# FileAct – Concepts and Implementation

## Intended Audience

This document gives an overview of the FileAct Messaging Interface of BOX Messaging Hub (BOX). It is intended to be read by business managers, system administrators and systems architects. More detailed information on specific topics can be obtained by contacting Intercope through any of the addresses on the final page of the document.

## Management Summary

**High volumes of FileAct messages**

In addition to the traditional FIN-service, FileAct is becoming increasingly important for the exchange of financial messages between financial institutions via SWIFT. In Europe all national credit transfer and direct debit schemes must be replaced by SEPA Credit Transfer (SCT) and SEPA Direct Debit (SDD) by February 1st, 2014, at the latest. This implies that very large volumes of FileAct messages have to be sent and received by financial institutions and service providers.

**Multiple interfaces for business applications**

BOX Messaging Hub (BOX) includes a SWIFT certified Messaging Interface for the FileAct protocol which extends the scope of functions mandated by SWIFT for these services in several areas. Firstly, all flavours of the protocol, such as store and forward or real-time transmission and push or pull mode, are supported and can be configured and used in a very flexible way. Secondly, the communication with business applications is not restricted to the most commonly used interfaces based on IBM WebSphere MQ, but for situations where these facilities are not available, or desired, alternative interfaces can be used where the information exchange is based on a relational database or simple files.

**Flexible format conversion and message mapping profiles**

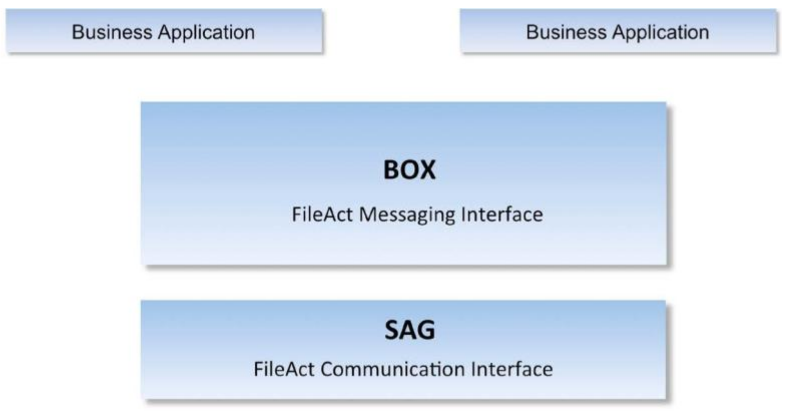
Because of BOX's flexible conversion plugins all the interfaces are capable of understanding virtually any type of message format and generating valid FileAct messages from these formats using different message mapping profiles. So that the complexity of protocol related information in the FileAct protocol is completely shielded for business applications.

**Highest performance**

Several benchmark tests show that even the File Act data volumes handled by the largest financial institutions and service provides can be processed by one BOX instance immediately without any delay, bottleneck or throughput issue.

## System Overview

### A dedicated messaging interface FileAct



The major components for the FileAct message flow are:

- Business applications generating and receiving ISO 20002 formatted financial messages and files
- BOX providing the protocol elements mandated by SWIFT for the messaging interface
- SAG providing the communications interface including the link to SWIFTNet

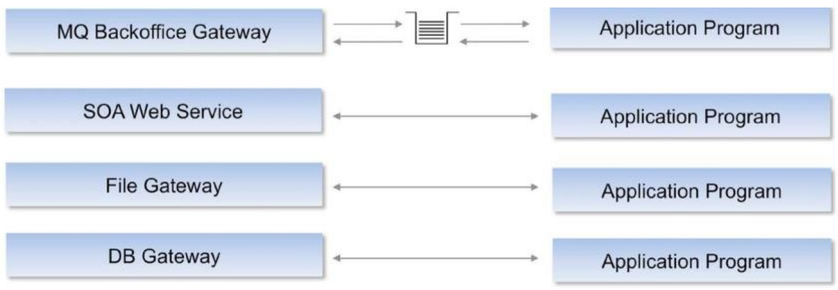
In this scenario BOX provides a dedicated messaging interface which implements the protocol specific requirements of FileAct so that business applications do not need to handle this communication layer. This includes retry handling, handling of possible duplicates, security aspects, session monitoring etc.

For the storage of the payload of FileAct messages in BOX one of the following two methods can be selected:

- As file in the file system
- As BLOB in the database

## Business Application Interfaces

### Multiple Integration Options



BOX offers several interfaces (gateways) for business applications to exchange FileAct messages and delivery notifications with BOX including:

- The MQ (IBM WebSphere MQ) Gateway
- The Database Gateway
- The File Gateway

### The MQ gateway

The MQ gateway is the most commonly used interface to exchange messages between business applications and BOX as IBM WebSphere MQ has evolved into a de facto standard for the exchange of SWIFT messages. The sending business application puts a message to a MQ- queue, the MQ-manager forwards this to a queue which is read by the BOX MQ gateway and BOX processes the message.

When the “FACTBA” plugin of BOX is used transmission parameters can be provided in the RFH2 header or the MQMD header.

As an alternative the highly customizable XSLT plugin of the MQ gateway allows the automatic translation of various flavours of message formats into valid FileAct structures. This is a feature which is particularly useful for generating elements of the message header from data provided by the business applications.

Instead of providing transmission parameters in the header or XML structure of the MQ message these values can be stored in BOX Submission Profiles. It is also possible to combine these two methods by specifying one or some parameters such as the recipient of the message in the message header and access the remaining parameters from predefined Submission Profiles.

When the payload of the transaction is transferred as part of the MQ message MQ grouping/segmentation is supported to split large data volumes into several MQ messages.

In addition, for FileAct a file drop facility is provided which can be used in both directions, where the files to be sent to SWIFT or to be processed by the business application are stored in configurable file system directories, and the MQ messages mainly contain a pointer to that file and further delivery parameters.

Delivery notifications and messages received from SWIFT are routed by BOX to specific business applications, put into a corresponding MQ queue, transferred by the MQ queue manager and processed by the business application.

**The database Gateway**

In addition, or as an alternative to the MQ interface the BOX database gateway can be deployed for information exchange between business applications and BOX. In this case the data to be transferred is stored in the tables of an RDBMS such as DB2 or Oracle Database instead of a MQ queue. The structure of the database table – the format and the content of the table columns - can be different for different business applications as the BOX database gateway includes configuration options to map this data to BOX internal data structures. This is functionally equivalent to the XSLT plugins for the MQ gateway and the file gateway. In addition, for FileAct a file drop facility can be used in both directions, where the files to be sent to SWIFT or to be processed by the business application are stored in configurable file system directories and the database table mainly contains a pointer to that file and further delivery parameters.

**The file Gateway**

A third option for exchanging messages between business applications and BOX is the file gateway. With this integration option the information is simply stored in files by either the business application (for messages to be sent to SWIFT) or by BOX (for messages and delivery notifications received from SWIFT) configurable file system directories. Also, this interface provides a highly customizable XSLT plugin for Transmission Parameters as well as the usage of Submission Profiles. Each time files received from SWIFT are transferred from BOX to business applications it optionally possible to start external processes and so to initiate further message processing. This method is e.g. used by BOX to transfer files via Connect: Direct.

**MERVA SDI – SDO**

Like for FIN messages BOX also supports the MERVA SDI and SDO batch functions for FileAct allowing MERVA users to continue using this interface without changing existing applications.

With any of these interfaces the payload can be compressed using any of the standard compression methods such as gzip or GLib.

**Manual message handling**

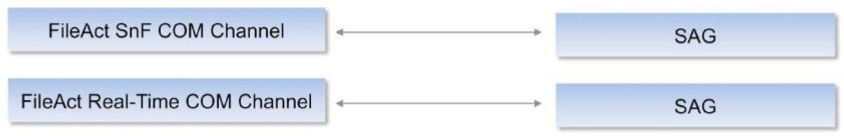
Files received from SWIFT can be manually stored on disk – a function in particular useful when directory information has been requested from SWIFT. Files can also be sent manually – a function which had previously been provided by SAG 6 but is no more available with SAG 7.

In general, all message processing functions provided for FIN messages are also available for FileAct messages including:

- Message Creation
- Message Authorization
- Message Repair
- Configurable Journals
- Applications Queues
- Automatic and manual printing

## Communication Channels

### All protocol flavours



SWIFT provides several options for the FileAct protocol. The protocol can be used in store and forward or in real-time mode and in addition a push or pull delivery mode can be selected, when a receiver opens a SWIFT queue.

### Store and forward mode

In store and forward mode, the Messaging Interface opens an input channel and thereby establishes an input session with SWIFT. Messages are transferred via this input channel and safely stored by SWIFT. The delivery of the messages to the recipients, and of delivery confirmations to the sender, happens asynchronously and the sending of messages does not depend on the immediate availability of services with the receiving institution.

To receive messages from SWIFT, queues can be opened in either push delivery or pull delivery mode.

### Push mode / Push mode

In push mode SWIFT automatically delivers traffic to the receiver. Any traffic pending on the receiver’s queue is automatically and immediately delivered. This mode thus guarantees that traffic is delivered without unnecessary delay as soon as the receiver has started a session for the relevant queue.

In pull mode SWIFT does not send anything automatically to a receiver, but the receiver has to send a pull request to SWIFT for each message he wants to retrieve from the queue. SWIFT then delivers one message (if available), the receiver acknowledges the receipt of the message and sends the next pull request. This mode is recommended by SWIFT only for low message volumes, and for use with manually operated messaging interfaces.

### Real-time mode

In real-time messaging mode SWIFT establishes an online connection to the receiver of the message. The sender receives either an immediate acknowledgment from the receiver to indicate the message was received, or an error message. This mode requires both the sender and receiver to be connected at the same time to SWIFTNet.

### Configurable communication channels

BOX supports the exchange of FileAct messages in store & forward mode as well as in real-time mode together with push and pull mode to receive messages from SWIFT. The connectivity to SWIFTNet is implemented in BOX as communication channels and these different protocol options can be specified for each communication channel by means of configuration parameters.

### FileAct operations

For FileAct this includes the following modes:

- Real-time
  - Send Files (Send PutFileRequest)
  - Receive Files (Receive PutFileRequest)
  - Send Files (Receive GetFileRequest)
  - Receive Files (Send GetFileRequest)
  - Scheduled transfer possible
  - Monitoring of File Transfers
- Store and Forward
  - Send Files
  - Receive Files

In addition, for FileAct the Remote File Handler, provided by SWIFT, is supported, thus avoiding the necessity to store confidential data outside of the DMZ.

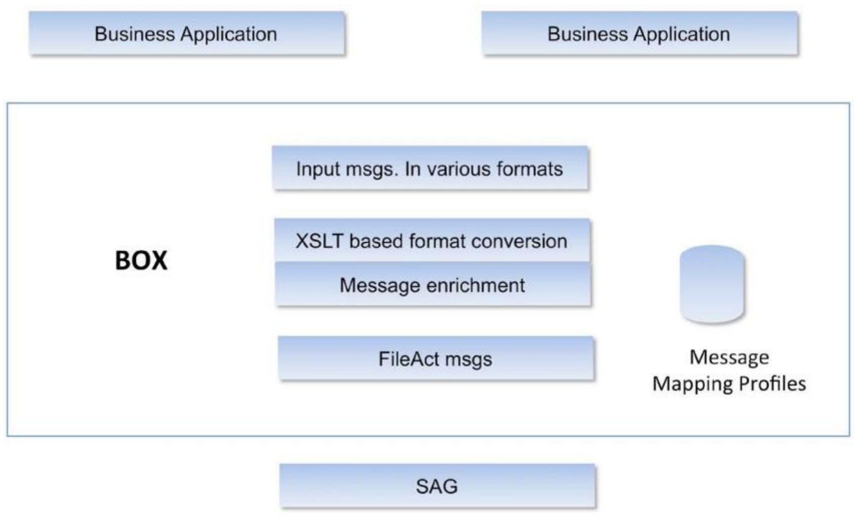
### File bulking and debulking

With the current BOX release 16 additional functionality will be provided for bulking and debulking for the FileAct interface. This feature includes options to:

- Collect messages of the same type within an application queue
- Use of scheduling to trigger the bulking process
- Modules to calculate a checksum (e.g. SEPA)

## Message Enrichment

### Message mapping profiles



### Protocol related header data

Business applications typically provide and process only the business-related parts of FileAct messages (the business payload). However, FileAct messages require in addition various header elements related to message routing, delivery instructions and security aspects of the SWIFT protocols. These elements are typically constructed by BOX based on different FileAct service profiles, which can be loaded and maintained within the BOX GUI.

### XSLT technology

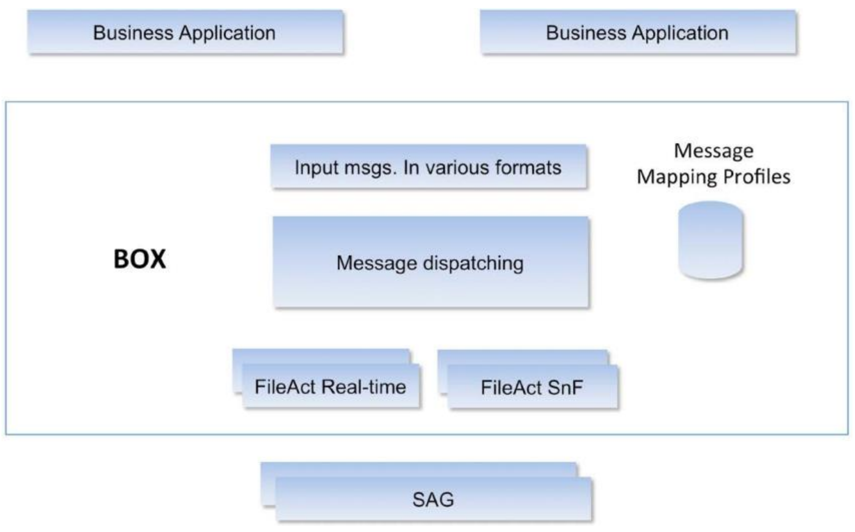
If a profile is to be used, it can be specified in the message generated by the business application or be determined by configuration data such as the receiving channel. The combination of highly flexible conversion tools for the input data stream based on XSLT technology and the flexible setup and access options of different profiles allow the construction of complete I FileAct messages in a very flexible way from virtually any input data format provided by the business applications.

Similarly, when messages are received from SWIFT, BOX converts the FileAct message into a business application specific format stripping protocol related information which is not needed for processing by the business applications.

It is important to note that these advanced data extraction and conversion options are an integral part of the product and do not require any additional external component and do not incur any additional license cost.

## Message Dispatching

### Flexible channel selection



As described above the different flavors of the FileAct protocols are implemented in BOX as communication channels. When a message is received from a business application and formatted by the message conversion layer BOX dispatches the message to an appropriate communication channel for delivery. Just as in message conversion, the dispatching criteria can be included in the message generated by the business application and / or be defined in a BOX profile. Furthermore, the profile can be specified either in the message generated by the business application or constructed from configuration data.

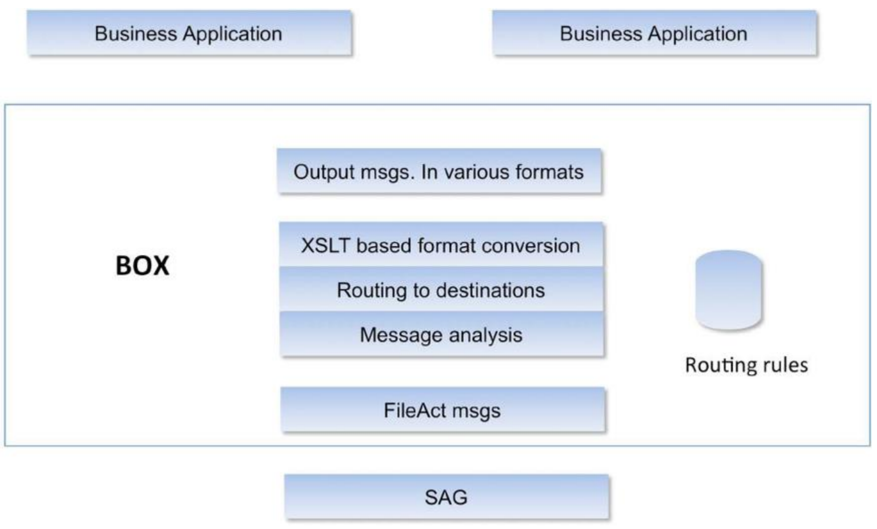
### Load balancing and availability

The dispatching algorithms are in particular optimized to:

- Support several SAGs
- Optimize load balancing
- Handle fail-over of SAGs
- Maximize throughput
- Guarantee uninterrupted availability

## Message Routing

### Configurable routing rules



FileAct messages received from SWIFT typically have to be delivered to business applications for further processing. In addition, specific messages may require manual intervention or automatic printing. The criteria for these routing decisions may be complex and require the analysis of the content of the payload of the message as well as of various header fields. BOX provides sophisticated facilities for this purpose as part of the product and includes a graphical user interface to define these algorithms. All elements of a message can be analyzed, including substrings and the use of regular expressions, in arbitrary logical operations to determine any number of routing destinations for a message or delivery confirmation.

### Generic routing table analysis

The routing rules can also be defined with a table-based method called “Generic Routing Table Analysis”. This tool does not require any programming knowledge and at the same time provides documentation of the routing rules which can easily be understood by business users.

## Message Audit Log and Monitoring

**Complete message audit log**

In BOX the status and complete history of all messages is stored in a single database avoiding the need to perform laborious searches in several data sources followed by reconciliation between these data sources. This data can be accessed under various filter criteria within a customizable graphical user interface which allows searching for specific messages under various filter criteria.

**Real-time monitoring**

For real-time monitoring of the communications layer and the tracing of all messages sent and received via FileAct the following tools are included in the product:

- The “Session Layer Channel Monitor” provides status information and various other data from the SWIFT communication layer in real-time for each LT session
- In addition, the “Channel Session History” provides comprehensive information about all messages exchanged through a search facility using various criteria

## Performance and Throughput

**Daily 20 million SEPA messages**

A large European bank calculated that from February 2014 on daily 20 million SEPA messages with a data volume of some 20 GB will have to be processed daily on top of other FileAct transactions and there were concerns that these data volumes can be handled in a reasonable time- frame.

Performance tests on an IBM Z Enterprise System EC12 in the IBM Client Center Germany revealed that BOX can process these data volumes in 6 minutes not considering transmission times or response times of SWIFT and / or EBA Clearing.

**Successful EBA Clearing testing**

From October 7 to October 10, 2013 SWIFT and EBA Clearing conducted SEPA volume testing with the participation of several large financial institutions. A large BOX customer successfully participated in these tests running one BOX instance on an Oracle Sun Server without facing any problems. Transmission times for SEPA files reached up to 1095 KB / seconds and were only limited by the available bandwidth of 10 Mbit / second and the response times of EBA Clearing.

**Virtually unlimited data volumes**

These tests show that BOX can handle virtually unlimited data volumes of SEPA or other FileAct messages with moderate resource consumption and that the processing time of BOX for SEPA files is insignificant in relation to data transmission and response times of EBA Clearing and / or SWIFT.