



Service

Partners

SWIFTReady Alliance Add-on

Label Criteria 2012

This document provides a structured and detailed view of the criteria that an add-on application must fulfil in order to be granted the SWIFTReady Alliance Add-on 2012 label.

27 January 2012

Table of Contents

Preface	3
1 SWIFTRReady Alliance Add-on	4
1.1 SWIFTRReady Programme.....	4
2 Alliance Add-on	5
3 Alliance Add-on Typology	7
3.1 Message Repair	7
3.2 Business Activity Monitoring (BAM)	7
3.3 Anti-Money Laundering and Sanction Lists Screening	7
3.4 Operational Activities	7
4 Alliance Developers Toolkits	8
4.1 Alliance Developers Toolkit.....	8
4.2 Alliance Integrator Developer Kit	9
5 SWIFTRReady Alliance Add-on	10
5.1 Scope	10
5.2 Changes compared to 2011.....	10
5.3 Add-on Typology	11
6 Standards	12
6.1 MT	12
6.2 MX.....	13
6.3 ISO 20022	13
7 SWIFT Messaging Protocols	15
7.1 FIN.....	15
7.2 InterAct.....	15
7.3 FileAct	15
8 Alliance Access Integration	17
8.1 Alliance Developers Toolkit Developer Guide Compliance	17
8.2 Alliance Access Message Recovery	18
8.3 Alliance Access Add-on Installation and Configuration	19
8.4 Alliance Access User Documentation	20
9 Alliance Integrator Integration	21
Legal Notices	22

Preface

Purpose of this document

This document provides a structured and detailed view of the criteria that an add-on application must fulfil in order to be granted the SWIFTReady Alliance Add-on 2012 label.

Intended audience

This document is for the following audience:

- application vendors considering the certification of a product
- SWIFT users seeking an overview of the SWIFTReady Label contents.

The audience must be familiar with the SWIFT world from both technical and business perspectives.

Related documentation

The following documents can be found at http://www.swift.com/partners/certify_your_application

- *SWIFTReady Application Programme Overview*
The document provides an overview of the SWIFTReady programme, including the benefits to join for application vendors. It also explains the SWIFTReady validation process, including the technical, functional and customer validations.
- *SWIFTReady Technical Validation Guide*
The document provides a detailed description of the technical validation processes for each label.

1 SWIFTReady Alliance Add-on

SWIFT has developed this programme to certify third-party applications and middleware products that support SWIFT messaging systems and standards, and integrate with SWIFT through Alliance interfaces using a dedicated adapter or through the Application Programming Interface (API).

Alliance Add-ons are software components that beef up the value proposal of Alliance interfaces in providing the business features required by the financial market. These added-value features range from Anti-Money Laundering (AML), financial fraud detection, black-list filtering, duplicate checking to STP enrichment, Business Activity Monitoring (BAM), Exceptions and Investigations, Straight-Through Processing (STP) enhancer, messaging data services including translation and validation, in addition to other operational activities.

1.1 SWIFTReady Programme

The SWIFTReady Label programme extends throughout the entire financial application chain to include Trade, Treasury, Payment, Corporate, and Securities applications.

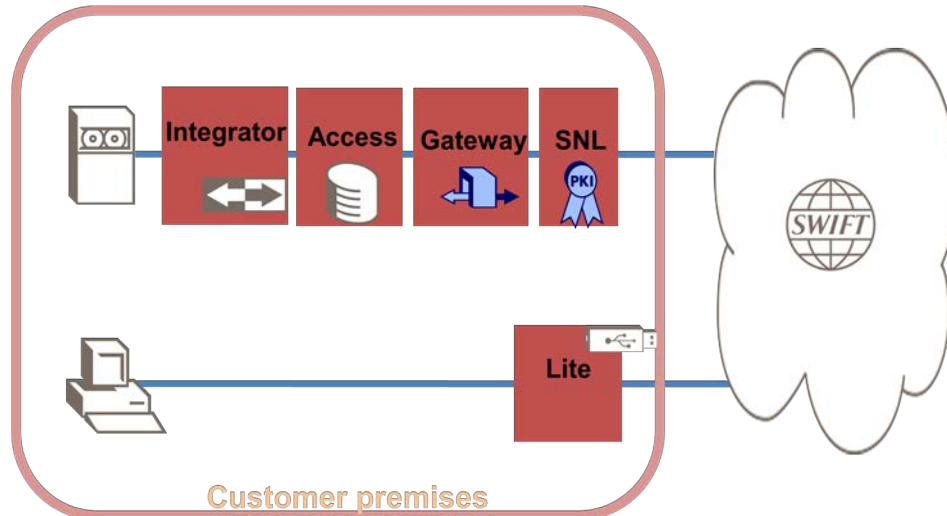
Each SWIFTReady label defines a set of criteria that are reviewed every year to ensure that the software remains aligned with the financial market evolution and with customer needs.

These criteria are designed to reflect the capability of a vendor product to provide message processing automation in a SWIFT context, and to support Straight-Through Processing in order to increase customer value, limit customisation needs and cost, and reduce time to market.

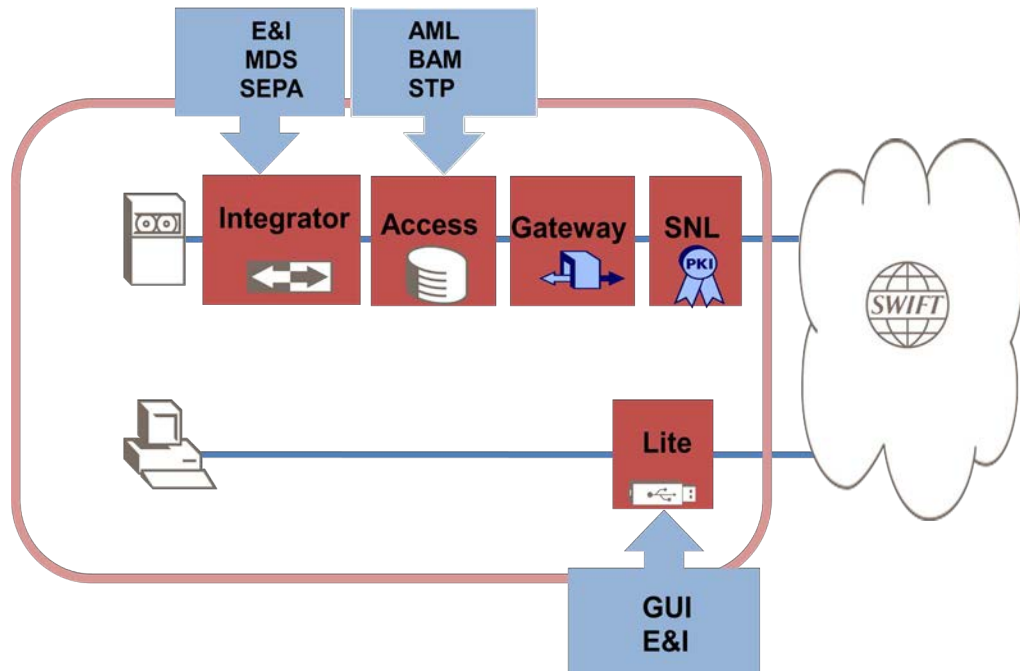
2 Alliance Add-on

Applications can be deployed over Alliance interfaces in two modes:

- as a **back-office application** connecting to Alliance interfaces through the various Alliance adapters (for example, MQ File or Simple Object Access Protocol (SOAP)). Messages are typically generated by the application and are routed through the Alliance interfaces, where business processes can enrich, change, validate and route them. In this model, applications are the source or sink of the messages.



- as a **business add-on over Alliance** interfaces, which acts upon passing messaging transactions to perform various tasks on the messages. These tasks range from messaging data services such as validation, transformation and enrichment to reporting, archiving, business activity monitoring, Exceptions and Investigations management, Anti-Money Laundering, or duplicate checking. The add-on can be deployed as a web service or service-oriented architecture (SOA) component, and is activated by the business process engine of the Alliance interfaces.



Add-ons do not generate or terminate messages. Instead, add-ons act on the messaging transactions generated by the back-office applications. Actions on the messages will typically modify the message contents (that is, translate, transform, enrich, add security info) or the message routing scheme (that is, validate, approve, reject, send to manual check, copy, archive, monitor).

The **SWIFTReady Alliance Add-on** focuses on business add-ons that can be deployed over Alliance Access and/or Integrator:

- **Alliance Access** is the general purpose messaging interface used by SWIFT customers to exchange messages and files. In the past years, many vendors have been developing add-on components using the *Alliance Developers Toolkit* to enrich the features of Alliance Access.
- **Alliance Integrator** is an extension of Alliance Access that eases the integration of back-office application. Integrator provides a Business Process Management (BPM) framework (the Alliance Integrator controller) that can be enriched with vendor components that are deployed as web services.

This document provides an overview of the Alliance Developers Toolkit and of the Alliance Integrator web services API.

3 Alliance Add-on Typology

3.1 Message Repair

The Alliance Add-on enables editing messages in order to repair them in exceptional cases. This holds for messages that do not pass validation and need urgent repair prior to sending them to SWIFT.

Messages manipulation can be restricted to the user with dedicated security profile. Actions taken on messages are then logged, and messages are stored before and after manipulation for audit purposes.

The message editor is format-aware. In particular, field names are expanded, and field options are provided depending on expected type (for example, code words, dates, BIC look-up).

3.2 Business Activity Monitoring (BAM)

Business Activity Monitoring tracks processing chains at both technical and business levels. This dual competence helps financial institutions maintain control of their data flows, through detailed views of inter-application data transformation. Through multiple connectors positioned at all levels of the processing chain, Business Activity Monitoring ensures end-to-end supervision and monitoring, guaranteeing complete control of all data flows.

A business transaction is made of messages flowing from back-office to SWIFT (for example, payment initiation) and back from SWIFT to the back-office (for example, payment confirmation). Message events (that is, message generated, validated, repaired, sent, acknowledged, delivered, reconciled) can be tracked and displayed in the dashboard for monitoring purposes. Rules can be generated to associate alarms to some events (for example, message DeIN note received after 3 days). Reports must be generated.

3.3 Anti-Money Laundering and Sanction Lists Screening

Anti-money laundering refers to the legal controls that require financial institutions and other regulated entities to prevent or report money laundering activities.

Today, all financial institutions globally are required to monitor, investigate, and report transactions of a suspicious nature to the financial intelligence unit of the central bank in the respective country. For example, a bank must perform due diligence by having proof of a customer's identity and that the use, source and destination of funds do not involve money laundering.

Anti-Terrorist Financing (ATF) is another filtering component, classified under the Suspicious Activity Reports (SAR).

The Office of Foreign Assets Control (OFAC) is a US agency that enforces economic and trade sanctions against targeted foreign states, organizations, and individuals.

Add-on component extract messages from the Alliance Access flows, and compare message data against sanction lists. Suspected messages can be disposed to a security officer through a GUI, or routed to a "suspicious queue" for further validation.

3.4 Operational Activities

The operational add-ons will do not modify the message, but provide sanity and global features, such as archiving, auditing, back-up and reporting features.

4 Alliance Developers Toolkits

Two developers toolkits are considered for the development of Alliance Add-ons:

- The Alliance Developers Toolkit
- The Alliance Integrator Developer Kit

4.1 Alliance Developers Toolkit

The Alliance Developers Toolkit is a collection of software and documentation that constitutes the open interface to Alliance Access. The Alliance Developers Toolkit enables third party and financial institution developers to build their own applications for interfacing with Alliance Access.

The Alliance Developers Toolkit relies on some key design aspects of the Alliance Access architecture. Alliance Access is made out of functional components. The Alliance Access database, installation process and process control are organized along these components. With the Alliance Developers Toolkit, third party applications can smoothly integrate into Alliance Access as a new component.

The Alliance Developers Toolkit provides libraries containing APIs that can call a subset of the services provided by the Alliance Access servers. APIs are provided in C development language. No C++ or Java versions exist at this point.

The Alliance Developers Toolkit APIs offer the required robustness and security level required by Alliance kernel to protect the main functions against uncontrolled Alliance Developers Toolkit applications.

4.1.1 Alliance Developers Toolkit Usage

The Alliance Developers Toolkit provides developers with APIs that enable intercepting messages from Alliance Access workflow, to inspect them, and re-route them after optional modification. Messages can be captured in both direction (messages coming from back-office application and route to SWIFT, or messages coming from SWIFT, before they reach any business applications).

Messages are locked by the Alliance Developers Toolkit component. No message can be created nor deleted by an Alliance Developers Toolkit component.

The fact that messages can be intercepted just before they get out to SWIFT (even if they are created locally using Alliance Access GUI), or before they can be processed by any business applications, makes the Alliance Developers Toolkit an interesting developer tool for Anti Money Laundering, anti-terrorist, OFAC, watch and caution lists, and other Politically Exposed Persons (PEP) filtering systems.

The integration with back-office applications can also be based on the Alliance Developers Toolkit, or to monitor the messaging flows for reporting or audit purpose, using Business Application Management technology. Some use it to convert or to enrich messages from STP, to manage duplicates, to derive statistics, to extract messages that need extra approval before being sent out, and for many other reasons.

Note that, if most of the message access and modifications function of Alliance Access are available for the Alliance Developers Toolkit developers, some functions are disabled for security reasons. With the Alliance Developers Toolkit, it is not possible to do the following tasks:

- modify messages that are not in the routing point investigated by the add-on
- delete messages present in Alliance Access (it is however possible to complete messages)
- update the Alliance Access correspondent information file
- read the Alliance Access Event Journal
- modify the System Management configuration parameters
- authenticate or test a message

4.1.2 Alliance Developers Toolkit Licence

The Alliance Developers Toolkit is distributed together with the Alliance Access software. The following two licensed packages exist for the Alliance Developers Toolkit:

- The **toolkit run-time** license is to be installed in each operational site where an Alliance Access Add-on is to run. Add-on customers need to purchase the Alliance Developers Toolkit run-time license.
- The **toolkit development** license is required to develop Alliance Developers Toolkit applications.

The runtime package consists of documentation, shared libraries that implement the Alliance Developers Toolkit APIs, and a specific Alliance Developers Toolkit installation procedure that must be used to integrate Alliance Developers Toolkit components into Alliance environments.

4.2 Alliance Integrator Developer Kit

Alliance Integrator defines a framework for the development of Web services using the Java CAPS Repository.

The Alliance Integrator Developer Toolkit provides the necessary API, samples, documentation and support to develop, test and deploy a web services over Alliance Integrator.

5 SWIFTReady Alliance Add-on

5.1 Scope

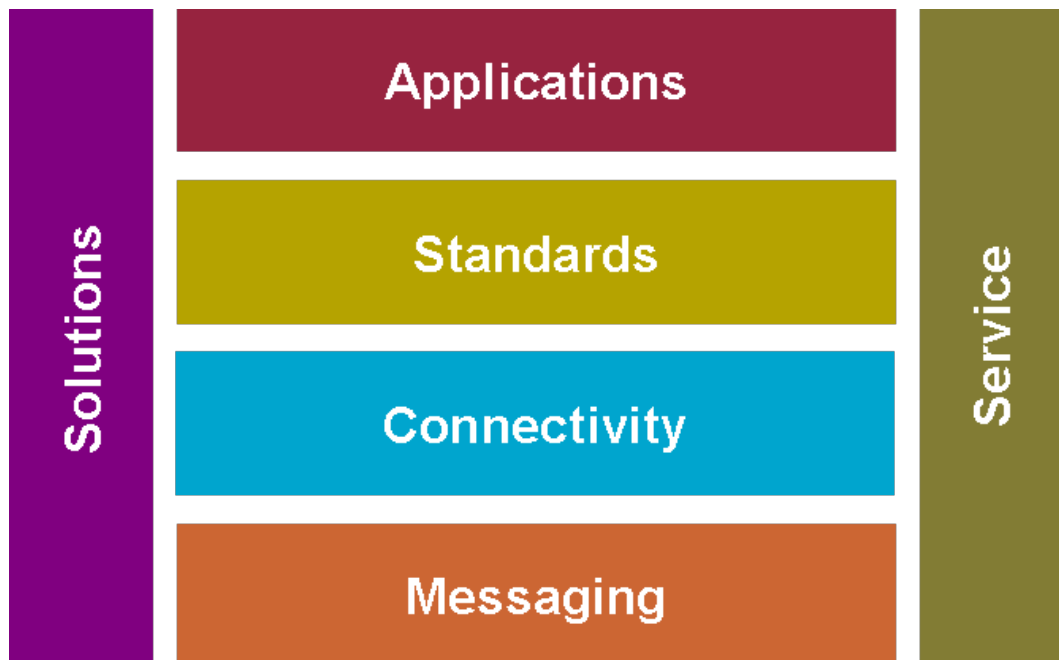
The SWIFTReady label programme was created in 1998 to ensure the proper support of SWIFT Standards, messaging services and interface connectivity by back-office applications and middleware.

An **Alliance Add-on** refers to the combination of processes, software components built around Alliance interfaces, and which exhibit business or technical features that SWIFT Users are looking for, and cannot find today as part of the Alliance interfaces.

The SWIFTReady Alliance Add-on label is conferred to software products that have been developed with the Alliance Access or Alliance Integrator Developer Kit, and are compliant with the development guidance and good coding practices as laid down in this document.

The Alliance Add-on focuses on the specific needs of the financial community, including actors such as banks, brokers, fund managers, traders, banking and securities market infrastructures and corporate treasury departments.

The Alliance Add-on integrates a business logic with messaging services and acts upon the flow of business messages which transit through Alliance interfaces.



5.2 Changes compared to 2011

New label

Vendors applying for the SWIFTReady Alliance Add-on label for the first time must comply with all criteria defined in this document.

Existing label (renewal from previous year)

Vendors that have been granted the SWIFTReady Alliance Add-on label in 2011 are required to prove compliance to Standards Release (SR) 2012.

5.3 Add-on Typology

The Alliance Add-on label intends to capture the features requested by SWIFT users. The most common usage of add-ons can be categorised as follows:

1. Sanction list Filtering (AML, OFAC, ATF, Embargo, PEP)
2. Business Activity Monitoring (BAM)
3. Technical flow monitoring
4. Duplicate checking
5. Straight Through Processing (STP) enhancer
6. Archiving
7. Reporting
8. Back-office connectivity
9. Network connectivity

6 Standards

SWIFT develops business standards to support transactions of the financial market.

FIN, based on proprietary message standards, is the original messaging service. Later, support for standards on FIN was extended to include ISO 15022 standards and the service name was changed to FIN. Message types that operate on FIN are referred to as **MT**.

In 1997 SWIFT launched an IP-based secure network. This network enables the exchange of message types represented in XML format. It provides support for ISO 20022 and for FpML message standards.

For more information, please refer respectively to www.iso20022.org and www.isda.org. Messages Types of standard ISO 20022 are referred to as **MX**.

Today the network traffic of MT still represents the majority of SWIFT traffic. These are being progressively complemented and/or replaced by XML-based messages, which ease validation, and enable the transfer of richer data for complex transactions.

Label requirement	Reference number 1	Mandatory
The Alliance Add-on must support the MT and MX related to the solution at stake, as listed in SWIFT User Handbook (UHB Online). Message support implies the capacity to capture messages for many category and type, treat them as appropriate, and route them within Access, according to the Add-on treatment result.		

6.1 MT

FIN messages convey business payloads named Message Type (MT). Each MT is defined by a three-digit number and holds a specific business meaning. MTs are categorised as follows:

Category	Category name	Number of messages (estimate)
0	System Messages	56
1	Customer Payments and Cheques	18
2	Financial Institution Transfers	19
3	Treasury - Foreign Exchange, Money Markets & Derivatives	26
4	Collections & Cash Letters	18
5	Securities	67
6	Treasury – Commodities & Syndications	15
7	Documentary Credits & Guarantees	29
8	Travelers Cheques	11
9	Cash Management & Customer Status	21

MT messages are gathered into Business Transactions (that is, Trading, Settlement, Funds). For example, a Trading Transaction includes an Order to Buy (MT 502), a Trade Confirmation (MT 515, MT 518), or a Statement of Open Orders (MT 576).

More information over supported Business Transactions can be found on <http://www.swift.com/solutions/standards/business>.

The Alliance Add-on must support the MT messages required for the Add-on Application domain. It means that they must be able to acquire messages, access the meaningful data for checking and releasing the messages as necessary.

Label requirement	Reference number 2	Mandatory
The Alliance Add-on must support all the MT messages pertaining to the solution at stake		

6.2 MX

Leveraging the emergence of XML as de-facto standards for inter-systems communication, SWIFT uses the ISO 20022 methodology to design standards, based on business processing modelling. The ISO 20022 methodology uses a central data dictionary containing re-usable business elements to build XML standards, named MX, that are used in business transactions and provide interoperability across financial services.

At the end, the whole financial community must be moving to MX, but adoption will vary by business area, depending on the drivers. SWIFT intends to provide translation rules to support interoperability in business areas where coexistence of MT and MX is necessary.

An MX message contains the business area specific payload. It has the structure defined by the corresponding XML Schema Definition (XSD), as published in the UHB. These schemas provide not only message structure but also instructions on message scope and usage, rules and guidelines. Sample data is also provided in the UHB.

Label requirement	Reference number 3	Optional
The Alliance Add-on Application must support the MX required for the application domain, as listed in SWIFT User Handbook (UHB Online) for the business application.		

6.3 ISO 20022

ISO 20022 provides the financial industry with a common platform for the development of messages in a standardised syntax (mainly XML), using:

- a modelling methodology, based on Unified Modelling Language (UML) to capture financial business models, business transactions and associated message flows into a syntax-independent data dictionary
- a set of XML design rules to convert the messages described in UML into XML schemas

SWIFT applies the ISO 20022 definitions to structure SWIFT Standards XML (MX) message types. Any MX artefact (XML element, simple or complex type, attribute) has a corresponding business artefact definition (message component, element, data type). MX and business artefacts are both represented in UML.

The ISO 20022 Financial Dictionary contains reusable items, which can be categorised into business concepts (association, component, rule, actor and role), data types and message concepts (message element and rule). Non-reusable artefacts (business processes, information flows, MX messages, and technical message elements) are found in the *Standards Message Reference Guides*.

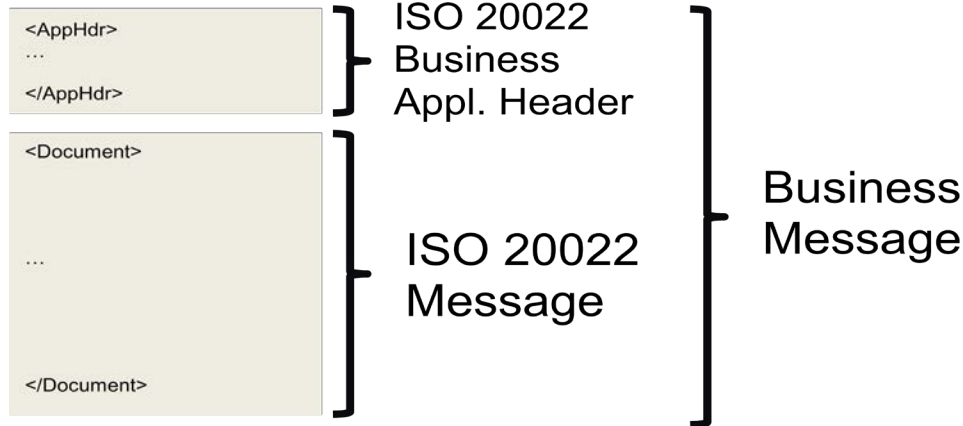
The ISO20022 data dictionary is available, free of charge, in electronic downloadable format on www.iso20022.org.

The complete catalogue of ISO 20022 messages, including the ISO 20022 data dictionary, the Message Definition Reports (MDR) and XML schemas (XSD) for current and future versions, is available on the ISO 20022 website www.iso20022.org.

Label requirement	Reference number 4	Optional
The Alliance Add-on must support the ISO 20022 data dictionary, that is, all reusable business elements and components, as a basis for parsing, validating and storage of MX messages and associated rules.		

6.3.1 Business Application Header

In 2010, ISO introduced a Business Application Header (BAH) to harmonise the access to operational data and make it easier to implement ISO 20022 messages. The idea behind the Business Application Header is to extract routing and message features information (such as sender, receiver, possible duplicate, signature, priority) from the business payload and to expose it in a header that can be accessed by business applications and middleware in the same way for all MX messages. The Business Application Header is network-independent and enables intermediate applications to access and update the routing information without having to touch to the business payload.



The Business Application Header is available as an additional XML schema, which comes on top of the business payload. It replaces the application header that is found today in some of the MX messages. New ISO 20022 messages and new version of existing MX messages will progressively require the Business Application Header.

The Business Application Header is published in the [ISO 20022 Catalogue of Messages](#).

Label requirement	Reference number 5	Optional
The Partner Application must support the ISO 20022 Business Application Header for routing and operational data processing.		

6.3.2 Extensions and Restrictions

ISO 20022 also introduces extensions and restrictions to meet the needs of individual customers and communities.

Extensions provide supplementary data to existing ISO 20022 messages, available as ISO 20022 compliant definitions and processed the same way as the rest of the messages. Extensions must be developed and used by exception when the data at stake is too specific or too volatile to include it in the ISO standard, for urgent business need only or to accommodate market specific needs. ISO 20022 extensions are submitted to ISO for approval prior to publishing.

Restrictions will be made available as ISO 20022 Variants. Variants are usually defined and published by communities to bring clarity and raise awareness on specificities. Variants need to be compliant with the published ISO 20022 version of the message.

Label requirement	Reference number 6	Optional
The Partner Application must support the ISO 20022 Extensions and Restrictions		

7 SWIFT Messaging Protocols

SWIFT provides several messaging services to support the needs of the financial community. FIN and InterAct cater for the automation of structured or important financial information transfers to business partners, while FileAct conveys large payload of free format and bulked/low value information.

7.1 FIN

FIN is a secure, reliable and resilient, access-controlled, structured, store-and-forward messaging service. Value-added processing includes message validation against SWIFT Standards, delivery monitoring and prioritisation, on top of central message storage and retrieval.

FIN supports more than 240 message types (MT) categorised into nine market segments according to their business usage (Payment, Treasury, FXMM, Derivatives, Collections, Securities, Trade, Commodities and Cash Management).

FIN messages are made of data blocks for addressing and control (block 1 to 3), MT business payload (block 4) and system trailers block (block 5).

{1: BASIC HEADER BLOCK}
{2: APPLICATION HEADER BLOCK}
{3: USER HEADER BLOCK}
{4: TEXT BLOCK}
{5: TRAILER BLOCK}

Label requirement	Reference number 7	Optional
The Alliance Add-on must support the FIN messaging. In particular, it must be able to extract the block 4 and to parse it according to the add-on business requirements.		

7.2 InterAct

InterAct caters for the interactive real-time and store-and-forward exchange of messages between applications. InterAct is widely used in many of SWIFT solutions, including Funds, Exceptions and Investigations and Collateral Management. It is also used for Market Initiatives such as TARGET2, SEPA, or TARTET2-Securities.

InterAct supports XML format and other structured formats (FIX, FpML) wrapped into an XML envelope. InterAct Central Services provide enhanced validation services over MX messages, which are XML messages designed with the ISO20022 methodology.

The Alliance Add-on must support InterAct messaging service to cover the needs of all of SWIFT solutions and Market Initiatives. The *SWIFTNet Messaging Operations Guide* (see [UHB Online](#)) must be strictly adhered to.

Label requirement	Reference number 8	Optional
The Alliance Add-on must support the InterAct messaging services and adhere to the latest release of the <i>SWIFTNet Messaging Operations Guide</i> in the UHB Online .		

7.3 FileAct

FileAct enables secure and reliable transfer of files and is typically used to exchange batches of structured financial messages and large reports.

FileAct supports tailored solutions for market infrastructure communities, closed user groups and financial institutions.

It is particularly suitable for bulk payments, securities value-added information and reporting, and for other purposes, such as central-bank reporting and intra-institution reporting

The Alliance Add-on must support FileAct messaging service The *SWIFTNet Messaging Operations Guide* (see [UHB Online](#)) must be strictly adhered to.

Label requirement	Reference number 8	Optional
The Alliance Add-on must support the FileAct messaging services and adhere to the latest release of the <i>SWIFTNet Messaging Operations Guide</i> in the UHB Online .		

8 Alliance Access Integration

The Alliance Add-on is tightly integrated with Alliance Access using the APIs that are made available to the developer of the Alliance Developers Toolkit.

The Alliance Developers Toolkit is a collection of software and documentation that enables tight integration with the Alliance Access workflow engine (routing points, journalizing, and interventions history).

The Alliance Developers Toolkit provides API libraries that call a subset of the services provided by the Alliance servers both for MT and MX messages.

Label requirement	Reference number 9	Mandatory
<p>The architectural design shall be provided to SWIFT for review and advice.</p> <p>The requirements must be assessed during the design and the development phases. SWIFT will verify some of these requirements in the product source code and documentation during the formal presentation of vendor's product at SWIFT HQ premises for the Add-on validation.</p> <p>The product must be compliant with the last versions of Alliance Access software, and shall follow the release cycle of Alliance Access. In particular, the product shall be at least re-linked, re-packaged and re-tested within 3 months following the delivery of any new Alliance Access release (mandatory or optional releases).</p>		

Alliance Access 7.0 is a major release, which extend the scope of these adapters to support any XML messages using UTF-8 encoding.

The Alliance Add-on must support Alliance Access 7.0 or Alliance Integrator 7.0 (chapter 9 Alliance Integrator Integration).

Label requirement	Reference number 10	Mandatory
The Alliance Add-on must support Alliance Access 7.0.		

8.1 Alliance Developers Toolkit Developer Guide Compliance

An Alliance Add-on product must support all the mandatory features referred to in the Alliance Developers Toolkit documentation and may support any of the additional Optional features.

The following is a non-exhaustive list of Alliance Developers Toolkit Developer and Reference guides compliance statements:

1	The design and software code shall be compliant with the Alliance and Alliance Developers Toolkit conventions on Alliance Developers Toolkit coding.
2	The Add-on component names shall be four capital characters long and must end either with "S" (server) or "A" (application with GUI). Several components can co-exist within the same product.
3	The product shall be designed as to make proper usage of all necessary Alliance Developers Toolkit APIs (Access control, event journal, message files, routing, security definition, process control).
4	A registration programme shall be provided to register Add-on component to Alliance Access as an integrated part of the installation process.
5	Component directory structure shall be defined in accordance with developer guide, to enable the application to run in an operational Alliance site (for example, bin, \$ARCH), and to build correct Alliance Developers Toolkit installation media.
6	It is recommended to store configuration files under the <i>ADK_DIR</i> sub-directory of the Alliance Access database directory, and Data files under a subdirectory named after the Add-on component name. This sub-directory must not contain any trace files.

7	Programme and executable names shall be prefixed by the Add-on component name.
8	The Add-on component data files shall be different from the ones used by Alliance and Alliance Developers Toolkit.
9	Reserved symbol names used by Alliance shall not be used, as this can lead to unexpected behaviour of the faulty programme even if its compilation succeeded. A list of all the reserved symbol names is provided in Developer Guide Appendix A.
10	All entity variables used in programmes shall be initialised before being used.
11	Automatic re-launch after crash shall be configured via the fields exsa_num_retries and exsa_retry_period
12	All Add-on processes shall register themselves with Alliance Access using the adequate API <i>PcsRegister</i> . A registration script must be provided for this purpose.
13	Significant events related to the start/stop of the Add-on and any message processing shall be journalised in standardised format at a centralised location known as the Event Journal.
14	The journalisation of an event may involve informing users through an alarm in the form of a window that pop-up on specified operator screens to display alarm information. Journal event have a severity. Information and warnings events must not be defined as alarms. Severe and fatal errors must be defined as alarms.
15	Journal events have a class. At least the following must be used: <ul style="list-style-type: none"> · Process: for process start/stop and general information · Message: for message processing · Communication: if the component provide a link to an external system · Data: for internal data handling, like configuration files
16	At installation time each application component with a GUI shall register an <i>appl</i> record that will define its icon (as described in Developer Guide's Process Control section).
17	If a GUI needs to exhibit usage restrictions, it shall register a <i>func</i> record for every restriction function. If the functions need parameter values then a <i>perm</i> record must be registered for every permission.
18	All Alliance Developers Toolkit APIs return codes shall be trapped and appropriate message shall be logged in the event journal or in the trace file.
19	Interventions shall be written in the message history each time a significant event happens to the message (message checked, modified, approved...)
20	A message appendix shall be created when a message enters or leave Alliance Access through the component (that is, link to an external system).
21	In case a message is modified (Message enrich/repair Add-on), the original text shall be kept in a free text intervention and clear indication on what has been changed/added/removed.
22	The Add-on shall follow the Alliance Access release cycle. Partners shall at least re-link, re-package and re-test the Add-on product within the three months of reception of a new Alliance Access release. From time to time, code updates shall be performed as indicated in Alliance release letter. This holds for major and minor Alliance Access releases.

Label requirement	Reference number 11	Mandatory
The Alliance Developers Toolkit component must be compliant with Alliance Developers toolkit Developer and Reference Guides.		

8.2 Alliance Access Message Recovery

1	Processes using API to update the database shall have a recovery process available to deal with API failure (return code ADK_FAILURE).
---	--

2	Message processing functions shall be designed with the assumption that it is potentially being started after the crash of an earlier message processing function process. Message processing function must first do its recovery and afterwards do its normal processing. Alternatively, an Alliance Developers Toolkit component can decide to have an Alliance Developers Toolkit exec to perform the recovery.
3	The recovery process shall check the status of reserved messages, and perform an undo or completion of the message depending on the message status.
4	If at any stage of processing, a message processing API returns ADK_FAILURE, then the message processing function shall exit with the failure code ADK_FAILURE_EXIT. The Alliance Access control software must then take care of restarting the message processing function, whenever this is possible.
5	If a message processing function makes more than one update to a message instance then after each update the message shall be in a recognisably different state (state held in Alliance Access database that the recovery process is able to detect) using for instance an appendix or an intervention, so that the recovery process can know how much of the normal processing has been performed and possibly activate a roll-back or a process completion.
6	The recovery process shall either undo (roll-back) what was done by the normal processing or can finish what was left unfinished. In particular, every message shall be unreserved at the end of the recovery process.

Label requirement	Reference number 12	Mandatory
The Alliance Developers Toolkit component must develop recovery procedure to ensure that no message got lost or result in an unknown status as a follow up of Access failure.		

8.3 Alliance Access Add-on Installation and Configuration

The different Add-on components to be installed and registered with the Alliance Developers Toolkit Add-on must be clearly described and provided to SWIFT, in order to facilitate customer support.

1	Alliance Developers Toolkit server (Component name (4 capital letters). Server names must ends with an "S").
2	Alliance Developers Toolkit client (Component name (4 capital letters). Client names must end with an "A").
3	Message processing function - Each message processing function must have a process (exsa), a processing (mpfn), routing points (rpoi) and an application (appli) records.
4	Event Messages Journal entries must all start with the component names. This enables easy tracking of Add-on behaviour in isolation.
5	Routing points - Routing points are logical locations where message instances are located and processed.
6	Registration programme - Registration programme must be written in Tool Command Language (TCL).
7	Configuration Parameters – each cnfg parameter name must be provided.
8	Functions - functions (func) names.
9	Application - each appl parameter name must be provided
10	Permissions - permissions (perm) names.

Label requirement	Reference number 13	Mandatory
The Alliance Developers Toolkit components must be able clearly described and provided to SWIFT, in order to facilitate customer support.		

8.4 Alliance Access User Documentation

A user and installation manual shall be provided together with the Add-on product. The following information shall be made available in the product documentation. The aim is to ensure smooth installation, configuration and daily usage at customer premises.

1	Installation of server and application components on each operating system
2	Component registration using cipher password
3	List of installed entities (components, routing points, routing rules, configuration parameters)
4	Typical product configuration (routing rules and schema), when Add-on can be configured.
5	Create, Approve and Activate new schema using Routing Application in Alliance Workstation (when appropriate)
6	Refine product specific routing rules using Alliance Access Routing application. Customize routing points thresholds and routing rules (sequence numbers, routing conditions and results)
7	Modify default Add-on parameters using system Management (sleep time, in and out directories)
8	List journal entries (number, classification, text and explanation)

Label requirement	Reference number 14	Mandatory
A complete user and installation manual must be provided to SWIFT together with every Add-on product		

9 Alliance Integrator Integration

Alliance Integrator defines a framework for the development of Web services inside the Java CAPS Repository.

The development of Web services over Integrator must be compliant with at least one of:

- GlassFish V2,
- JBoss-5.1.0.GA
- Tomcat 6.0 platforms

The Development of a web services over Integrator 7.0 requires the deployment of Alliance Integrator Developer kit, dedicated training and running of a proof of concept with the SWIFT Integration team, Marketing, and Sales

Collaboration in terms of administrative and marketing information is requested. In particular the Application Provider must provide SWIFT under non-disclosure agreement with customer related information.

The application provider must provide SWIFT with a list of at least three live customers that actively use the Alliance Add-on to connect to SWIFT.

By "customer" we mean separate financial institutions using the product to generate and receive messages and files transported by SWIFT. Five live sites from the same customer are not enough to qualify for a label.

SWIFT reserves the right to contact the relevant customer to validate the functionality of the application submitted for SWIFTReady certification. A questionnaire will be sent as the basis for the customer validation which can be in the form of a telephone interview, an e-mail or a discussion at the customer site.

The information provided by the customer will be treated as confidential and will not be disclosed, unless explicitly expressed by the customer.

Label requirement	Reference number 15	Mandatory
<p>The application provider must supply the following information under non-disclosure agreement:</p> <ul style="list-style-type: none"> · A list of at least one live customer that actively uses the Alliance Add-on in a SWIFT context. · A list of all customers active in the finance sector. The list must provide institution names, locations, and an overview of the integration scope (domain, features, and sites) for the present and previous year. · A product roadmap for 2012 and 2013 containing the plans for further Alliance Add-on development, support of SWIFT solutions and new releases. · A complete set of Alliance Add-on documentation, including features overview, SWIFT adapters, workflow engine capability and user manuals. · A dedicated web page on Vendor web site describing the SWIFTReady application used in a SWIFT context. The page must be maintained with the same URL for a complete year and will be referenced to on www.swift.com. 		

Legal Notices

Copyright

SWIFT © 2012. All rights reserved.

You may copy this publication within your organisation. Any such copy must include these legal notices.

Confidentiality

This publication contains SWIFT or third-party confidential information. Do not disclose this publication outside your organisation without the prior written consent of SWIFT.

Disclaimer

The information in this publication may change from time to time. You must always refer to the latest available version on www.swift.com.

Translations

The English version of SWIFT documentation is the only official and binding version.

Trademarks

SWIFT is the trade name of S.W.I.F.T. SCRL. The following are registered trademarks of SWIFT: SWIFT, the SWIFT logo, the Standards Forum logo, 3SKey, Innotribe, Sibos, SWIFTNet, SWIFTReady, and Accord. Other product, service, or company names in this publication are trade names, trademarks, or registered trademarks of their respective owners.