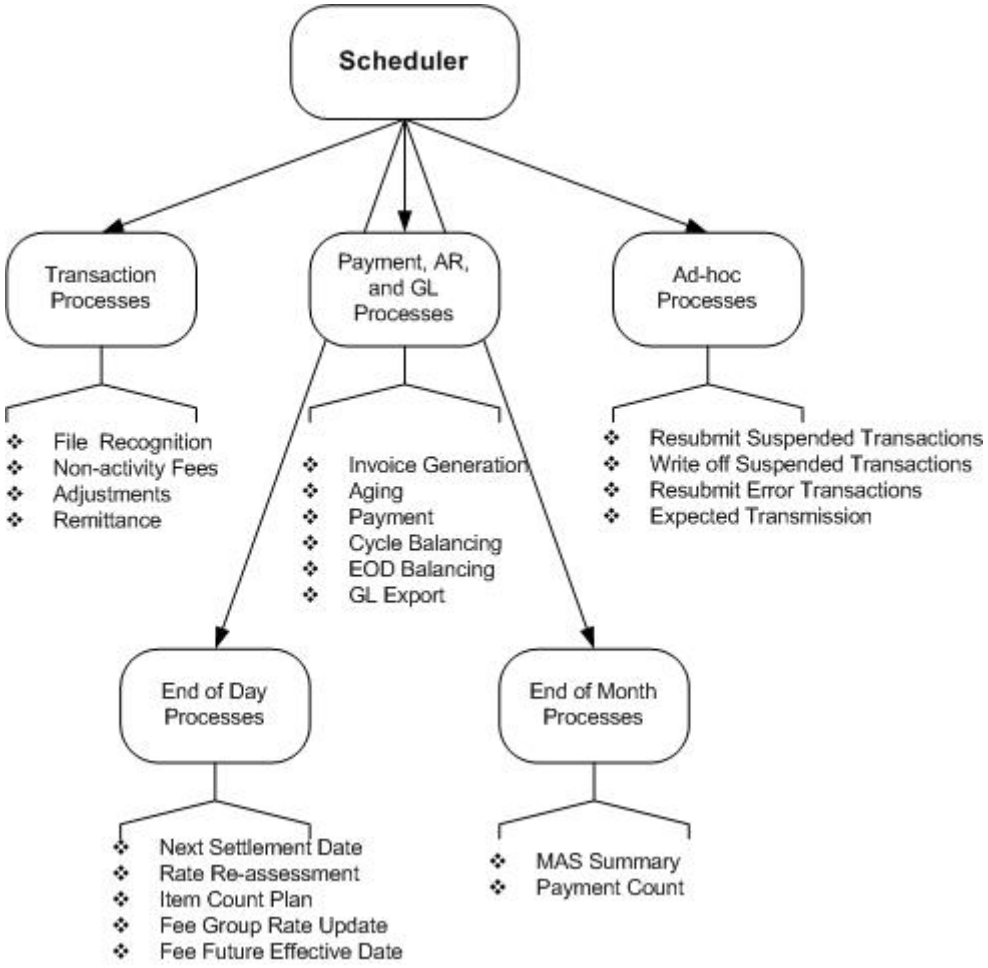


Figure 25 Scheduler—Commands

Logging

Event logging is a standard action performed by all components in *IST/MAS*. Events are logged to the database for auditing purposes, including event details, date-stamp, and initiating user.

Logging is done at two levels:

- *IST/MAS* Client level (GUI)—every change done via GUI is saved into the event log (an internal table). These are business events/errors of interest to *IST/MAS*, like suspended transactions. The log contains the transaction and reason codes to allow manual editing of suspended transactions for submittal and reprocessing.

- *IST/MAS* Server level—every component logs information in the EVENT_LOG table. These are lower-level processing events/errors, like errors writing to the database.

Logging can be customized via configuration. See *Setting Debug Levels* in *IST Installation Procedure Guide* for more information.

Currency Conversion

The system supports merchants with multiple payment currencies. In this case, merchants have multiple accounts, each having a unique account currency code defined in the Entity Account table. Processors, on the other hand, have a unique currency defined in the Institution table.

During transaction processing, currency conversion may occur at several stages. The conversion from one currency to another is done through the Currency Conversion module, and the conversion rates are defined in multiple Currency Conversion Plans in the same module.

The Currency Conversion module is very flexible, allowing for both direct and triangulation conversion. It allows the user to define exchange rates between the defined base currency and any other currencies. Additionally, *IST/MAS* allows the processor to choose the pre-converted rate from the incoming file (from external system like IST/Clearing) if the currency to be converted happens to be one of the Clearing converted currency. Also, additional currency conversions that may be required within the *IST/MAS* processing - it can as well use the historical rate that was effective at the date/time the transaction was processed in the external system (to be in sync with the same rate) or use the latest exchange rate as desired by the user".

The Currency module may store values for the following parameters for each defined set of exchange rules (Buy Rate, Mid Rate, Sell Rate), as well as the ability to define other exchange rules (Rounding, Mark Up/Down).

In addition, *IST/MAS* via the Currency Conversion module allows for currency rate update from numerous sources, which include the following:

- Diners
- Discover
- ENECUR
- Europay
- IPM
- MasterCard

- Visa
- Client-specific

Refer to the *Currency Conversion Guide* for details.

Processes Requiring Currency Conversion

The following processes may require currency conversion.

1. Posting—During transaction posting, the transaction amount is posted to the merchant's account in the transaction currency, provided that the currency is the account currency. If that is not the case, the transaction amount is converted before posting into the currency of the account.

2. Pricing (Fees)—Fees are calculated in the currency of the fee as defined in the MAS Fees table. When Pricing applies fees, base amounts (which can be in different currencies) will always be converted to the currency of the fee.

   Example:

   The fee currency is USD, the transaction amount currency is CAD, and to calculate the fee both Per Item and Percent fee rates are used.

Base Amount CAD	Count	Per Item Rate USD	Percent Rate %
500	10	0.5	2

   The Per Item fee is automatically calculated in CAD. To calculate the percent fee, the system converts the CAD amount to USD using the currency conversion rate and then applies the Percent fee rate.

   Fee = 10 x 0.5 + (500 x USD_conversion_rate) x 0.02

3. Item Count Plan—The Item Count Plan is used to calculate the base counts and amounts used by Pricing to calculate fees. Based on the Item Count Plan configuration (Currency Code to Accumulate and Convert to Currency fields in the Item Count Plan Det table) two scenarios are possible:

   a. Amounts are accumulated only for the currency specified in the Currency Code to Accumulate field and converted to the currency specified in the Convert to Currency field. This scenario may be required, for example, when preferred rates are offered for processing in one currency.

   b. The Currency Code to Accumulate field is set to 000, and amounts for all merchant transactions are converted to the currency specified by the Convert to Currency field before accumulation.

Example:

Six amounts are used to calculate the base transaction amount, three in USD (10, 20, 30) and 2 in CAD (15, 25, 35). The following table illustrates the conversions occurring based on the values found in the Currency Code to Accumulate and Convert to Currency fields.

Currency Code to Accumulate	Convert to Currency	Result	Calculations
USD	USD	USD	10 + 20 + 30 = 60 USD
USD	CAD	CAD	10 + 20 + 30 = 60 USD To obtain CAD: 60 x CAD_Conversion_rate
CAD	CAD	CAD	15 + 25 + 35 = 75 CAD
000	CAD	CAD	60 x CAD_Conversion_rate + 75

4. GL Posting—To be able to obtain a balanced GL record, the system must post to GL in a single currency. For this purpose, transaction amounts are converted to the institution currency (set up in the Institution table) before being posted to the GL.

Currency Conversion Setup

To initiate currency conversion whenever required, the following setup must be performed:

1. In the Merchant Account Management GUI:
   a. Set the account currency in the Entity Account table.
   b. Set the institution currency in the Institution table.
   c. Set the ID of the Currency Conversion Plan in the Entity Account table.
   d. Set the ID of the Currency Conversion Plan in the Institution table.
   e. Set The Item Account Plan in the Item Account Plan Det table (Currency Code to Accumulate and Convert to Currency fields).

   See *IST/MAS Business Operations Guide* for details on how to set up the account currency, the institution currency, the Currency Plan ID, and the Item Count Plan.
2. In the Currency Conversion module, set Currency Conversion Plans.

   Refer to the *Currency Conversion Guide* for details on how currency conversion is done and how Currency Conversion Plans are set up.

Reporting

Although it does not have a reporting component, *IST/MAS* prepares reporting data into the database, and a third-party report generator can be used to create various reports (i.e., Crystal Reports, Mobius, etc.). The system provides the ability to perform the following:

- Obtain reporting information at all levels within the merchant hierarchy
- Specify the level of reporting by role in the hierarchy
- Breakdown reports to merchants to lower child levels, but do not allow aggregation to a parent level.

All transactions/messages, which enter and exit the application, independent of source and destination, are logged in the database as separate records. Since all activity is logged to a database, the information held can be queried to support multiple purposes. For example, by using a third party report writer, specific user/client information can be compiled.

The following components are used to facilitate reporting:

- MAS Summary Table—contains the reporting information.
- Merchant Reporting System (MRS)—contains the reporting setup.
- Report Files—contains monitoring information related to transaction processing, like EOD and End of Cycle balancing information.

MAS Summary Table

The MAS Summary table is used as the basis for creating several reports. The information in this table is collected from other log tables by the following processes:

- MAS Summary—This process creates monthly summary records from the MAS Transaction Log table, which stores information on the transaction life cycle. However, the process excludes all write-off transactions since these transactions are considered a loss, and no reporting is required. If no date is supplied as a parameter, the process calculates the default date as being the month of the current GL date minus one month.
- Payment Count—This process uses the Payment Log table to count all payments made to an entity in a month.

Although the information in the MAS Summary table can be used to create any kind of report, the following is a list of possible reports or inquiries that you may find useful. Note that any third-party application can be used to create reports.

- Month End Totals Roll-up Inquiry—This inquiry is used to monitor the volumes, revenue, accounts receivable, and collection of revenue. The inquiry would include some of the following items: sales, credits and cash advance volumes, interchange pass through, fees collected.
- Month End Totals Comparison—This is used to analyze revenue spreads, revenue as a percentage of volumes, and expense to revenue ratio. These reports are helpful in reviewing a specific portfolio or hierarchy level and give information on a merchant / portfolio's profitability.
- Top Merchants in a TID, MAS Code, or MAS Group—The inquiry is used to view a ranking of merchants based on some element of revenue or expense. The query can be used to address loss of revenue due to an under performing merchant or portfolio and re-pricing.
- Commission Report—This is a commission report for equipment sales for a specific period. The sales offices use the report to track productivity and commissions. This is helpful in reviewing commission payments to sales offices or groups. Reports are usually generated daily.

Merchant Reporting System (MRS)

The MRS is used to setup entity reporting. It defines reports, assigns reports to entities, and sets up a reporting schedule. The setup information is saved in the MRS table.

Note, however, that the system does not generate reports. A reporting system must be used to create the actual reports.

For details on the MRS, see the *IST/MAS Business Operations Guide*.

Report Files

During processing, the system generates the following report files:

- Payment Log Report—From the Payment Log table, a Payment Log report file is generated to contain all payment records.
- Outbounder Report—Based on the Payment Log records, the process takes the appropriate detail transactions from the MAS Transaction Log and creates the Outbounder Report file.
- End of Cycle Balancing Report—This report contains the balancing of the system, which includes the entire processing in the current cycle, from merchant's transmission to merchant settlement and payment. The report documents the GL posting and indicates whether all incoming transactions have been processed for the current cycle.

- EOD Balance Report—This report contains the balancing of the system, which includes the entire processing in the current GL date, from merchant's transmission to merchant settlement and payment. The report documents the GL posting, and indicates whether all incoming transactions have been processed for this GL date.

5

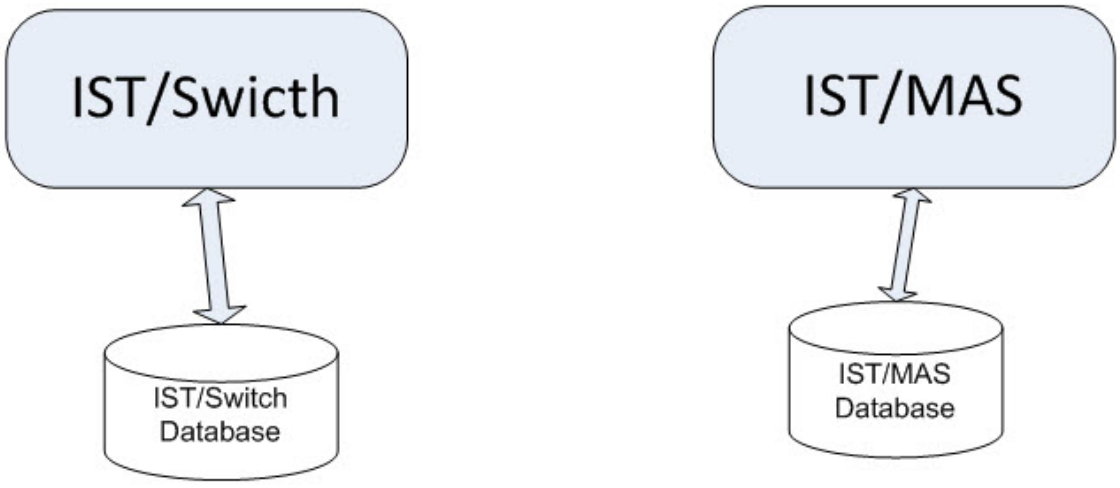
Direct Import Transaction from IST/Switch

IST/MAS can directly import transaction from IST Switch based on following criteria:

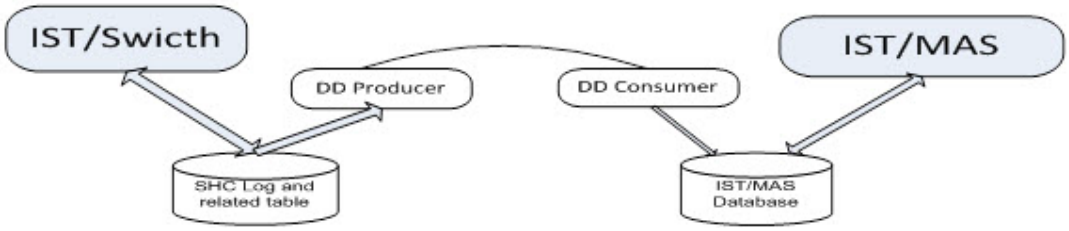
- Direct import transactions to IST/MAS from IST/Clearing takes place when two products are installed in the same database schema. Refer to Direct Import section for more details.
- IST/MAS can directly import transactions from other IST products (IST/Switch) when different products are installed within different database schema. This is handled by DD Producer and DD Consumer.

Transactions Directly Imported to IST/MAS from IST/Switch

IST/Switch and IST/MAS are usually installed in different database considering the nature of these two products as following diagram:



In order to directly import transactions from IST/Switch, DataDistributor maybe used to build the connection between IST/Switch and IST/MAS as following diagram:



When the DD Producer on Switch site receives a request from DD Consumer from MAS site, it will use pre-configured condition to extract certain completed transaction data from SHC_LOG and other tables such as POS related tables through DD to MAS. These pre-configured condition can be, for example, every single message transaction in Switch as Switch receives the response. Using DataDistributor will ensure guaranteed delivery of transactions. DD Consumer on MAS site will store the raw transaction data in a raw transaction table, DATADIST_CONTROL and DATADIST_QUEUE_01. Refer to *IST MAS Data Distributor - Consumer User Guide* for more details.

The process sw_to_mas_proc from MAS will read raw transaction data from DATADIST_CONTROL, DATADIST_QUEUE_01, and enrich it with MAS related information. The enrichments may include:

- MAS TID (Transaction Identifier) from Switch message type, e.g. 200 message in Switch mapped to Debit transaction TID in MAS
- MAS Code matching or assignment
- Card scheme

After transaction is enriched, batch and transaction records can be logged in mas_in_batch_log and in_mas_trans_log tables after forming batch based on predefined configurable conditions are met, such as number of transactions or certain time interval. After a new record is created in, Mas_dir_imp will read the transaction data from in_mas_trans_log and mas_in_batch_log to finish import.

NOTE:	
	• TID will be mapped by fetching the value from column mapped_TID of Switch2masTID table. In Switch2masTID, user configured TIDs has to be already loaded for possible combination of Msgtype, pcode and respcode values from SCHLOG. • MASCODE will be generated by direct casting of two character from the fields pos_entry_cd, pos_condition_code, pos_pin_cap_code, pos_cap_code, cardprod and chip_typ from SHCLOG table and configured into corresponding MASCODE related tables. For value less than two character then fill '-' prior to the character. If any value is null from these fields, then fill default value as 99.

Processing Specifications

Input and Output Files

File Name Masks

The file-name format for the input file consists of several fields separated by a period. For the current definition of the file-name mask, the fields are as follows:

```
<institution>.<job_name>.<file_type>.<julian_date>.
<sequence_no>
```

institution	10-character field containing the base institution
job_name	8-character field containing the job name, like clearing, invoice, etc.
file_type	2-character field containing the type of file. Example: 01 (clearing file), 21 (clearing file without file duplication check), 04 (reversal), 24 (reversal file without file duplication check), 05 (adjustment).
julian_date	3-character field containing the Julian[1] date when the file was created.
sequence_no	3-character field containing the sequence number

1. A Julian date is expressed as the number of days since January 1 of the current year.

The name of the incoming file is flexible, and you can define your own format via GUI in the MAS_FILE_TYPE table. However, only certain elements may be used to create the mask. It is possible to change the length and position of particular elements, but the elements must always conform to this format:

```
'$' + length + 'category'
```

where category must be either of I (institution), N (job number), J (julian date), or S (sequence number).

NOTE:	Fields without a dollar sign are compared with the exact value provided in the file name mask.

Example:

```
$10I.$8N.01.$3J.$3S
```

$10I	10-character string for the institution
$8N	character string for the job number
01	value used for the input file type
$3J	3-character string for the Julian date
$3S	3-character string for the sequence number assigned to all files created within the Julian date

Naming Output Files

The output-file naming convention follows the same convention as the input files. See "File Name Masks" on page 170 for details. However, the name of the job is used to indicate the type of output file (for example, ACH). The value for the numeric ID of the file type must match the value set up in the MAS_FILE_TYPE table for the selected type.

Bank name can also be specified in the output file name. This identifies the settlement bank where the payment file will be sent to.

Example:

The following name mask creates ACH payment files.

```
10I.ACH.$B.52.$3J.$3S
```

File Formats

IST/MAS uses the Universal Agent (UA) Subsystem to generate all *IST/MAS* output files and to read the input file. To generate an output file, the agent receives the source information from various *IST/MAS* subsystems. The translation is based on detailed format descriptions provided to the agent via configuration files.

These are the input and output files handled by the agent:

- Input File
  See "Input File" on page 172 for details on the file format.
- Adjustment Input File
  See "Adjustment Input File" on page 177 for details on the file format.
- GL
  See "GL File" on page 186 for details on the file format.

- ACH Payment
  See “ACH Payment File” on page 187 for details on the file format.
- Wire Payment
  See “Wire Payment File” on page 191 for details on the file format.
- Outbounder
  See “Outbounder File” on page 193 for details on the file format.
- Invoice
  See “Invoice File” on page 198 for details on the file format.
- Manual Wire Payment
  See “Manual Wire Payment File” on page 200 for details on the file format.
- Payment Log
  See “Payment Log File” on page 202 for details on the file format.
- EOD/ End of Cycle Report
  See “EOD/ End of Cycle Report File” on page 204 for details on the file format.

Input File

The input file consists of message detail records organized into multiple batches. One batch is identified by a batch header record and a batch trailer record. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 145-character long and are delimited by the new line character.

This is the format of the input file:

```
File header
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type

Field name	Offset	Length	Type	Description
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
trans_id	2	12	alpha	Unique identifier for the transmission.
entity_id	14	18	alpha	Internal ID assigned to an entity.
card_scheme	32	2	alpha	A code used to identify a card scheme. Values: 03 (Amex), 04 (Visa), 05 (MasterCard), 06 (Europay), 07 (Diners Club), 08 (Discover), 09 (JCB), or other card specified in the table CARD_SCHEME.
mas_code	34	25	alpha	A code used to determine how fees are to be charged to a transaction.
mas_code_downgrade	59	25	alpha	Identifies the rates to be charged for a downgraded transaction.
nbr_of_items	84	10	number	The number of items included in the transaction record.
amt_original	94	12	number	The original amount of the transaction. Decimal places: 2.
add_cnt1	106	10	number	The number of items included in add_amt1.
add_amt1	116	12	number	Additional amount of the incoming transaction, e.g., credit amount. Decimal places: 2.
add_cnt2	128	10	number	The number of items included in add_amt2.
add_amt2	138	12	number	Additional amount of the incoming transaction, e.g., cashback. Decimal places: 2.
external_trans_id	150	25	alpha	The external transaction ID is used for reporting purposes when it is needed to identify source transactions in an external system.
trans_ref_data	175	25	alpha	Transaction Reference Data.

Field name	Offset	Length	Type	Description
suspend_reason	200	2	alpha	Reason for Suspend.
orig_ext_trans_id	202	25	alpha	Original transaction sequence number.
trans_date_time	227	16	alpha	Local date on which the transaction was authorized.
retrieval_ref_nbr	243	12	alpha	A unique document reference number automatically generated by the system for each processed transaction.
lookup_key	255	16	number	IST/Clearing supplied key that serves as a reference pointer to a date/time which is used to pick exchange rates effective that time when MAS does a fresh conversion.
curr_code1	271	3	alpha	1st Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the MAIN institution (that owns the GL).
conv_amt1	274	12	number	Converted amount value in Curr_Code1.
curr_exp1	286	1	number	A number indicating the decimal point location in the currency amount.
curr_code2	287	3	alpha	2nd Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the destination institution (where the transaction is cleared to, e.g. Visa/MasterCard).
conv_amt2	290	12	number	Converted amount value in Curr_Code2.
curr_exp2	302	1	number	A number indicating the decimal point location in the currency amount.

Field name	Offset	Length	Type	Description
curr_plan_inst_id	303	10	alpha	The institution whose exchange rate was used by IST/Clearing in the currency conversion Viz., MasterCard, Visa etc. Combination of LOOKUP_KEY and this field makes it meaningful.
Message File header				
trans_type	0	2	alpha	Transmission type. (FH, File header)
file_type	2	2	alpha	A code used to identify the file type. • Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) • Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
file_date_time	4	16	number	Day and time when the file was created.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
suspend_level	44	1	alpha	Indicator for the level of suspension applied to the incoming activity file.
Message File trailer				
trans_type	0	2	alpha	Transmission type. (FT, File trailer)
file_amt	2	12	number	Total amount of the transactions in the file. Decimal places: 2.
file_cnt	14	10	number	Total count of the transactions in the file.
file_add_amt1	24	12	number	Total additional amount of the incoming transaction in the file, e.g., credit amount. Decimal places: 2.

Field name	Offset	Length	Type	Description
file_add_cnt1	36	10	number	The number of items included in file_add_amt1.
file_add_amt2	46	12	number	Total additional amount of the incoming transaction in the file, e.g., cashback. Decimal places: 2.
file_add_cnt2	58	10	number	The number of items included in file_add_amt2.
Message Batch header				
trans_type	0	2	alpha	Transmission type.
batch_curr	2	3	alpha	Currency of the batch.
activity_date_time	5	16	number	System date and time.
merchantId	21	30	alpha	ID of the merchant.
inBatchNbr	51	9	number	Batch number assigned by the clearing system and used to match the *IST/MAS* transaction to the clearing transaction.
inFileNbr	60	9	number	File number assigned by the clearing system and used to match the *IST/MAS* transaction to the clearing transaction.
billInd	69	1	alpha	Y/N flag indicating whether the merchant is billed for each batch (non-activity batch header fee).
orig_batch_id	70	9	alpha	Original Batch Identifier.
orig_file_id	79	9	alpha	Original File Identifier.
ext_batch_id	88	25	alpha	External Batch Identifier.
ext_file_id	113	25	alpha	External File Identifier.
sender_id	138	25	alpha	Sender Identifier.
microfilm_nbr	163	30	alpha	Microfilm Number
institution_id	193	10	alpha	Institution Identifier.
batch_capture_dt	203	16	alpha	Batch Capture Date.
Message Batch trailer				

Field name	Offset	Length	Type	Description
trans_type	0	2	alpha	Transmission type.
batch_amt	2	12	number	Total amount of the transactions in the batch. Decimal places: 2.
batch_cnt	14	10	number	Total count of the transactions in the batch.
batch_add_amt1	24	12	number	Total additional amount of the incoming transaction in the batch, e.g., credit amount. Decimal places: 2.
batch_add_cnt1	36	10	number	The number of items included in batch_add_amt1.
batch_add_amt2	46	12	number	Total additional amount of the incoming transaction in the batch, e.g., cashback. Decimal places: 2.
batch_add_cnt2	58	10	number	The number of items included in batch_add_amt2.

Adjustment Input File

The adjustment input file consists of message detail records organized into multiple batches. One batch is identified by a batch header record and a batch trailer record. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 250-character long and are delimited by the new line character.

This is the format of the input file:

```
File header
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
trans_id	2	12	alpha	Unique identifier for the transmission.
entity_id	14	18	alpha	Internal ID assigned to an entity.
card_scheme	32	2	alpha	A code used to identify a card scheme. Values: 03 (Amex), 04 (Visa), 05 (MasterCard), 06 (Europay), 07 (Diners Club), 08 (Discover), 09 (JCB), or other card specified in the table CARD_SCHEME.
mas_code	34	25	alpha	A code used to determine how fees are to be charged to a transaction.
mas_code_downgrade	59	25	alpha	Identifies the rates to be charged for a downgraded transaction.
nbr_of_items	84	10	number	The number of items included in the transaction record.
amt_original	94	12	number	The original amount of the transaction. Decimal places: 2.
add_cnt1	106	10	number	The number of items included in add_amt1.
add_amt1	116	12	number	Additional amount of the incoming transaction, e.g., credit amount. Decimal places: 2.
add_cnt2	128	10	number	The number of items included in add_amt2.
add_amt2	138	12	number	Additional amount of the incoming transaction, e.g., cashback. Decimal places: 2.
external_trans_id	150	25	alpha	The external transaction ID is used for reporting purposes when it is needed to identify source transactions in an external system.

Field name	Offset	Length	Type	Description
fee_text_description	175	100	alpha	Text description associated with the transaction.
lookup_key	275	16	number	IST/Clearing supplied key that serves as a reference pointer to a date/time which is used to pick exchange rates effective that time when MAS does a fresh conversion.
curr_code1	291	3	alpha	1st Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the MAIN institution (that owns the GL).
conv_amt1	294	12	number	Converted amount value in Curr_Code1.
curr_exp1	306	1	number	A number indicating the decimal point location in the currency amount.
curr_code2	307	3	alpha	2nd Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the destination institution (where the transaction is cleared to, e.g. Visa/MasterCard).
conv_amt2	310	12	number	Converted amount value in Curr_Code2.
curr_exp2	322	1	number	A number indicating the decimal point location in the currency amount.
curr_plan_inst_id	323	10	alpha	The institution whose exchange rate was used by IST/Clearing in the currency conversion Viz., MasterCard, Visa etc. Combination of LOOKUP_KEY and this field makes it meaningful.
Message File header				

Field name	Offset	Length	Type	Description
trans_type	0	2	alpha	Transmission type. (FH, File header)
file_type	2	2	alpha	A code used to identify the file type. • Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) • Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
file_date_time	4	16	number	Day and time when the file was created.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
suspend_level	44	1	alpha	Indicator for the level of suspension applied to the incoming activity file.
Message File trailer				
trans_type	0	2	alpha	Transmission type. (FT, File trailer)
file_amt	2	12	number	Total amount of the transactions in the file. Decimal places: 2.
file_cnt	14	10	number	Total count of the transactions in the file.
file_add_amt1	24	12	number	Total additional amount of the incoming transaction in the file, e.g., credit amount. Decimal places: 2.
file_add_cnt1	36	10	number	The number of items included in file_add_amt1.
file_add_amt2	46	12	number	Total additional amount of the incoming transaction in the file, e.g., cashback. Decimal places: 2.

Field name	Offset	Length	Type	Description
file_add_cnt2	58	10	number	The number of items included in file_add_amt2.
Message Batch header				
trans_type	0	2	alpha	Transmission type.
batch_curr	2	3	alpha	Currency of the batch.
activity_date_time	5	16	number	System date and time.
Message Batch trailer				
trans_type	0	2	alpha	Transmission type.
batch_amt	2	12	number	Total amount of the transactions in the batch. Decimal places: 2.
batch_cnt	14	10	number	Total count of the transactions in the batch.
batch_add_amt1	24	12	number	Total additional amount of the incoming transaction in the batch, e.g., credit amount. Decimal places: 2.
batch_add_cnt1	36	10	number	The number of items included in batch_add_amt1.
batch_add_amt2	46	12	number	Total additional amount of the incoming transaction in the batch, e.g., cashback. Decimal places: 2.
batch_add_cnt2	58	10	number	The number of items included in batch_add_amt2.

Remittance Input File

The Remittance input file consists of message detail records organized into multiple batches. One batch is identified by a batch header record and a batch trailer record. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum xxx-character long and are delimited by the new line character.

This is the format of the input file:

```
File header
   Batch header
         Message detail
         Message detail
         ...
   Batch trailer
   Batch header
         Message detail
         Message detail
         ...
   Batch trailer
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
trans_id	2	12	alpha	Unique identifier for the transmission.
entity_id	14	18	alpha	Internal ID assigned to an entity.
card_scheme	32	2	alpha	A code used to identify a card scheme. Values: 03 (Amex), 04 (Visa), 05 (MasterCard), 06 (Europay), 07 (Diners Club), 08 (Discover), 09 (JCB), or other card specified in the table CARD_SCHEME.
mas_code	34	25	alpha	A code used to determine how fees are to be charged to a transaction.
mas_code_downgrade	59	25	alpha	Identifies the rates to be charged for a downgraded transaction.
nbr_of_items	84	10	number	The number of items included in the transaction record.
amt_original	94	12	number	The original amount of the transaction. Decimal places: 2.
add_cnt1	106	10	number	The number of items included in add_amt1.

Field name	Offset	Length	Type	Description
add_amt1	116	12	number	Additional amount of the incoming transaction, e.g., credit amount. Decimal places: 2.
add_cnt2	128	10	number	The number of items included in add_amt2.
add_amt2	138	12	number	Additional amount of the incoming transaction, e.g., cashback. Decimal places: 2.
external_trans_id	150	25	alpha	The external transaction ID is used for reporting purposes when it is needed to identify source transactions in an external system.
invoice_nbr	175	12	number	Invoice Number.
cheque_nbr	187	26	alpha	Cheque Number.
payer_name	213	30	alpha	Payer Name.
lookup_key	243	16	number	IST/Clearing supplied key that serves as a reference pointer to a date/time which is used to pick exchange rates effective that time when MAS does a fresh conversion.
curr_code1	259	3	alpha	1st Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the MAIN institution (that owns the GL).
conv_amt1	262	12	number	Converted amount value in Curr_Code1.
curr_exp1	274	1	number	A number indicating the decimal point location in the currency amount.
curr_code2	275	3	alpha	2nd Currency Code to which the Source Transaction currency is converted in IST/Clearing. This is synonymous with the currency associated with the destination institution (where the transaction is cleared to, e.g. Visa/MasterCard).

Field name	Offset	Length	Type	Description
conv_amt2	278	12	number	Converted amount value in Curr_Code2.
curr_exp2	290	1	number	A number indicating the decimal point location in the currency amount.
curr_plan_inst_id	291	10	alpha	The institution whose exchange rate was used by IST/Clearing in the currency conversion Viz., MasterCard, Visa etc. Combination of LOOKUP_KEY and this field makes it meaningful.
Message File header				
trans_type	0	2	alpha	Transmission type. (FH, File header)
file_type	2	2	alpha	A code used to identify the file type. • Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) • Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
file_date_time	4	16	number	Day and time when the file was created.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
suspend_level	44	1	alpha	Indicator for the level of suspension applied to the incoming activity file.
Message File trailer				
trans_type	0	2	alpha	Transmission type. (FT, File trailer)
file_amt	2	12	number	Total amount of the transactions in the file. Decimal places: 2.

Field name	Offset	Length	Type	Description
file_cnt	14	10	number	Total count of the transactions in the file.
file_add_amt1	24	12	number	Total additional amount of the incoming transaction in the file, e.g., credit amount. Decimal places: 2.
file_add_cnt1	36	10	number	The number of items included in file_add_amt1.
file_add_amt2	46	12	number	Total additional amount of the incoming transaction in the file, e.g., cashback. Decimal places: 2.
file_add_cnt2	58	10	number	The number of items included in file_add_amt2.
Message Batch header				
trans_type	0	2	alpha	Transmission type.
batch_curr	2	3	alpha	Currency of the batch.
activity_date_time	5	16	number	System date and time.
merchantId	21	30	alpha	ID of the merchant.
inBatchNbr	51	9	number	Batch number assigned by the clearing system and used to match the *IST/MAS* transaction to the clearing transaction.
inFileNbr	60	9	number	File number assigned by the clearing system and used to match the *IST/MAS* transaction to the clearing transaction.
billInd	69	1	alpha	Y/N flag indicating whether the merchant is billed for each batch (non-activity batch header fee).
orig_batch_id	70	9	alpha	Original Batch Identifier.
orig_file_id	79	9	alpha	Original File Identifier.
ext_batch_id	88	25	alpha	External Batch Identifier.
ext_file_id	113	25	alpha	External File Identifier.
sender_id	138	25	alpha	Sender Identifier.
microfilm_nbr	163	30	alpha	Microfilm Number

Field name	Offset	Length	Type	Description
institution_id	193	10	alpha	Institution Identifier.
batch_capture_dt	203	16	alpha	Batch Capture Date.
Message Batch trailer				
trans_type	0	2	alpha	Transmission type.
batch_amt	2	12	number	Total amount of the transactions in the batch. Decimal places: 2.
batch_cnt	14	10	number	Total count of the transactions in the batch.
batch_add_amt1	24	12	number	Total additional amount of the incoming transaction in the batch, e.g., credit amount. Decimal places: 2.
batch_add_cnt1	36	10	number	The number of items included in batch_add_amt1.
batch_add_amt2	46	12	number	Total additional amount of the incoming transaction in the batch, e.g., cashback. Decimal places: 2.
batch_add_cnt2	58	10	number	The number of items included in batch_add_amt2.

GL File

The GL file consists of message detail records. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 120-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Message detail
   Message detail
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
gl_acct_id	0	20	alpha	ID of the GL account.
amt_credit	28	12	number	The positive amount accumulated. Decimal places: 2.
amt_debit	40	12	number	The negative amount accumulated. Decimal places: 2.
File header				
file_id	0	12	alpha	ID of the file.
gl_date	12	8	date	Current GL date for which accumulation of the current code is done. Format: YYYYMMDD.
export_date	20	8	date	Date when the GL account data was exported to a file. Format: YYYYMMDD.
export_time	28	8	number	Time when the GL account data was exported to a file.
File trailer				
file_id	0	12	alpha	ID of the file.
credit_amt	12	12	number	Total credit amount for all GL records in the file. Decimal places: 2.
debit_amt	24	12	number	Total debit amount for all GL records in the file. Decimal places: 2.
record_cnt	36	12	number	Total count of all GL records in the file.

ACH Payment File

The ACH Payment file consists of message detail records and ACH addenda records organized into a single batch. The batch is identified by a batch header record and a batch trailer record. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 120-character long and are delimited by the new line character.

NOTE:	The ACH addenda record is not generated for *IST/MAS*.

This is the format of the file:

```
File header
   Batch header
          Message detail
          Message detail
          ACH addenda
          ...
   Batch trailer
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
rec_type	0	1	alpha	Record type.
trans_cd	1	2	alpha	Transaction code.
recv_dfi_id	3	9	alpha	Receiving DFI number (e.g., transaction routing number)
dfi_acct_nbr	12	17	alpha	DFI account number (DDA account).
amt	29	10	number	Amount of the activity transaction. Decimal digits: 2.
indv_id_nbr	39	15	alpha	*IST/MAS* entity ID.
indv_name	54	22	alpha	Name of the entity ID.
discret_data	76	2	alpha	Private data.
addenda_rec_ind	78	1	number	Addenda record indicator. Values: 1 (addenda present), 0 (no addenda records)
trace_nbr	79	15	alpha	Trace number.
ACH addenda (not generated for *IST/MAS*)				
...				
File header				
rec_type	0	1	alpha	Record type.
priority_cd	1	2	alpha	Priority code.
imm_dest	3	10	alpha	Name of the immediate destination (receiving point or ACH).
imm_origin	13	10	alpha	Name of the immediate origin (sending point or ACH).

Field name	Offset	Length	Type	Description
transm_date	23	6	date	Transmission date. Format: YYYYMMDD.
transm_time	29	4	number	Transmission time.
file_id_mod	33	1	alpha	File ID
rec_size	34	3	alpha	Size of the record.
blk_factor	37	2	alpha	Block factor (round up to the next whole number) for the total number of records written.
format_cd	39	1	alpha	Code identifying the format.
imm_dest_name	40	23	alpha	Name of the immediate destination (receiving point or ACH).
imm_origin_name	63	23	alpha	Name of the immediate origin (sending point or ACH).
ref_cd	86	8	alpha	Reference code.
File trailer				
rec_type	0	1	alpha	Record type.
batch_cnt	1	6	number	Number of batches in the file.
blk_cnt	7	6	number	Block count.
entry_addenda_cnt	13	8	number	Number of entry/addenda on file.
entry_hash	21	10	number	Sum of the batch hash.
tot_db_entry_amt	31	12	number	Total debit amount. Decimal digits: 2.
tot_cr_entry_amt	43	12	number	Total credit amount. Decimal digits: 2.
reserved	55	39	alpha	Reserved field.
Batch header				
rec_type	0	1	alpha	Record type.
serv_class_cd	1	3	alpha	Code identifying the service class.
company_name	4	16	alpha	Name of the processor.
company_discret_data	20	20	alpha	Processor's ID.
company_id	40	10	alpha	Configurable; usually the tax ID of the processor.

Field name	Offset	Length	Type	Description
stndrd_entry_class_cd	50	3	alpha	Code identifying the standard entry class.
company_entry_desc	53	10	alpha	Descriptive label for the company.
company_desc_date	63	6	date	System date. Format: YYMMDD.
eff_entry_date	69	6	date	Intended effective date for all transactions within the batch. Format: YYMMDD.
settl_date_j	75	3	number	Settlement date (Julian)
origin_stat_cd	78	1	alpha	Original status code.
origin_dfi_id	79	8	alpha	Original DFI identifier.
batch_nbr	87	7	number	Batch number.
Batch trailer				
rec_type	0	1	alpha	Record type.
serv_class_cd	1	3	alpha	Code identifying the service class.
entry_addenda_cnt	4	6	number	Number of addenda entries in the batch.
entry_hash	10	10	number	Sum of the first 8 trace number positions.
tot_db_entry_amt	20	12	number	Total debit amount. Decimal digits: 2.
tot_cr_entry_amt	32	12	number	Total credit amount. Decimal digits: 2.
company_id	44	10	alpha	Configurable; usually the tax ID of the processor.
msg_auth_cd	54	19	alpha	Message authorization code.
reserved	73	6	alpha	Reserved field.
origin_dfi_id	79	8	alpha	Original DFI identifier.
batch_nbr	87	7	number	Batch number.

Wire Payment File

The Wire Payment file consists of message detail records organized into multiple batches. One batch is identified by a batch header record (A batch trailer record is not needed). The beginning of the file is marked by the file header record, and the end of the file is marked by an empty file trailer record. Records are maximum 150-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Batch header
          Message detail
          Message detail
          ...
   Batch header
          Message detail
          Message detail
          ...
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
_rec_type	0	4	alpha	Label for the value of the rec_type field. (RID =)
rec_type	4	2	alpha	Record type.
_item_ref	6	7	alpha	Label for the value of the item_ref field. (TID = R)
item_ref	13	7	number	Unique identifier for the wire.
_value_date	20	6	alpha	Label for the value of the value_date field. (VDT =)
value_date	26	6	date	Date when the wire is to be sent. Format: YYMMDD.
_amount	32	6	alpha	Label for the value of the amount field. (AMT =)
amount	38	16	alpha	Transfer amount.
_source_id	54	6	alpha	Label for the value of the source_id field. (SID =)
source_id	60	20	alpha	Customer short name.
_repeat_code	80	6	alpha	Label for the value of the repeat_code field. (RPT =)
repeat_code	86	8	alpha	Repeat number.

Field name	Offset	Length	Type	Description
_auth_code	94	6	alpha	Label for the value of the auth_code field. (AUT =)
auth_code	100	16	alpha	Authentication code.
Batch header				
_rec_type	0	4	alpha	Label for the value of the rec_type field. (RID =)
rec_type	4	2	alpha	Record type.
_trans_id	6	6	alpha	Label for the value of the trans_id field. (TID =)
trans_id	12	5	alpha	Unique identifier for the transmission.
batch_seq	17	3	number	Sequence number for the batch.
_value_date	20	6	alpha	Label for the value of the value_date field. (VDT =)
value_date	26	6	date	Date when the wire is to be sent. Format: YYMMDD.
_rec_total	32	6	alpha	Label for the value of the rec_total field. (RCT =)
rec_total	38	5	alpha	Total number of records in the batch.
_dollar_total	43	6	alpha	Label for the value of the dollar_total field. (AMT =)
dollar_total	49	16	alpha	Total amount in the batch.
_auth_code	65	6	alpha	Label for the value of the auth_code field. (AUT =)
auth_code	71	16	alpha	Authentication code.
File header				
_rec_type	0	4	alpha	Label for the value of the rec_type field. (RID =)
rec_type	4	2	alpha	Record type.
_trans_id	6	6	alpha	Label for the value of the trans_id field. (TID =)
trans_id	12	8	alpha	Unique identifier for the transmission.
_value_date	20	6	alpha	Label for the value of the value_date field. (VDT =)
value_date	26	6	date	Date when the wire is to be sent. Format: YYMMDD.
_rec_total	32	6	alpha	Label for the value of the rec_total field. (RCT =)
rec_total	38	5	alpha	Total number of records in the file.
_dollar_total	43	6	alpha	Label for the value of the dollar_total field. (AMT =)

Field name	Offset	Length	Type	Description
dollar_total	49	16	alpha	Total amount in the file.
_user_id	65	6	alpha	Label for the value of the user_id field. (UID =)
user_id	71	20	alpha	CIW user ID.
_auth_code	91	6	alpha	Label for the value of the auth_code field. (AUT =)
auth_code	97	16	alpha	Authentication code.
File trailer (empty)				

Outbounder File

The Outbounder file consists of message detail records. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 601-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Message detail
   Message detail
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
File header				
trans_type	0	2	alpha	ID of the transmission type. (FH, File Header)
file_type	2	2	alpha	A code used to identify the file type. ● Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) ● Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
activity_date_time	4	16	alpha	System date and time.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
payment_run	44	4	alpha	Indicates a running position within a payment processing date. Values: 0 - 999.
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
institution_id	2	1	alpha	Institution ID.
tid	3	6	alpha	Transaction identifier (TID).
tid_settl_method	9	1	alpha	Identifies whether merchant's account is credited (C) or debited (D) for this transaction.
mas_code	10	12	alpha	A code used to determine how fees are to be charged to a transaction.

Field name	Offset	Length	Type	Description
card_scheme	22	2	number	A code used to identify a card scheme. Values: 03 (Amex), 04 (Visa), 05 (MasterCard), 06 (Europay), 07 (Diners Club), 08 (Discover), 09 (JCB), or other card specified in the table CARD_SCHEME.
amt_original	24	19	number	The original amount of the transaction. Decimal places: 7.
curr_cd_original	43	3	alpha	Original currency code.
nbr_of_items	46	13	number	The number of items included in the transaction record.
entity_id	59	18	alpha	Internal ID assigned to an entity.
add_amt1	77	19	number	The number of items included in add_amt1. Decimal places: 7.
add_cnt1	96	13	number	Additional amount of the incoming transaction, e.g., credit amount.
add_amt2	109	19	number	The number of items included in add_amt2. Decimal places: 7.
add_cnt2	128	13	number	Additional amount of the incoming transaction, e.g., cashback.
activity_date_time	141	12	alpha	System date and time.
amt_billing	153	19	number	Billing amount. Decimal places: 7.
curr_billing	172	3	alpha	Code of the billing currency.
priced_date_time	175	12	alpha	Date and time for pricing.
date_to_settle	187	8	date	Date on which the transaction is to be billed to the entity account. Format: YYYYMMDD.
curr_rate	195	19	number	The exchange rate used to convert the original amount to the billing amount. Decimal places: 7.

Field name	Offset	Length	Type	Description
settl_flag	214	1	alpha	Settled Flag for activity and fees. Y or y (merchant is paid or billed) N, null or other (merchant is not paid).
posting_entity_id	215	18	alpha	Indicates with which entity to settle.
percent_amt	233	19	number	The amount charged as a result of applying a rate expressed as a percentage. Decimal places: 7.
per_item_amt	252	19	number	Amount charged as a result of applying a per item rate. Decimal places: 7.
percent_rate	271	19	number	Rate used to calculate fees on amounts. Decimal places: 7.
per_item_rate	290	19	number	Rate used to calculate fees on a number of transactions. Decimal places: 7.
fee_pkg_id	309	12	number	A fee package is used to define the set of fees that can be applied to a merchant.
fee_grp_id	321	12	number	System-assigned ID of the fee group.
external_trans_nbr	333	25	alpha	The external transaction ID is used for reporting purposes when it is needed to identify source transactions in an external system.
invoice_nbr	358	12	number	A unique number used to identify the invoice in the system.
payment_term	370	2	number	Defines the payment term, i. e., the number of days from the invoice to the due date.
cheque_nbr	372	25	alpha	The number from the cheque used as remittance.
trans_routing_nbr	397	12	alpha	Identifies the institution (bank) which holds the ACCT_ NBR.
mas_code_downgrade	409	12	alpha	Identifies the rates to be charged for a downgraded transaction.

Field name	Offset	Length	Type	Description
expected_mas_code	421	12	alpha	The MAS code expected for the entity to which the transaction belongs.
trans_description	433	100	alpha	Transaction description.
reversal_ind	533	1	alpha	Reversal indicator. Values: R (reversed), blank (not reversed).
unapplied_cash	534	19	number	Stores the total of unapplied cash from cheques. Decimal places: 7.
payment_seq_nbr	553	12	number	A unique number that identifies a payment record.
payment_effect_date	565	8	date	Payment effective date. Format: YYYYMMDD.
gl_date	573	8	date	Current GL date for which accumulation of the current code is done. Format: YYYYMMDD.
entity_level	581	1	alpha	Numeric value representing the level of an entity in the entity hierarchy.
settl_freq	582	1	alpha	Settlement frequency. Values: D (daily), W (weekly), B (bi- weekly), M (monthly), Q (quarterly), S (semi-annually), A (annually), L (fixed list).
payment_proc_dt	583	8	date	Payment processing date, Format: YYYYMMDD.
acct_nbr	591	25	alpha	Account Number.
File trailer				
trans_type	0	2	alpha	ID of the transmission type. (FT, File trailer)
record_cnt	2	10	number	Total number of records in the file.

Invoice File

The Invoice file consists of message detail records. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 600-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Message detail
   Message detail
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
File header				
trans_type	0	2	alpha	Transmission type. (FH, File header)
file_type	2	2	alpha	A code used to identify the file type. • Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) • Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
activity_date_time	4	16	alpha	System date and time.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
institution_id	2	1	alpha	Institution ID.
invoice_nbr	3	12	number	A unique number used to identify the invoice in the system.
entity_acct_nbr	15	12	number	Merchant's invoice account.
payment_term	27	3	alpha	Defines the payment term, i. e., the number of days from the invoice to the due date.

Field name	Offset	Length	Type	Description
due_date	30	8	date	The date the payment is to be made to the invoice before it is considered overdue.
invoice_date	38	8	date	The date the invoice was generated.
amt_invoiced	46	19	number	Amount of the invoice. Decimal places: 7.
curr_code	65	3	alpha	ISO currency code.
amt_applied	68	19	number	Amount of remittance applied to the invoice. Decimal places: 7.
invoice_status	87	1	alpha	Used to indicate the state of the invoice. Sample values: O (invoiced, open, still unpaid), P (partially paid, user input), C (billed, completely paid, closed), R (returned cheque), N (written off invoice).
aging_cycle	88	1	alpha	Number of aging cycles an invoice passed through.
inv_credit_amt	89	19	number	Accumulated unapplied cash from previous cheque payments or amounts resulted from a fee calculation. Decimal places: 7.
entity_id	108	18	alpha	Internal ID assigned to an entity.
entity_level	126	20	alpha	Numeric value representing the level of an entity in the entity hierarchy.
File trailer				
trans_type	0	2	alpha	Transmission type. (FT, File trailer)
record_cnt	2	10	number	Total number of records in the file.

Manual Wire Payment File

The Manual Wire Payment file consists of message detail records. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 635-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Message detail
   Message detail
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
bank_name	0	30	alpha	Name of the settlement bank.
city	30	20	alpha	City.
state	50	20	alpha	State.
trans_routing_nbr	70	20	alpha	Identifies the institution (bank) which holds the ACCT_ NBR.
dda	90	19	alpha	Direct deposit account.
next_line	109	1	alpha	Next line.
merchant_name	110	30	alpha	Name of the merchant.
merchant_number	140	20	alpha	ID of the merchant.
processing_date	160	20	alpha	Processing date.
deposit_amt	180	20	alpha	Deposit amount.
payment_seq_nbr	200	20	alpha	A unique number that identifies a payment record.
File header				
header	0	12	alpha	Header.

Field name	Offset	Length	Type	Description
day_of_week	12	11	alpha	Day of week.
sys_date	23	18	alpha	System date.
blank_line	41	2	alpha	Empty line.
column1_1	43	30	alpha	Column 1, cell 1.
column1_2	73	20	alpha	Column 1, cell 2.
column1_3	93	20	alpha	Column 1, cell 3.
column1_4	113	20	alpha	Column 1, cell 4.
column1_5	133	20	alpha	Column 1, cell 5.
column2_1	153	30	alpha	Column 2, cell 1.
column2_2	183	20	alpha	Column 2, cell 2.
column2_3	203	20	alpha	Column 2, cell 3.
column2_4	223	20	alpha	Column 2, cell 4.
column2_5	243	20	alpha	Column 2, cell 5.
File trailer				
spaces	0	50	alpha	Spaces.
footer	50	20	alpha	Content of footer.
total_amt	70	20	alpha	Total amount of all records in file.

Payment Log File

The Payment Log file consists of message detail records. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 120-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Message detail
   Message detail
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
trans_type	0	2	alpha	Transmission type. (01, Detail transaction)
institution_id	2	1	alpha	Institution ID.
entity_id	3	18	alpha	Internal ID assigned to an entity.
entity_level	21	1	alpha	Numeric value representing the level of an entity in the entity hierarchy.
payment_effect_dt	22	8	date	Date and time, based on the calendar of the merchant's bank, when a payment is going to be effective. Format: YYYYMMDD.
entity_acct_id	30	12	number	A system- assigned ID used to identify an account that is set up for an entity.
payment_method_cd	42	1	alpha	A code used to identify the payment method. Sample values: A (ACH), W (wire), I (invoice), M (manual wire).
payment_seq_nbr	43	12	number	A unique number that identifies a payment record.
link_pay_seq_flag	55	1	alpha	Flag set by the system to indicate whether more than one accumulator was used to create the payment record (Wire and Manual Wire payment methods).
payment_proc_dt	56	12	alpha	Date and time when the payment was processed.

Field name	Offset	Length	Type	Description
payment_amt	68	19	number	Amount of payment. Decimal places: 7.
payment_run	87	3	number	Indicates a running position within a payment processing date. Values: 0 -999.
trans_routing_nbr	90	12	alpha	Identifies the institution (bank) which holds the ACCT_ NBR.
acct_nbr	102	25	alpha	Number used to identify the entity's account to which payment is to be credited/ debited.
dda_changed	127	1	alpha	Currently not used.
gl_date	128	8	alpha	Current GL date for which accumulation of the current code is done.
filler	136	11	alpha	Filler.
File header				
trans_type	0	2	alpha	Transmission type. (FH, File header)
file_type	2	2	alpha	A code used to identify the file type. • Input files: 01 (clearing), 02 (credits), 03 (authorization), 04 (reversal), 05 (adjustment) • Output files: 51 (GL Export File), 52 (ACH Payment File), 53 (Wire Payment File), 54 (Outbounder File), 55 (Invoice File), 56 (Manual Wire Payment File), 57 (GL End of Day and End of Cycle report files), 58 (Payment Log report file).
activity_date_time	4	16	alpha	System date and time.
activity_source	20	16	alpha	The source of the activity file.
activity_job_name	36	8	alpha	The job name that produced the activity file.
File trailer				

Field name	Offset	Length	Type	Description
trans_type	0	2	alpha	Transmission type. (FT, File trailer)
record_cnt	2	10	number	Total number of records in the file.
file_total	12	19	number	Total amount of all payment records in the file. Decimal places: 7.

EOD/ End of Cycle Report File

The EOD/ End of Cycle Report file consists of message detail records organized into multiple batches. One batch is identified by a batch header record and a batch trailer record. The beginning of the file is marked by the file header record, and the end of the file is marked by the file trailer record. Records are maximum 155-character long and are delimited by the new line character.

This is the format of the file:

```
File header
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   Batch header
          Message detail
          Message detail
          ...
   Batch trailer
   ...
File trailer
```

Each record has an offset format, and these are the fields expected in each record type:

Field name	Offset	Length	Type	Description
Message detail				
rpt_seq	0	5	alpha	Report sequential number.
rpt_subgroup	5	60	alpha	Report subgroup.
master_gl	65	25	alpha	Master GL.
debit_amt	90	20	alpha	Debit amount for this record.

Field name	Offset	Length	Type	Description
credit_amt	110	20	alpha	Credit amount for this record.
net_amt	130	20	alpha	Total net amount for this file.
File header				
fh1	0	5	alpha	Filled with spaces.
fh2	5	60	alpha	Filled with spaces.
fh3	65	25	alpha	Label in the report header (GL Number).
fh4	90	20	alpha	Label in the report header (Debit).
fh5	110	20	alpha	Label in the report header (Credit).
fh6	130	20	alpha	Label in the report header (Net).
File trailer				
ft1	0	5	alpha	Filled with spaces.
ft2	5	60	alpha	Label in the report footer (Balance Check).
ft3	65	25	alpha	Filled with spaces.
debit_amt	90	20	alpha	Total debit amount for this file.
credit_amt	110	20	alpha	Total credit amount for this file.
net_amt	130	20	alpha	Total net amount for this file.
Batch header				
bh1	0	5	alpha	Filled with spaces.
rpt_group	5	60	alpha	Report group.
Batch trailer				
bt1	0	5	alpha	Filled with spaces.
bt2	5	60	alpha	Batch trailer value.
bt3	65	25	alpha	Filled with spaces.
debit_amt	90	20	alpha	Total debit amount for this batch.
credit_amt	110	20	alpha	Total credit amount for this batch.
net_amt	130	20	alpha	Total net amount for this file.
bt4	150	1	alpha	Batch trailer 4.

Transaction Identifier (TID)

Transactions are either sent to *IST/MAS* by external and internal entities or generated by *IST/MAS* itself. Based on their characteristics, transactions are grouped and assigned an internal identifier called Transaction Identifier (TID).

The TID consists of the following five data elements:

```
<txn_domain><txn_scope><txn_class><txn_code><txn_subtype>
```

txn_domain	Transaction Domain
txn_scope	Scope
txn_class	Transaction Class
txn_code	Transaction Code
txn_subtype	Transaction Subtype

Transaction Domain

A Transaction Domain defines an industry of interest, such as the financial industry. The following table identifies the transaction domains currently defined in the system:

Domain	Description
01	Financial Industry—Retail
02	Health Industry
03	Insurance
04	Educational services
05	Government

NOTE:	Transactions related to *IST/MAS* are considered part of the financial industry domain.

Transaction Scope

The Transaction Scope differentiates between static and non-static TIDs. Static TIDs are defined by FIS and used to process various classified transactions.

Non-static TIDs are defined and maintained by the user. An example of non-static TID is the user-generated fee transactions in *IST/MAS*.

The following table identifies the transaction scopes currently defined in the system:

Domain	Scope	Description
01	00	Non Static (User-Defined—*IST/MAS*)
01	01	Static (FIS-defined—IST/Clearing)

Transaction Class

The Transaction Class is used to classify transactions within a defined domain and scope. For example, transactions can be grouped as activity or fee transactions.

The following table identifies the transaction classes currently defined in the system:

Domain	Scope	Class	Description
01	02	00	Default Class
01	01	02	Authorization Transactions
01	02	03	Activity Transactions
01	02	04	Fee Transactions

Transaction Group

The Transaction Group is used to group a set of transactions. For example, certain fee transactions may be designated as processing fees and others as authorization fees. Transaction groups are generally used for reporting and inquiry, when a set of fees can be viewed together.

Transaction group is an attribute of a transaction.

IST/MAS Default TIDs

When the system is installed, a set of TIDs are set up. All TIDs belong to the following domain, scope, and class:

- Domain 01—Financial Transactions
- Scope 02—*IST/MAS*
- Class 03—Activity Transactions
- Class 04—Fee Transactions
- Class 05—Adjustment Transactions

You can choose to use these values or add new TIDs depending on how you want transactions to be identified.

Refer to the *IST/MAS Table Reference Guide* for a complete list of default TIDs currently defined in the system.

User-defined TIDs

User-defined TIDs can be added under the following conditions:

- The transaction domain must be 01 (financial industry; retail).
- The transaction scope must be 02 (user defined; *IST/MAS*).
- Activity transactions must be defined within class 03.
- Fee transactions must be defined within class 04.
- Adjustment transactions must be defined within class 05.
- Subtypes can be used to add multiple TIDs of the same type.

The following table lists sample TIDs:

TID	Class	Code	Subtype	Description
		Activity TIDs		
010003005001	03	005	001	Sale (POS)
010003005002	03	005	002	Sale (Debit)
		Fee TIDs		
010004060098	03	060	001	Handling Fee 1
010004060099	03	060	002	Handling Fee 2

TID	Class	Code	Subtype	Description
Adjustment TIDs				
010005080000	05	080	000	Fee Adjustment
010005090000	05	090	000	Manual Adjustment

Transaction Processing

Validation Enrichment

The following information is saved in a transaction during validation:

From Table	Field Name	Description
ACQ_ENTITY	SETTL_PLAN_ID	Settlement Plan ID
	CALENDAR_ID	Calendar ID
	FEE_PKG	Fee Package
	EXPECTED_MAS_CODE	Expected MAS Code
SETTL_PLAN_TID	GL_ACCT_TO	GL Account To
	GL_ACCT_FROM	GL Account From
	POSTING_ENTITY_ID	Posting Entity ID
	PAYMENT_TERM	Payment Term
	SETTL_FLAG	Settlement Flag
	DATE_TO_SETTLE	Date to Settle
	TID_RESERVE	TID Reserve
ACCT_POSTING_PLAN	ENTITY_ACCT_ID	Entity Account ID

Invoice Status Indicators

The following possible invoice status indicators are defined. Note that invoice-status edits made through the GUI result in lower case status indicators, while processor-generated edits result in uppercase status indicators.

O	invoiced, open, still unpaid
c	paid, after user input, not yet billed
C	billed, completely paid, closed
p	partially paid, user input
P	billed (processed) partially paid
r	reopened (returned cheque - GUI)
R	reopened (returned cheque - processed)
s	reopened partially paid (GUI)
S	reopened partially paid (processed)
N	non collectable (written-off invoice)
m	manually closed
M	manually closed (processed)

The transition between the status indicators is as follows:

An Invoice status of the following:	May be edited through the GUI Remittance Window to the following:	And updated by the Processor automatically to the following:
An Invoice status of the following:	May be edited through the GUI Remittance Window to the following:	And updated by the Processor automatically to the following:
‘O’	‘c’ ‘p’	‘C’ ‘P’
‘P’	‘c’ ‘s’ ‘p’	‘C’ ‘S’ ‘P’
‘C’	‘r’	‘R’
‘O’,’P’,’R’,’S’	‘m’	‘M’

1. When an invoice process creates a new invoice, the invoice is assigned a status of ‘O’.

2. Through the GUI's Remittance screen, the user may select all the invoices to be paid with one cheque payment. The status of those invoices is changed to 'c' (paid in full). If some of the invoices are paid partially, then their status is changed to 'p' (partially paid).

   Statuses 'c' and 'p' that are assigned by selecting invoices through the Remittance Window are used only as intermediate statuses to control user input. The user is allowed to mark or to reserve invoices to be paid, and those invoices are actually paid in the billing process.

   All the remittance transactions that are entered through the Remittance window are processed at the end of the day by the posting process, which posts the remittance transactions and changes corresponding invoice statuses. For example, status 'c' on the invoice is changed to 'C' (closed), and 'p' is changed to 'P' (billed, partially paid).

3. When a partially paid or billed invoice with a status of 'P' is paid in full, the transaction is entered through the Remittance Window, and its status is changed to 'c'.

4. After the posting process, this invoice is finally closed, and the status is changed to 'C'.

Cycle Balance Calculations

The following table shows how cycle balance amounts are calculated based on the cycle balance type (i.e., P, A, I, S).

Total	Description
Payment Suspense (P)	
Transactions In	All activity transactions that came in the current balance cycle
Transactions Out	All activity transactions released in the current balance cycle
System Total Amount	All activity transactions pending payment for the current balance cycle
GL Amount	Total of all GL accounts of the type P. The amounts accumulated in the GL accounts reflect only the current balance cycle.
Accounts Receivable Suspense (A)	
Transactions In	All newly calculated fees to be charged for the current balance cycle
Transactions Out	All released fee amounts for the current balance cycle
System Total Amount	All fees pending collection for the current balance cycle

Total	Description
GL Amount	Total of all GL accounts of the type A. The amounts accumulated in the GL accounts reflect only the current balance cycle.
Accounts Receivable Invoice (I)	
Transactions In	All newly created invoices for the current balance cycle
Transactions Out	All released (paid) invoices for the current balance cycle
System Total Amount	All open invoices for the current balance cycle
GL Amount	Total of all GL accounts of the type I. The amounts accumulated in the GL accounts reflect only the current balance cycle.
MAS Edit Suspense (S)	
Transactions In	All suspended transactions in the current balance cycle
Transactions Out	All transactions released from suspension in the current balance cycle
System Total Amount	All suspended transactions in the current balance cycle
GL Amount	Total of all GL accounts of the type S. The amounts accumulated in the GL accounts reflect only the current balance cycle.

EOD Cycle Balance Calculations

The following table shows how the EOD cycle balance amounts are calculated based on the cycle balance type (i.e., P, A, I, S).

Total	Description
Payment Suspense (P)	
Transactions In	All activity transactions that came in the current GL date
Transactions Out	All activity transactions released in the current GL date
System Total Amount	All activity transactions pending payment for the current GL date
GL Amount	Total of all GL accounts of the type P. The amounts accumulated in the GL accounts reflect only the current GL date.
Accounts Receivable Suspense (A)	
Transactions In	All newly calculated fees to be charged for the current GL date
Transactions Out	All released fee amounts for the current GL date
System Total Amount	All fees pending collection for the current GL date

Total	Description
GL Amount	Total of all GL accounts of the type A. The amounts accumulated in the GL accounts reflect only the current GL date.
Accounts Receivable Invoice (I)	
Transactions In	All newly created invoices for the current GL date
Transactions Out	All released (paid) invoices for the current GL date
System Total Amount	All open invoices for the current GL date
GL Amount	Total of all GL accounts of the type I. The amounts accumulated in the GL accounts reflect only the current GL date.
MAS Edit Suspense (S)	
Transactions In	All suspended transactions in the current GL date
Transactions Out	All transactions released from suspension in the current GL date
System Total Amount	All suspended transactions in the current GL date
GL Amount	Total of all GL accounts of the type S. The amounts accumulated in the GL accounts reflect only the current GL date.
Manual Adjustments (J)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	Total of all manual adjustment transactions that have been entered for the current GL date
GL Amount	Not Applicable
Equipment/Supplies Adjustments (E)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	Total of all equipment and supplies adjustment transactions that have been entered for the current GL date
GL Amount	Not Applicable
Retained Deposits (H)	
Transactions In	All transactions that came in for ‘on hold’ merchants for the current GL date
Transactions Out	All payments to ‘on hold’ merchants that are rerouted for the current GL date to another DDA

Total	Description
System Total Amount	Total of all transactions that have to be paid for the current GL date to ‘on hold’ merchants
GL Amount	Not Applicable
Reserved Deposits (R)	
Transactions In	All new reserve amounts corresponding to the transactions for the current GL date
Transactions Out	All released payments to the reserve accounts for the current GL date
System Total Amount	Total of all pending reserve amounts that have to be paid for the current GL date to the reserve account
GL Amount	Not Applicable
Non Settled Card Type Payments (N)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	All activity transactions that are not settled via *IST/MAS*
GL Amount	Not Applicable
Wire Payments (W)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	All payment transactions handled via Wire payment
GL Amount	Not Applicable
Manual Wire Payments (M)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	All payment transactions handled via Manual Wire payment
GL Amount	Not Applicable
ACH Payments (C)	
Transactions In	Not Applicable
Transactions Out	Not Applicable
System Total Amount	All payment transactions handled via ACH payment
GL Amount	Not Applicable

Wildcard MAS Codes

Instead of complete MAS codes, wildcard MAS codes can be used to define rates for an entire group of MAS codes. The wildcard MAS code can be up to 25 character long, and the dash (-) character is used as the wildcard character.

If the MAS code structure is AABBCCCCDDD, the following wildcard MAS codes can be defined in the Field Value Control table for each TID.

```
AABBCCCC---
AABB----DDD
AABB-------
AA--CCCCDDD
AA--CCCC---
AA------DDD
AA---------
-----------
```

NOTE:	Each character in the MAS code is replaced by one wildcard character.

Entitlement Setup

Predefined Classes

The Entitlement Subsystem is supplied with a number of predefined classes. Client specific classes may be may be created via Entitlement GUI with an existing permission. New permission tokens may be created/added as required by an application.

The following table lists the predefined classes:

Class ID	Class Name	Purpose
1	Users	Reserved class, with which all User objects known to the system must be associated. Permission is: **Z –** The user can be supervised.
2	SecTables	Reserved class, with which all *Entitlement Table* objects known to the system must be associated. Used by the entitlement server to control administrative access to the entitlement database tables. Permissions: **I –** user may add a new records to the table. **S –** user may select existing records from the table. **D –** user may delete existing records from the table. **U –** user may modify existing records in the table. **Z –** user may supervise existing records in the table.
6	GUI Profile	Reserved class, with which all Entitlement objects known to the system may be associated. Used by the Entitlement server to access the Entitlement Administration GUI. Permissions: **A –** user may access the object. **Z –** user through this screen may supervise the object.

Predefined Objects

The Entitlement Subsystem is supplied with a number of predefined classes. Client specific classes may be may be created via Entitlement GUI.

The following table lists the predefined objects.

Object ID	Class ID	Purpose
Admin_Group	1	Top-Level Entitlement Administrator
ObjectTable	2	Object Entitlement Table
GroupTable	2	Group Entitlement Table
ClassTable	2	Class Entitlement Table
PermTable	2	Permission Entitlement Table
ENTGUI	6	Entitlement Administration GUI Object

Entitlement Objects for IST/Switch

To create users in the system and assign privileges to them, you must know what objects are available to give access to. Although the Entitlement Subsystem displays all available objects, you must know how to recognize the various types of objects.

Object names are formed from elements separated by periods, and they always start with the element defining the subsystem or application owning the object. If the object is generic (i.e., common to many applications), the subsystem name used is ALL. Note also that ALL is used in other objects to indicate all objects of one type.

Objects are grouped by types into classes. For example, all institution objects are defined within the institution class. The following is a list of classes that are available in *IST/MAS*.

1. Subsystem—Provides the name of the application using this object. For *IST/MAS*, only *mas* or ALL are applicable. The format of the object is as follows:

   ```
   <subsystem_name>
   ```

   Example:

   ```
   ALL
   mas
   ```

2. Subfield—Identifies a class by its ID. These objects are used to grant access to an entire class of objects. Since entitlement is not forced for the institution and merchant classes, these are the only classes that require this extra step. The format of the object is as follows:

   ```
   <subsystem_name>.sp<class_ID>
   ```

Example:

```
mas.sp65
mas.sp75
```

To give access to institution information, grant users permissions over the mas.sp75 object, and only then grant permissions over individual institution objects (e.g., mas.i1234567890).

To give access to merchant information, grant users permissions over the mas.sp65 object, and only then grant permissions over individual merchant objects (e.g., mas.m1234).

3. Institution—Identifies an institution by its ID. To grant permissions over this object type, first give access to the institution class (see the Subfield class on page See page 217 for details). The format of the object is as follows:

   ```
   <subsystem_name>.i<institution_ID>
   ```

   Example:

   ```
   mas.iALL
   mas.i1234567890
   ```

4. Merchant—Identifies a merchant by its ID. To grant permissions over this object type, first give access to the institution class (see the Subfield class on page See page 217 for details). The format of the object is as follows:

   ```
   <subsystem_name>.m<merchant_ID>
   ```

   Example:

   ```
   mas.m1234
   ```

NOTE:	ALL is not implemented for the merchant class.

5. Form—Identifies a GUI form by its ID. The format of the object is as follows:

   ```
   <subsystem_name>.<form_ID>
   ```

   Example:

   ```
   mas.acq_entity
   ```

6. Field[1]—Identifies a field by its ID and the form to which it belongs. The format of the object is as follows:

   ```
   <subsystem_name>.<form_ID>.<field_ID>
   ```

   Example:

   ```
   mas.acq_entity.ACQ_ENTITY.ENTITY_ID
   ```

1. Not implemented

7. Form Button—Identifies a button by its ID and the form to which it belongs. The format of the object is as follows:

   ```
   <subsystem_name>.<form_ID>.<button_ID>
   ```

   Example:

   ```
   mas.allowance.D
   ```

8. Subsystem Button—Identifies a button by its ID. This is a button directly related to the application, like a toolbar button. The format of the object is as follows:

   ```
   <subsystem_name>.<form_ID>.<button_ID>
   ```

Predefined Groups

The Entitlement subsystem currently recognizes only one predefined group. New client specific groups may be created via Entitlement GUI. The predefined group is listed in table below.

Group ID	Class ID	Group Type	Purpose
Admin_Group	1	Group of Users	Group of top-level entitlement administrators

Event Levels

IST/MAS defines two levels of information and error events:

- Level 1—defines business-related events.
- Level2—defines system-related events.

Although the system may be configured to save level 2 events to debug files, it is advisable not to since debug files will grow to unmanageable sizes. Level 2 events should be traced only during system testing.

Level 1 events currently defined in the system are listed in this section. Note that the character sequence of the type %s is replaced by the system with actual values.

For example, the %s and %ld in WIRE CYCLE(%s) DATE(%ld) will be replaced with the actual Wire cycle number and date of the cycle.

Level 1 Information Events

```
NON-ACT date_today %ld
OUT-B Payment Run %ld
OUT-B Payment Run %ld
PAY-LOG Payment Run %ld
ACH CYCLE(%s) DATE(%ld)
ACH Payment Run %ld
INV Invoice File
INV CYCLE(%s) DATE(%ld)
Stop payment on acct %.0lf
WIRE CYCLE(%s) DATE(%ld)
Stop payment on acct %.0lf
WIRE Payment Run %ld
M-WIRE CYCLE(%s) DATE(%ld)
M-WIRE Payment Run %ld
REMIT EI(%s) TO(%s %.8lf) FROM(%s %.8lf)
GL-EXP file type %s
RATE-REASS TIER_ASSM_DATE %ld
RATE-UPDATE Effective date %ld
FGRP-RATE-UPDATE Effective date %ld
SUSPEND-RE-SUBMIT batch %s
ADJUSTMENT EI %s
RESUBMIT-TRANS-ERR INST_ID %s
EXP-FILE expected time %ld
CYCLE-BAL CYCLE %c SEQ %ld
EOD-BAL CYCLE %c SEQ %ld
MAS-SUMMARY
REPROCESS-TXN FILE %s
EOD-BAL-REP End of Cycle
EOD-BAL-REP End of Day
SYSCALL %s
VAL RELEASE FILE %s
VAL release batch %s
VAL release batch %s
```

Level 1 Error Events

```
NON-ACT error txn %.0lf
NON-ACT acq entity not found %s|%s
NON-ACT error txn %.0lf
OUT-B Payment method cd 'O' not in mas_payment_method
OUT-B Payment method cd 'O' not in mas_payment_method
OUT-B Cannot open file %s
OUT-B Payment method cd 'O' not in mas_payment_method
PAY-LOG Payment method cd 'L' not in mas_payment_method
PAY-LOG entity acct read failed %.0lf
PAY-LOG acq entity read failed %s
ERR Payment method cd 'A' not in mas_payment_method
ACH entity acct read failed %.0lf
ACH acq entity read failed %s
ACH acct_accum_det update failed.
ACH Payment method cd 'A' not in mas_payment_method
ACH entity acct read failed .0lf
ARH acq entity read failed %s
ACH entity acct read failed .0lf
ACH acq entity read failed %s
INV Payment method cd 'I' not in mas_payment_method
WIRE Payment method cd 'W' not in mas_payment_method
WIRE entity acct read failed %.0lf
WIRE acq entity read failed %s
WIRE acct_accum_det update failed
WIRE acct_accum_det update failed
WIRE acct_accum_det update failed
WIRE acct_accum_det update failed
WIRE acct_accum_det update failed
WIRE Payment method cd 'W' not in mas_payment_method
WIRE entity acct (%s) read failed.
M-WIRE Payment method cd 'M' not in mas_payment_method
M-WIRE entity acct read failed %.0lf
M-WIRE acq entity read failed %s
M-WIRE acct_accum_det update failed
M-WIRE acct_accum_det update failed
M-WIRE Payment method cd 'M' not in mas_payment_method
M-WIRE entity acct (%.0lf) read failed
M-WIRE acq entity read failed %s
GL-EXP GL_EXP_FILE_TYPE not defined in field_value_ctrl
```

```
RATE-REASS error txn %.0lf
RATE-REASS acq entity not found %s|%s
RATE-REASS error txn %.0lf
RATE-UPDATE Update failed
EOD-BAL eod already exists for gl_date %ld.
MAS-SUMMARY Summary Period (%ld) already exists.
MAS-SUMMARY DbmAllocStmt(%d)
MAS-SUMMARY DBMPrepare(%d)
MAS-SUMMARY freestmt(%d)
REPROCESS-TXN FILE %s was not yet processed.
EOD-BAL-REP EOD_RPT_FILE_TYPE not defined in field_value_ctrl
EOD-BAL-REP Out of Balance
EOD-BAL-REP LAST_CYCLE_BAL not defined in field_value_ctrl
FEE error txn %.0lf
POST error txn %.0lf
POST error txn %.0lf
POST error txn %.0lf
POST error txn %.0lf
VAL can't access %s
VAL can't access %s
VAL can't access %s
VAL Allocate Failure holdings %ld
VAL Allocate Failure wild_mas_code %ld
VAL Allocate Failure tid_reserve %ld
VAL Allocate Failure batch %ld
VAL suspend txn %.0lf
VAL not valid card scheme - %s
VAL not valid acq entity (%s)
VAL not valid fpkg mas code - %s
VAL not valid mas code - %s
VAL not valid fpkg mas code downgrade - %s
VAL not valid mas code downgrade - %s
VAL File (%s) has lost transactions
VAL insert batch %s failed.
VAL insert file %s failed.
VAL suspend batch %s
```

B

Examples of Transaction Processing

Transaction Pricing

This pricing example exemplifies Non-Qual Pricing and Downgrade Processing (Reclassification Advices).

This pricing example illustrates some possible setup scenarios, but it does not imply that this is the only way to set up this type of merchant.

Processing Scenario

Let us assume that the merchant "Shoe Store" is a plain vanilla retail merchant that qualifies for the best interchange rates, namely MasterCard Merit III. The MAS_CODE corresponding to the best qualification rate for MasterCard (Merit III) is MAS_CODE = 1247, so this is the expected rate for this merchant for MasterCard transactions.

However, we also need to consider cases where actual merchant transactions do not qualify for the best rate. In that case, the merchant should be differently charged to recover increased processing costs due to higher interchange fees like, for example, Key Entered, Merit I and Consumer Standard.

The transaction coming from a POS terminal is a Sale transaction, TID = 15. The following cases are considered:

- Case 1— Transaction meets best qualification criteria (actual equals the expected MAS code; MAS_CODE = 1247)
- Case 2—Transaction does not meet best qualification criteria (actual not equal to expected MAS code; actual MAS_CODE = 1001)
- Case 3—Transaction is downgraded (original MAS _CODE = 1247, downgraded MAS _CODE = 999)

Also, the following setup scenarios are considered:

- Merchant Setup with Non-Qualification Flag set to “N”.
- Merchant Setup with Non-Qualification Flange set to “Y”

The first setup does not invoke the non-qualification logic but uses a standard fee package for the merchant to calculate fees for all cases. The second setup invokes the non-qualification logic. However, both approaches should show equivalent results as follows:

- Credit Discount Fee (401). A credit Credit Discount Fee is charged for each of the three cases as 1.85 percent fee.
- Pass-through Fee (402). A Pass Through Fee is charged for fully qualified transactions.

Case	Expected MasCd	MasCd	%	perItem
Case 1	1247	1247	0	0.07
Case 2	1247	1001	0.69	0.12
Case 3	1247	999	1.27	0.12

- Credit Transaction Fee (Per Item Processing Fee) (403). A Credit Transaction Fee is charged for each of the three cases as 0.10 per item fee.

Non-Qualification Flag Set to “N”

Merchant "Shoe Store" has a Fee Package, PkgId = 501, for the MasterCard card scheme. The package has three fees for the TID = 15 (Sale) and MAS_CODE=1247 (corresponding to Merit III):

- Credit Discount Fee (401). A credit Credit Discount Fee is charged.
- Pass-through Fee (402). A Pass Through Fee is charged for fully qualified transactions as per item fee of 0.07.
- Credit Transaction Fee (Per Item Processing Fee) (403). A Credit Transaction Fee is charged.

Similarly, package 501 has all of the above fees set for the other applicable MAS Codes (corresponding to Key Entered, Merit I and Standard rates). The only difference is in the Pass-through Fees (402); unlike fully qualified transactions that have only 0.07 fixed per item fee, transactions that are not fully qualified have both per item and percentage fee as shown in the MAS_FEES table:

PkgId	TIDAct	MAS_CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit III									
501	15	1247	401	1.85	0	+	+	+	+
501	15	1247	402	0	0.07	+	+	+	+
501	15	1247	403	0	0.10	+	+	+	+
Sale Transaction, Key Entered									
501	15	1332	401	1.85	0	+	+	+	+
501	15	1332	402	0.51	0.12	+	+	+	+
501	15	1332	403	0	0.10	+	+	+	+
Sale Transaction, Merit I									
501	15	1001	401	1.85	0	+	+	+	+
501	15	1001	402	0.69	0.12	+	+	+	+
501	15	1001	403	0	0.10	+	+	+	+
Sale Transaction, Consumer Standard									
501	15	999	401	1.85	0	+	+	+	+
501	15	999	402	1.27	0.12	+	+	+	+
501	15	999	403	0	0.10	+	+	+	+

Case 1—Best Qualification Criteria Met

A MasterCard Sale Transaction comes from the merchant "Shoe Store", and the transaction qualifies for the best interchange rate (Merit III) corresponding to MAS_CODE = 1247.

During transaction validation, the appropriate fee package and settlement plan is selected for the merchant ("Shoe Store"), card scheme (MasterCard), Sale Transaction (‘TIDAct = ‘15’) is selected (package PkgId =‘501’ in this case). The incoming transaction is enriched with the appropriate fields from the corresponding settlement plan.

Using the fee package ID (PkgId = 501), incoming TIDAct (15) and MAS_CODE (1247), all fees that should be charged for the transaction are selected, as shown in the table:

PkgId	TIDAct	MAS_CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit III									
501	15	1247	401	1.85	0	+	+	+	+
501	15	1247	402	0	0.07	+	+	+	+
501	15	1247	403	0	0.10	+	+	+	+

NOTE:	The effective date is also part of the primary key for the MAS_FEES table, so that only valid fees for the particular day will be applied.

Next, fee charges are calculated and new fee transactions generated. The percent rate and rate per item are applied to the base transaction amount and count. For each TIDFee, a new fee transaction is created (in this example three new transactions are generated with TID of '401', '402', and '403'). For each combination of TIDFee ('401','402' and '403') and CARD_SCHEME, a record is expected to be found in the SETTL_PLAN_TID table. The fee transaction is enriched with settlement data.

The incoming sale transaction and the three additional fee transactions are logged in the MAS_TRANS_LOG table:

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
TRANS_SEQ_NBR*				
TRANS_SUB_SEQ		new	new	new
TID*	15	401	402	403
MAS_CODE *	1247	1247	1247	1247
CARD_SCHM *	05	05	05	05
AMT_ORIGINAL *	100	100	100	100
CURR_CD_ORIGINAL *	840	840	840	840
NBR_OF_ITEMS *	1	1	1	1
ENTITY_ID *				
ENTITY_TYPE *				
EXTERNAL_ENTITY_ID				
ADD_AMT1 *	0			

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
ADD_CNT1 *	0			
ADD_AMT2 *	0			
ADD_CNT2 *	0			
FILE_ID				
BATCH_ID				
ACTIVITY_DATE_TIME *				
ENTITY_ACCT_ID				
AMT_BILLING	97.98	1.85	0.07	0.10
CURR_BILLING				
DATE_PRICED	system date	system date	system date	system date
DATE_TO_SETTLE	calc. using delay f. from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN
CURR_RATE				
SETTL_FLAG	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
POSTING_ENTITY_ID	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
POSTING_ENTITY_TYP	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_FROM	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_TO	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
PERCENT_AMT	null	calc. amt using percent_rate	calc. amt using percent_rate	calc. amt using percent_rate

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
PER_ITEM_AMT	null	calc. amt using per_item_rate	calc. amt using per_item_rate	calc. amt using per_item_rate
PERCENT_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees
PER_ITEM_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees
CALENDAR_ID_TO_USE *				
SETTL_PLAN_TO_USE *				
FEE_PKG_TO_USE *	501			
EXTERNAL_TRANS_ID				
PAYMENT_SEQ_NBR	null			
INVOICE_NBR	null			
PAYMENT_TERMS	null	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
CHEQUE_NBR	null			
TRANS_ROUTING_NBR				
MAS_CODE_DOWNGRADE*	null			
EXPECTED_MAS_CODE *	null			
SUSPEND_REASON	null			

1. Fields marked with a * are used to generate fee transactions.
2. The empty fields in the original transaction column are obtained either from the incoming transaction message or enriched by the validation module. The empty fields in the fee transaction columns are the same as the corresponding fields of the original transaction.

Case 2—Best Qualification Criteria Not Met

This case covers the example of a transaction that does not meet the expected qualification rate for the merchant (the actual MAS_CODE is different from the expected MAS_CODE).

Let us assume a MasterCard Sale Transaction comes from the merchant "Shoe Store", and the transaction does not qualify for the best interchange rate corresponding to MAS_CODE = 1247. The actual interchange rate corresponds to MAS_CODE = 1001 (Merit I).

For the merchant "Shoe Store" and card scheme MasterCard, based on the TIDAct = '15' there is a record found in the package 501with MAS_CODE = 1001.

Using fee package PkgId = 501, incoming TIDAct = 15 and incoming MAS_CODE = 1001, all fees that should be charged for the transaction are selected, as shown in the table:

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit I									
501	15	1001	401	1.85	0	+	+	+	+
501	15	1001	402	0.69	0.12	+	+	+	+
501	15	1001	403	0	0.10	+	+	+	+

Next, fee charges are calculated and new fee transactions generated. The percent rate and rate per item are applied to the base transaction amount and count. For each TIDFee, a new fee transaction is created (in this example three new transactions are generated with TID of ‘401’, ‘402’, and ‘403’). For each combination of TIDFee (‘401’,’402’ and ‘403’) and CARD_SCHEME, a record is expected to be found in the SETTL_PLAN_TID table. The fee transaction is enriched with settlement data.

The incoming sale transaction and the three additional fee transactions are logged in the MAS_TRANS_LOG table:

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transactio n Fee
TRANS_SEQ_NBR*				
TRANS_SUB_SEQ		new	new	new
TID*	15	401	402	403
MAS_CODE *	1001			

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
CARD_SCHM *	05			
AMT_ORIGINAL *	100	100	100	100
CURR_CD_ORIGINAL *	840			
NBR_OF_ITEMS *	1			
ENTITY_ID *				
ENTITY_TYPE *				
EXTERNAL_ENTITY_ID				
ADD_AMT1 *	0			
ADD_CNT1 *	0			
ADD_AMT2 *	0			
ADD_CNT2 *	0			
FILE_ID				
BATCH_ID				
ACTIVITY_DATE_TIME *				
ENTITY_ACCT_ID				
AMT_BILLING	97.24	1.85	0.81[3]	0.10
CURR_BILLING				
DATE_PRICED	system date	system date	system date	system date
DATE_TO_SETTLE	calc. using delay f. from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN
CURR_RATE				
SETTL_FLAG	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
POSTING_ENTITY_ID	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
POSTING_ENTITY_TYP	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_FROM	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_TO	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
PERCENT_AMT	null	calc. amt using percent_rate	calc. amt using percent_rate	calc. amt using percent_rate
PER_ITEM_AMT	null	calc. amt using per_item_rate	calc. amt using per_item_rate	calc. amt using per_item_rate
PERCENT_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees
PER_ITEM_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees
CALENDAR_ID_TO_USE *				
SETTL_PLAN_TO_USE *				
FEE_PKG_TO_USE *	501			
EXTERNAL_TRANS_ID				
PAYMENT_SEQ_NBR	null			
INVOICE_NBR	null			
PAYMENT_TERMS	null	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
CHEQUE_NBR	null			
TRANS_ROUTING_NBR				

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
MAS_CODE_DOWNGRADE*	null			
EXPECTED_MAS_CODE*	null			
SUSPEND_REASON	null			

1. Fields marked with a * are used to generate fee transactions.
2. The empty fields in the original transaction column are obtained either from the incoming transaction message or enriched by the validation module. The empty fields in the fee transaction columns are the same as the corresponding fields of the original transaction.
3. 100*0.69% + 0.12 = 0.69 + 0.12 = 0.81

Case 3—Downgraded Transaction Processing

When a transaction is downgraded by a card association, the Clearing system sends the reclassification advice transaction to *IST/MAS* with the original transaction information, including original MAS_CODE (*e.g.,*1247) and downgraded MAS_CODE (*e.g.,* 999). *IST/MAS* reverses the entire fee transactions based on the original MAS_CODE, and re-generates the fee transactions based on the new MAS_CODE.

Let us assume that the original transaction was a Sale Transaction with MAS_CODE = 1247 (original MAS_CODE) processed as per Case 1. Further, the original transaction was downgraded by the card association to the interchange rate corresponding to MAS_CODE = 999 (downgraded MAS_CODE).

This reclassification advice is sent to *IST/MAS* with both the original and downgraded MAS_CODEs. The incoming transaction is considered a non-settlement transaction and used for fee calculation only (settlement flag is set to ‘N”). First, all the fees that were applied to the original transaction have to be calculated and reversed. Further, new fees corresponding to the downgraded MAS_CODE need to be applied.

Using fee package PkgId = 501, incoming TIDAct = 15 and original MAS_CODE = 1247, all fees that should be charged for the transaction are selected, as shown in the table:

PkgId	TIDAct	MAS_CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit III									
501	15	1247	401	1.85	0	+	+	+	+
501	15	1247	402	0	0.07	+	+	+	+
501	15	1247	403	0	0.10	+	+	+	+

For each **TIDFee**, a new fee transaction will be created (in this example 3 new transactions will be generated with TID of '401', '402', and '403') to reverse the original fees.

Using fee package PkgId = 501, incoming TIDAct = 15 and downgraded MAS_CODE = 999, all fees that should be charged for the transaction will be selected. In this example the following fees will be selected:

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Consumer Standard									
501	15	999	401	1.85	0	+	+	+	+
501	15	999	402	1.27	0.12	+	+	+	+
501	15	999	403	0	0.10	+	+	+	+

For each TIDFee, a new fee transaction is created (in this example, three new fee transactions are generated with TID of '401', '402', and '403') based on the downgraded MAS_CODE.

All transactions are logged. The incoming transaction is logged as a non-settlement transaction used to calculate all fees. The original fee transactions are reversed, and new fee transactions based on MAS_CODE 999 are generated. The results in the MAS_TRANS_LOG are shown:

Field[1]	Incoming non settled trans. (15 = Sale)	Credit Discount Fee Original Fee Reversed	Pass Through Fee Original Fee Reversed	Credit Transaction Fee Original Fee Reversed	Credit Discount Fee New Fee Calculated	Pass Through Fee New Fee Calculated	Credit Transaction Fee New Fee Calculated
TRANS_SEQ_NBR*							
TRANS_SUB_SEQ		new	new	new	new	new	new
TID*	15	401	402	403	401	402	403
MAS_CODE *	999	999	999	999	999	999	999
CARD_SCHM *	05						
AMT_ORIGINAL *	100						

Field[1]	Incoming non settled trans. (15 = Sale)	Credit Discount Fee Original Fee Reversed	Pass Through Fee Original Fee Reversed	Credit Transaction Fee Original Fee Reversed	Credit Discount Fee New Fee Calculated	Pass Through Fee New Fee Calculated	Credit Transaction Fee New Fee Calculated
CURR_CD_ORIGINAL *	840						
NBR_OF_ITEMS *	1						
ENTITY_ID *							
ENTITY_TYPE *							
EXTERNAL_ENTITY_ID							
ADD_AMT1 *	0						
ADD_CNT1 *	0						
ADD_AMT2 *	0						
ADD_CNT2 *	0						
FILE_ID							
BATCH_ID							
ACTIVITY_DATE_TIME *							
ENTITY_ACCT_ID							
AMT_BILLING	0	-1.85	-0.07	-0.10	1.85	1.39[2]	0.10
CURR_BILLING							
DATE_PRICED	system date	system date	system date	system date	system date	system date	system date
DATE_TO_SETTLE	calc. using delay f. from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN
CURR_RATE							

Field[1]	Incoming non settled trans. (15 = Sale)	Credit Discount Fee Original Fee Reversed	Pass Through Fee Original Fee Reversed	Credit Transaction Fee Original Fee Reversed	Credit Discount Fee New Fee Calculated	Pass Through Fee New Fee Calculated	Credit Transaction Fee New Fee Calculated
SETTL_FLAG	N	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN
POSTING_ENTITY_ID	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN
POSTING_ENTITY_TYP	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN
GL_ACCT_FROM	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN
GL_ACCT_TO	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN	from SETTL _PLAN
PERCENT_AMT	null	calc. amt using percent _rate	calc. amt using percent _rate	calc. amt using percent _rate	calc. amt using percent _rate	calc. amt using percent _rate	calc. amt using percent _rate
PER_ITEM_AMT	null	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate
PERCENT_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees
PER_ITEM_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees	from Mas Fees
CALENDAR_ID_TO_USE *							

Field[1]	Incoming non settled trans. (15 = Sale)	Credit Discount Fee Original Fee Reversed	Pass Through Fee Original Fee Reversed	Credit Transaction Fee Original Fee Reversed	Credit Discount Fee New Fee Calculated	Pass Through Fee New Fee Calculated	Credit Transaction Fee New Fee Calculated
SETTL_PLAN_TO_USE *							
FEE_PKG_TO_USE *							
EXTERNAL_TRANS_ID							
PAYMENT_SEQ_NBR	null						
INVOICE_NBR	null						
PAYMENT_TERMS	null	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
CHEQUE_NBR	null						
TRANS_ROUTING_NBR							
MAS_CODE_DOWNGRADE*	1247	1247	1247	1247	1247	1247	1247
EXPECTED_MAS_CODE *	null						
SUSPEND_REASON	null						

1. Fields marked with a * are used to generate fee transactions.
2. (100 * 1.27%) + 0.12=1.39

Non-Qualification Flag Set to “Y”

Merchant "Shoe Store" has a Fee Package PkgId = 501 for the MasterCard card scheme. The package has three fees for the TID = 15 (Sale) and MAS_CODE=1247:

- Credit Discount Fee (401). A credit Credit Discount Fee is charged.
- Fixed Pass-through Fee (402). A Fixed Pass Through Fee is charged for fully qualified transactions.
- Credit Transaction Fee (Per Item Processing Fee) (403). A Credit Transaction Fee is charged.

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit III									
501	15	1247	401	1.85	0	+	+	+	+
501	15	1247	402	0	0.07	+	+	+	+
501	15	1247	403	0	0.10	+	+	+	+

In this case, package 501 does not have fees set for MAS Codes corresponding to Key Entered, Merit I and Consumer Standard rates. The non-qualification merchant setup processing of MAS is invoked to calculate non-qualification charges for the transactions that do not meet expected MAS_CODE. These charges are added to regular charges that apply to the expected MAS_CODE.

The following processing is performed:

1. The MAS_CODE corresponding to the best rate (1247) is set to be the expected MAS_CODE. Also, the non-qualification indicator for the merchant is set to “Y”.

   This means that if the incoming transaction contains the ‘MAS_CODE’ that is not expected for this merchant, calculation of the non–qual charge amount follows the non-qualification pricing rules:

   Additional non-qualification charges are calculated using a surcharge:

   Surcharge = sales_amt * (default rate for incoming MAS code (actual) % – default rate for expected MAS code (expected) %) + nbr_of_items(sales+credit)*(actual – expectedrate_per_item)

2. The difference between the actual and expected rate is calculated based on the default package settings[1].

1. The default package must be set for all merchants that have non-qual flag set to “Y”

The actual rate (rate for the incoming 'MAS_CODE') and the expected rate (rate for the expected 'MAS_CODE' for this merchant) is found in the default package (default PkgId[1] in the MAS_FEES table).

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
999	15	1247	402	1.36[1]	0.17	+	+	+	+
999	15	1332	402	1.87	0.22	+	+	+	+
999	15	1001	402	2.05	0.22	+	+	+	+
999	15	999	402	2.63	0.22	+	+	+	+

1. Please note that the rates in the default package do not need to be equal to the interchange rates. The rates here should be set so that they accommodate the difference in the expected and actual to match the non-qualification fee that we want to charge on top of the expected rate fees specified in the standard package under the expected MAS code.

Percent rate and per item rate define rates for the non-qual calculation. Amount and count method columns of the default fees are ignored in the non_qual calculation. Instead, amount and count methods are used from the 'surcharge' fee record. This means that the surcharge fee needs to be defined in the MAS_FEES whenever the non-qual indicator is set to "Y", as shown in the table:

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
501	402	1247	410	0	0	+	+	+	+

The surcharge fee is defined in the regular package (501 in our example) and will be calculated and potentially charged only as part of the non-qual charge processing.

The surcharge fee entry in the MAS_FEES table is linked through the 'TIDAct' to one of the regular fees that is charged for this merchant. The 'TIDFee' in this case is '402', that is the regular fee for that incoming transaction. Surcharge '410' is the 'conditional' fee that will be charged only in some cases.

The 'MAS_CODE' column in MAS_FEES contains the expected 'MAS_CODE' for the merchant.

'TIDFee' contains the TID code for the surcharge fee (for example '410').

3. Using the TIC code for the surcharge fee, the fee transaction is processed and sent to billing.

1. The Package ID of the default fee package will be taken from the configuration file. This package will be probably assigned to the top level-Entity in the merchant hierarchy.

The results of the non-qual calculation is summarized in this transaction. In other words, non-qual charges (or upgrade) are charged as a surcharge fee.

If there is no explicit surcharge fee, the surcharge record must be present as well (with 0 in the rate columns) in order to calculate non-qual charges using the default rate table.

Case 2—Non-Qualification Processing

This case covers the example of a transaction that does not meet the expected qualification rate for the merchant (the actual MAS_CODE is different from the expected MAS_CODE).

A MasterCard Sale Transaction comes from the merchant "Shoe Store". The transaction does not qualify for the best interchange rate corresponding to MAS_CODE = 1247. The actual interchange rate corresponds to MAS_CODE = 1001.

For the merchant "Shoe Store" and card scheme MasterCard, based on the TIDAct = '15' there is no record found in the package 501 with MAS_CODE = 1001. Therefore, the expected MAS_CODE = 1247 is used to find the package and start pricing.

Using fee package PkgId = 501, incoming TIDAct = 15 and expected MAS_CODE = 1247, all fees that should be charged for the transaction will be selected, as shown in the table:

PkgId	TIDAct	MAS_ CODE	TIDFee	%	PerItem	am	cm	am1	cm1
Sale Transaction, Merit III									
501	15	1247	401	1.85	0	+	+	+	+
501	15	1247	402	0	0.07	+	+	+	+
501	15	1247	403	0	0.10	+	+	+	+

Fee charges are calculated. The percent rate and rate per item are applied to the base transaction amount and count. If percent rate and rate per item are both set as 0 values, a fee transaction is not generated. For each TIDFee, a new fee transaction is created (in this example three new transactions are generated with TID of ‘401’, ‘402’, and ‘403’).

The actual rate to be used for non-qual fee calculation is found in the default package (999) under the actual MAS_CODE (1001). The expected rate used for non-qual calculation is found in the default package under expected MAS_CODE = 1247. The difference in actual and expected rate is used for the non-qual fee calculation. If applicable, a surcharge fee is also added.

PkgId	TIDAct	MAS_CODE	TIDFee	%	PerItem	am	cm	am1	cm1
999	15	1247	402	1.36	0.17	+	+	+	+
999	15	1001	402	2.05	0.22	+	+	+	+
				0.69	0.05				

The surcharge TID (410) is used to log the non-qual fee transaction to the transaction log. In this case, the transaction log stores the original transaction, all fee transactions as per the expected MAS_CODE and the non-qual fee transaction. The MAS_TRANS_LOG is populated as follows:

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee	Non-qual fee transaction
TRANS_SEQ_NBR*					
TRANS_SUB_SEQ		new	new	new	new
TID*	15	401	402	403	410
MAS_CODE *	1001				
CARD_SCHM *	05				
AMT_ORIGINAL *	100	100	100	100	100
CURR_CD_ORIGINAL *	840				
NBR_OF_ITEMS *	1				
ENTITY_ID *					
ENTITY_TYPE *					
EXTERNAL_ENTITY_ID					
ADD_AMT1 *	0				
ADD_CNT1 *	0				
ADD_AMT2 *	0				
ADD_CNT2 *	0				

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee	Non-qual fee transaction
FILE_ID					
BATCH_ID					
ACTIVITY_DATE_TIME *					
ENTITY_ACCT_ID					
AMT_BILLING	97.24	1.85	0.07	0.10	0.74
CURR_BILLING					
DATE_PRICED	system date	system date	system date	system date	system date
DATE_TO_SETTLE	calc. using delay f. from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN	next_settl_date from SETTL_PLAN
CURR_RATE					
SETTL_FLAG	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
POSTING_ENTITY_ID	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
POSTING_ENTITY_TYP	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_FROM	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN
GL_ACCT_TO	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN	from SETTL_PLAN

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee	Non-qual fee transaction
PERCENT_AMT	null	calc. amt using percent _rate	calc. amt using percent _rate	calc. amt using percent _rate	calc. non-qual charges: percent amt.
PER_ITEM_AMT	null	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. amt using per_ite m_rate	calc. non_qual charges: per item amt.
PERCENT_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees	calc. percent rate (actual – expected + surcharge)
PER_ITEM_RATE	null	from Mas Fees	from Mas Fees	from Mas Fees	calc. per item rate (actual – expected)
CALENDAR_ID_TO_USE *					
SETTL_PLAN_TO_USE *					
FEE_PKG_TO_USE *	501				
EXTERNAL_TRANS_ID					
PAYMENT_SEQ_NBR	null				
INVOICE_NBR	null				
PAYMENT_TERMS	null	from SETTL_ PLAN	from SETTL_ PLAN	from SETTL_ PLAN	from SETTL_PLA N
CHEQUE_NBR	null				
TRANS_ROUTING_NBR					

Field[1]	Incoming Transaction (15)[2]	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee	Non-qual fee transaction
MAS_CODE_DOWNGRADE*	null				
EXPECTED_MAS_CODE *	1247	1247	1247	1247	1247
SUSPEND_REASON	null				

1. Fields marked with a * are used to generate fee transactions.
2. The empty fields in the original transaction column are obtained either from the incoming transaction message or enriched by the validation module. The empty fields in the fee transaction columns are the same as the corresponding fields of the original transaction.

ISO Billing

Processing Scenario

Let us assume that the merchant "Shoe Store" is a plain vanilla retail merchant, and the following are true:

- The Merchant "Shoe Store" (Entity ID = 401011) has the Fee Package PkgId = 501 for the MasterCard card scheme.
- The Interchange Reimbursement Program is Merit III (1.36%) with MAS Code 1247.
- The Merchant Discount Rate is 1.85%.
- The original transaction amount is $100; TIDAct =15.
- The Regional Sales Office Entity ID = 401010; profit sharing rate = 0.1% of original transaction amount.
- The Regional Branch (parent entity) Entity ID = 401001; profit sharing rate = 0.2% of the original transaction amount.
- The Processor Entity ID = 401000; profit sharing rate = 1.55% of the original transaction amount (i.e., the rest of the fee).

Configuration Details

To facilitate ISO billing, the following setup is required:

- See “Merchant Hierarchy Setup” on page 244.
- See “Settlement Plan Setup” on page 244.
- See “Fee Package Setup” on page 245.

Merchant Hierarchy Setup

These records must be set up in the ACQ_ENTITY table, as shown in the figure:

- Regional Branch entity_id = 401001.
- Regional Sales Office entity_id = 401010 and parent_entity_id = 401001.
- Shoe Store entity_id = 401011 and parent_entity_id = 401010.

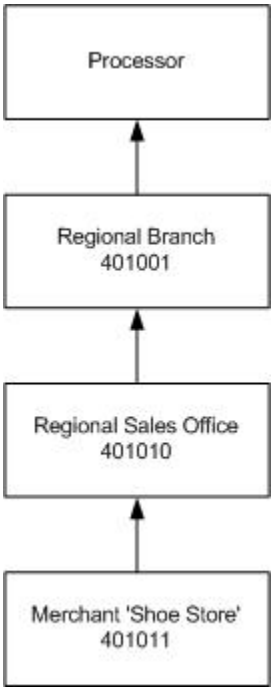


Figure 1 ISO Billing Sample Hierarchy

Settlement Plan Setup

In the SETTL_PLAN_TID table, the Shoe Store, the Regional Sales Office, and the Regional Branch use Settlement Plans 31, 310, and 301 respectively.

- Shoe Store SETTLEMENT PLAN (31) has the Parent ISO TID populated (491).

- Regional Sales Office SETTLEMENT PLAN (310) has the Parent ISO TID populated (492).
- Regional Branch SETTLEMENT PLAN (301) has no value in the Parent ISO TID.

SettlPlanId	Tid…	…	parent_iso_tid	…
SettlPlanId	Tid	…	parent_iso_tid	…
31	401		491	
310	491		492	
301	492			

Fee Package Setup

In the MAS_FEES table, these entries must be created:

- Shoe Store's Fee Package (Pkg Id=501) has an entry for the fee applied to Sale Transaction, Merit III MAS Code.
- Regional Sales Office's Fee Package (Pkg Id = 551) has an entry for the Profit Sharing Transaction (491), Merit III MAS Code. The entry contains a negative rate fee (-0.1%).
- Regional Branch's Fee Package (Pkg Id = 552) has an entry for the Profit Sharing Transaction (492), Merit III MAS Code. The entry contains a negative rate fee (-0.2%).

PkgId	TIDAct	MAS_CODE	TIDFee	%	PerItem	am	cm	am1	cm1
501	15	1247	401	1.85	0	+	+	+	+
551	491	1247	491	-0.1	0	+	+	+	+
552	492	1247	492	-0.2	0	+	+	+	+

Processing Flow

To apply ISO billing, the system will have to do the following:

1. Locate Fee Package 501 for the Shoe Store merchant based on the incoming Sale Transaction (TIDAct =15), MasterCard Card Scheme and Merit III MAS Code (1247).

   The TIDFee 401 applied to the Sale Transaction and the Merit III MAS Code are used to generate revenue to the Processor in this Fee Package, 501.

2. Locate the settlement plan for 401 that has a Parent ISO TID populated 491.

3. Locate the Shoe Store's parent entity in the Acquiring Entity table (Regional Sales Office).

   The fee package of this parent entity (551) has an entry for the ISO TIDAct (491). The profit fee transaction is generated (TIDFee = 491) using this fee package.

4. Locate the settlement plan for 491 that has a Parent ISO TID = 491.

5. Locate the Regional Sales Office's parent entity in the Acquiring Entity table (Regional Branch Office).

   The fee package of this parent entity (552) has an entry for the ISO TIDAct 491. The profit fee transaction is generated (TIDFee = 492) using this fee package.

6. Locate the settlement plan for 492.

   Since the Parent ISO TID is not populated, the ISO billing process is complete.

   After billing is done, the following three fee transactions are found in the MAS_TRANS_LOG:

   - Shoe Store's original transaction fee (TID = 401).
   - Regional Sales Office profit fee transaction (TID = 491).
   - Regional Branch profit fee transaction (TID = 492).

7. Post fees to the merchant entity and calculated 'profit' to the parent entities.

This is the distribution of the fees:

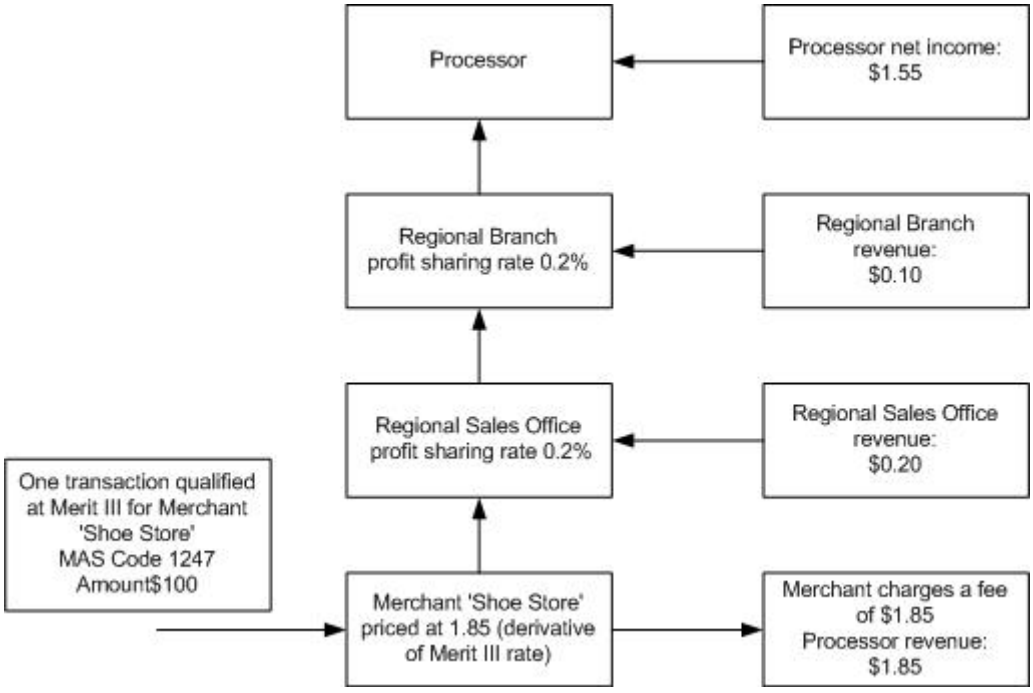


Figure 2 ISO Billing Sample Revenue

This is a snapshot of the values in the MAS_TRANS_LOG table. For your convenience, important values are highlighted.

Field	Incoming Transaction (15)	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
TRANS_SEQ_NBR	125017	125017	125017	125017
TRANS_SUB_SEQ	0	1	2	3
TID	15	401	491	491
AMT_BILLING	98.15	1.85	-0.1	-0.2
CURR_BILLING	USD	USD	USD	USD
DATE_PRICED	sysdate	sysdate	sysdate	Sysdate

Field	Incoming Transaction (15)	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
DATE_TO_SETTLE	Oct. 17,2001	Nov. 3, 2001	Nov. 3, 2001	Nov.3, 2001
CURR_RATE				
SETTL_FLAG	Y	Y	Y	Y
POSTING_ENTITY_ID	401011	401011	401010	401001
POSTING_ENTITY_TYP				
GL_ACCT_FROM	INC-305-Y-0	WIP-AR	WIP-AR	WIP-AR
GL_ACCT_TO	WIP-AP	R-400-Y-0	R-400-Y-0	R-400-Y-0
PAYMENT_TERMS	null	null	null	Null
PERCENT_AMT	null	1.85	-0.1	-0.2
PER_ITEM_AMT	null	0	0	0
PERCENT_RATE	null	0	0	0
PER_ITEM_RATE	null	0	0	0
FEE_PKG_TO_USE	501	501	551	552
SETTL_PLAN_TO_USE	31	31	310	301
MAS_CODE	1247	1247	1247	1247
CARD_SCHM	5	5	5	5
AMT_ORIGINAL	100.00	100.00	100.00	100.00
CURR_CD_ORIGINAL	USD	USD	USD	USD
NBR_OF_ITEMS	7	7	7	7
ENTITY_ID	401011	401011	401010	401001
EXTERNAL_ENTITY_ID				
ADD_AMT1	0	0	0	0
ADD_CNT1	0	0	0	0

Field	Incoming Transaction (15)	Credit Discount Fee	Pass Through Fee	Credit Transaction Fee
ADD_AMT2	0	0	0	0
ADD_CNT2	0	0	0	0
FILE_ID				
BATCH_ID				
ACTIVITY_DATE_TIME	Oct. 15,2001	Oct. 15,2001	Oct. 15,2001	
ENTITY_ACCT_ID	401001 991	401000 992	401010 999	401 001 999
CALENDAR_ID_TO_USE	null	null	null	null
EXTERNAL_TRANS_ID	7770017	7770017	7770017	7770017
PAYMENT_SEQ_NBR				
INVOICE_NBR	null	null	null	
CHEQUE_NBR	null	null	null	
TRANS_ROUTING_NBR				
MAS_CODE_DOWNGR	null	null	null	
EXPECTED_MAS_CODE	null			
TRANS_DESCRIPTION		null	null	
SUSPEND_REASON	null	null	null	

Glossary

ACH	Automated Clearing House, for electronic distribution payments.
acquirer	A company which processes credit card transactions for direct customers and submits them to the credit card association, and has the ability to manage the settlement with the customer.
acquiring entity	An entity for whom activity and fees are processed.
activity transactions	Transactions originating from a source external to *IST/MAS* and posted to an entity's account. Activity transactions can be either settled or non-settled. Settled transactions are posted to an entity's account and are settled via a mechanism external to *IST/MAS*. Non-settled transactions are posted to an entity's account only for reporting purposes. Examples of activity transactions are as follows: sales, credits, authorizations, chargebacks.
agent	A company that contracts with the owner of *IST/MAS* for credit card processing for multiple small merchants.
association	A credit card provider. Examples VISA, MasterCard.
authorizations	Transactions that are a result of an authorization request on a credit card transaction.
bank account	An account residing at the bank where money can be paid in or paid out.
batch summary	A record created at the start of a batch and used to maintain summary information about the batch.
calendar	A work schedule for an institution.
card schemes	Organization which owns one or more card brands and operates one or more card systems. The organization typically determines and governs the operating and processing regulations which are unique to this card scheme. Example: Visa, MasterCard, Europay, Amex, JCB, Din-ers, Discovery, Proprietary Card Brand, etc.

chargebacks	There is a separate group and system for chargeback processing. When a merchant disputes a charge, a new transaction is created with a new MAS code for the chargeback.
DDA	Acronym for Direct Deposit Account.
deposit	The value of transactions that require settlement coming in from Transaction Clearing for a Merchant.
Direct Deposit Accounts	An account for automatic deposit of payments.
entity account	A merchant entity's settlement account.
fee transactions	Transactions generated as charges for processing an activity transaction.
general ledger	Central accounting record of an organization summarizing changes in financial positions as transactions are posted during an accounting period.
interchange rates	Amount to be charged per transaction by the Association based upon transaction qualification.
interchange	The exchange of transaction data between acquirers and issuers in accordance with bylaws and rules.
invoice	A record of goods or services provided and the amount charged for them, sent as a request for payment.
ISO	Acronym for Independent Sales Organization (see Agent).
MAS codes	A code that uniquely identifies the transaction category. It contains the card type and qualification level to indicate the rates to be charged for a fee transaction billed to a Merchant Entity.
MAS	Acronym for Merchant Accounting System.
MCC	Acronym for Merchant Category Code. A code designating the principle trade, profession, or line of business in which a merchant is engaged. The acronym is used interchangeably with SIC.
merchant (corporate)	A company that contracts with the owner of *IST/MAS* for the clearing of transactions.
merchant location	A sales outlet or store where a cardholder initiates a financial transaction.
NACHA	An electronic transfer system developed and maintained by NACHA - The Electronic Payments Association.
network rates	Amount charged by the network provider during the processing of a debit transaction.
non-activity charges	Internally generated charges billed to merchant entities on a periodic basis.

payment date	The date on which an incoming transaction is paid.
payment delay factor	A number used to delay the date of a merchant payment.
payment	The action of crediting a merchant's bank account.
posting entity	An entity responsible for settlement of a transaction.
pricing	The process of determining fees to be charged as a result of processing an activity transaction.
processing date	The date on which *IST/MAS* releases the transaction into Interchange.
qualification level	The level assigned to a transaction, based on association regulations.
rate groups	The types of rates that can be attached to the MAS code (Interchange, Assessment, etc.).
rate re-assessment	Process by which entity performance is gauged over a period of time, and hence the rates that are applied in fees. Note that a fee cannot be re-assessed unless it is in an unique fee package.
remittance	Payment towards satisfaction of a debt.
residual	The amount due to an agent for the activity of an account per contract definition of terms.
SIC	Acronym for Standard Industry Code (see MCC above).
special rate	Charges currently specific to airline credit card transactions.
TID activity	Code specifying the type of transaction.
transaction code	Code identifying whether a transaction is a sale or credit.
transaction	Action between a cardholder and a merchant that results in activity on the cardholder account. Once transactions are acquired for interchange processing, this activity results in a credit or debit charge to a merchant's account.
UA adapter	A map definition used to translate from one format to another.
Wire	A method of transmitting payments directly to a bank.

Index

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

Statement of Confidentiality

The information contained in or supplied with this document is submitted solely for the purpose of evaluating the products and services of FIS, and/or its affiliates and subsidiaries. The information contained in or supplied with this document, in its entirety, is the confidential and proprietary information of FIS, and it may not be copied by or disclosed to any person or entity (other than to the intended recipient), without the prior written consent of FIS. With or without FIS' prior written consent, FIS accepts no liability whatsoever for any consequences arising from the reproduction of the information contained in or supplied with this document, or from its disclosure to any person or entity, including to the intended recipient. FIS additionally accepts no liability for the use of the information contained in, or supplied with this document, by the intended recipient, or by any other person or entity, with or without FIS' express prior consent. The intended recipient shall not use any part of the information contained in, or supplied with this document, in any way to the competitive disadvantage of FIS, and will take all steps designed to assure its compliance with this provision.

This proposal is neither an offer nor intended by FIS, upon acceptance by the intended recipient, or otherwise, to create a binding agreement with FIS. Such an agreement shall be reflected only by a definitive contract, executed by both parties.

Trademarks

All other trademarks are the property of their owners.

Company, product, and service names used by FIS within, or supplied with this document may be trademarks or service marks of other persons or entities.

Copyright