SWIFT Certified Application

Alliance Monitoring Add-On

Label Criteria 2018

This document provides a structured and detailed view of the criteria that an add-on application must fulfil to obtain the SWIFT Certified Application - Alliance Add-on 2018 label.

26 January 2018

Table of Contents

Preface

Purpose of the document

This document provides a structured and detailed view of the criteria that an add-on application must fulfil to obtain the SWIFT Certified Application - Alliance Add-on 2018 label.

Audience

This document is for the following audience:

- Application vendors considering the certification of a product
- SWIFT customers seeking to understand the SWIFT Certified Application Programme or involved in selecting third-party applications

The audience must be familiar with both the technical and business aspects of SWIFT.

Related documentation

- SWIFT Certified Application Programme Overview

  The document provides an overview of the SWIFT Certified Application Programme. It describes the benefits of the programme for SWIFT registered providers that have a software application they want to certify for compatibility with SWIFT Standards, messaging services, and connectivity. This document also describes the application and validation processes that SWIFT uses to check such SWIFT compatibility. SWIFT's certification of an application is not an endorsement, warranty, or guarantee of any application, nor does it guarantee or assure any particular service level or outcome with regard to any certified application.
- Documentation (User Handbook) on www.swift.com

1 SWIFT Certified Application Programme

Overview

The SWIFT Certified Application label programme extends throughout the financial application chain to include Trade, Treasury, Payment, Corporate, and Securities applications.

Each SWIFT Certified Application 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 do the following:

- provide automation of message processing in a SWIFT context
- support straight-through processing to increase customer value
- limit customisation needs and costs
- reduce time to market

SWIFT Certified Application - Alliance Add-on

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

Alliance Add-ons are software components that extend the value proposal of Alliance interfaces by providing the business features required by the financial market.

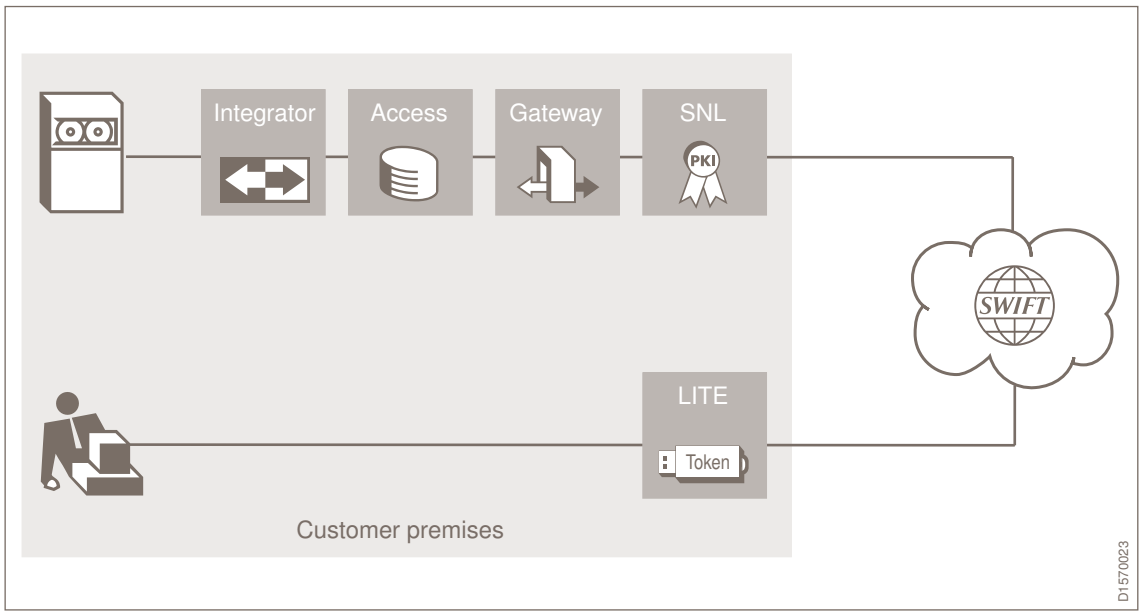
These added-value features include the following:

- Anti-Money-Laundering (AML)
- Financial fraud detection
- Blacklist filtering
- Duplicate checking
- Straight-through processing (STP) enrichment
- Business Activity Monitoring (BAM)
- Exceptions and Investigations
- Messaging data services including translation and validation
- Other operational activities

2 Alliance Add-on

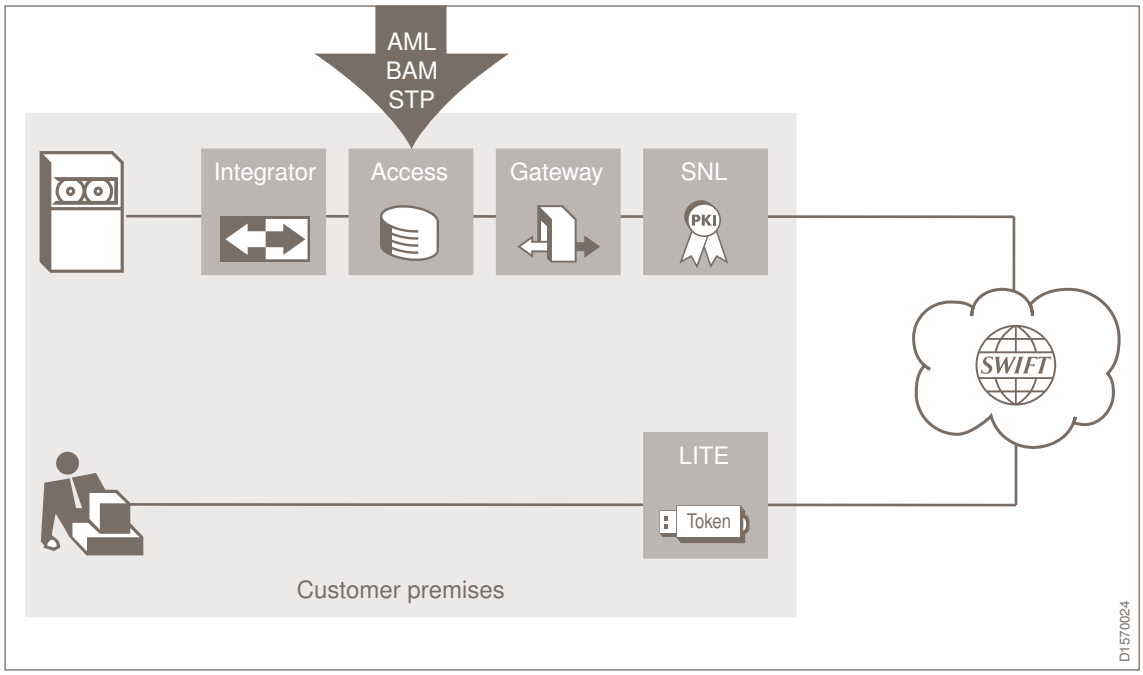
Deployment over back-office applications

A **Back-office application** connects to Alliance interfaces through the various Alliance adapters (for example, MQ, File or Simple Object Access Protocol (SOAP)). The application typically generates the messages that are routed through the Alliance interfaces, where business processes can enrich, change, validate, and route them. In this model, applications are the source or link of the messages.



Deployment over business add-ons

A **business add-on over Alliance** interfaces performs various tasks on the messages when the message transactions are completed. 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. The business process engine of the Alliance interfaces activates the add-on.



Legend

Term	Explanation
AML	Anti-Money-Laundering
BAM	Business Activity Monitoring
STP	Straight-through processing

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 typically modify the following:

- **message contents**

  Translate, transform, enrich, and add security information
- **message routing scheme**

  Validate, approve, reject, send to manual check, copy, archive, and monitor

The **SWIFT Certified Application - Alliance Add-on** focuses on business add-ons that can be deployed over Alliance Access.

Alliance Access is the general-)purpose messaging interface used by SWIFT customers to exchange messages and files. Vendors develop add-on components using the *Alliance Developers Toolkit* to enrich the features of Alliance Access.

For more information, see Alliance Developers Toolkits on page 10.

3 Alliance Add-on Typology

3.1 Add-On Usage

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:

- Sanction list Filtering (AML, OFAC, ATF, Embargo, PEP)
- Business Activity Monitoring (BAM)
- Technical flow monitoring
- Duplicate checking
- Straight through processing (STP) enhancer
- Archiving
- Reporting
- Back-office connectivity
- Network connectivity

3.2 Message Repair

The Alliance Add-on enables messages to be edited and repaired in exceptional cases. This applies to messages that do not pass validation and that require urgent repair before sending them to SWIFT.

Message manipulation can be restricted to the users with a 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).

3.3 Business Activity Monitoring

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. This guarantees complete control of all data flows by tracking and displaying all of the message events.

A business transaction consists of messages flowing from the 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 with some events (for example, message DeINnote received after three days, reports must be generated).

3.4 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 country in question. 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, organisations, and individuals.

Add-on components extract messages from the Alliance Access flows, and compare message data against sanctions lists. Suspect messages can be assigned to a security officer though a GUI, or routed to a "suspicious" queue for further validation.

3.5 Operational Activities

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

4 Alliance Developers Toolkits

4.1 About the 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 organised using 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 robustness and security level required by Alliance kernel to protect the main functions against uncontrolled Alliance Developers Toolkit applications. By using APIs, development and implementation are controlled and more standardised.

4.2 Alliance Developers Toolkit Usage

The Alliance Developers Toolkit provides developers with APIs to intercept messages from the Alliance Access workflow, to inspect them, and to re-route them after optional modification.

Messages can be captured in both directions:

- messages coming from the back-office application and routed to SWIFT
- messages coming from SWIFT, before they reach any business applications

The Alliance Developers Toolkit component locks messages. An Alliance Developers Toolkit component cannot create or delete messages.

Messages can be intercepted just before they reach SWIFT (even if they are created locally using Alliance Access GUI), or before they are processed by any business applications. This 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 purposes, 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.

Most of the message access and modification functions of Alliance Access are available to the Alliance Developers Toolkit developers. However, 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 (possibility 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

4.3 Alliance Developers Toolkit Licence

The Alliance Developers Toolkit is distributed together with the Alliance Access software.

The following licensed packages exist for the Alliance Developers Toolkit:

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

The run-time 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.

5 SWIFT Certified Application - Alliance Add-on Criteria 2018

5.1 Scope

The SWIFT Certified Application 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 and software components which are built around Alliance interfaces. An Alliance Add-on exhibits business or technical features that SWIFT Users are looking for and cannot find today as part of the Alliance interfaces.

SWIFT grants the SWIFT Certified Application - Alliance Add-on label to software products that have been developed with Alliance Access, and which are compliant with the development guidance and good coding practices described in this document.

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

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

5.2 Certification Requirements

New label

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

Label renewal

For all Alliance Add-ons, a *Security Conformance Requirements Statement* should be completed.

Vendors that have been granted the SWIFT Certified Application Alliance Add-on label in 2017 are required to prove compliance with Standards Release (SR) 2018.

6 Standards

6.1 Standards Overview

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. 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. ISO 20022 messages are referred to as **MX**.

You can find more information about standards on www.iso20022.org and www.isda.org.

Today the network traffic of MT messages still represents the majority of SWIFT traffic. These messages are being progressively complemented or replaced by XML-based messages. XML messages 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 in question, as listed in the User Handbook. Message support implies the capacity to capture messages for many categories and types, treat them as appropriate, and route them within Access, based on how the Add-on has been developed.		

6.2 MT Messages

FIN messages contain business payloads named message type (**MT**). Each MT is defined by a three-digit number and has 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, and Derivatives	26
4	Collections and Cash Letters	18
5	Securities Markets	67

Category	Category name	Number of messages (estimate)
6	Treasury - Commodities and Syndications Reference Data	15
7	Documentary Credits and Guarantees	29
8	Travellers Cheques	11
9	Cash Management and Customer Status	21

MT messages are grouped by 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).

You can find more information about supported Business Transactions at www.swift.com.

The Alliance Add-on must support the MT messages required for the Add-on Application domain. This means that it must be able to acquire messages and access the meaningful data to check and release 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.		

6.3 MX Messages

With 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 (MX messages). These standards are used in business transactions and provide interoperability across financial services.

The financial community must move eventually to MX, but adoption will vary by business area, depending on the drivers. SWIFT provides 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 on the User Handbook. These schemas provide not only message structure but also instructions on message scope and usage, rules, and guidelines. Sample data is also provided on the User Handbook.

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

6.4 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.

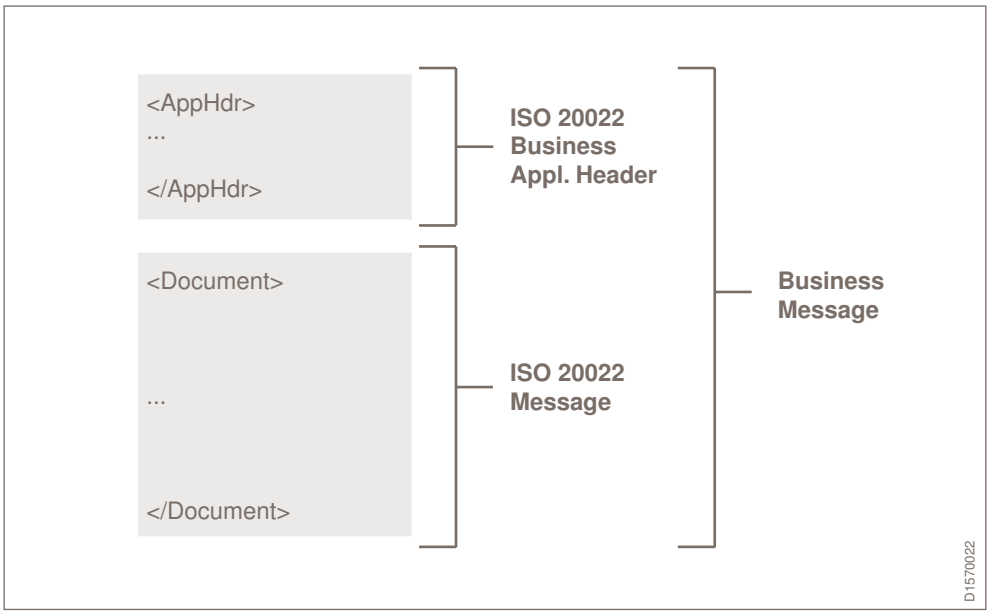
The ISO 20022 data dictionary is available, free of charge, in electronic downloadable format on www.iso20022.org.

You can find 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, 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.		

6.4.1 Business Application Header

In 2010, ISO introduced a Business Application Header to harmonise the access to operational data and make it easier to implement ISO 20022 messages. The Business Application Header extracts routing and message feature information (such as sender, receiver, possible duplicate, signature, priority) from the business payload. It then exposes 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 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 versions 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 Vendor Application must support the ISO 20022 Business Application Header for routing and operational data processing.		

6.4.2 Extensions and Restrictions

ISO 20022 also allows 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 on an exceptional basis. This includes when the data is too specific or too volatile to include it in the ISO standard, for urgent business needs only, or to meet market-specific needs. ISO 20022 extensions are submitted to ISO for approval before publication.

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

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

7 SWIFT Messaging Protocols

7.1 Overview of 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.
- **FileAct** conveys large payloads of free-format and bulked low-value information.

7.2 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 (see MT Messages on page 13).

FIN messages are made of data blocks for addressing and control (blocks 1 to 3), the MT business payload (block 4), and the system trailer (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 FIN messaging. In particular, it must be able to extract block 4 and to parse it according to the add-on business requirements.		

7.3 InterAct

InterAct caters for the interactive real-time and store-and-forward exchange of messages between applications. InterAct is widely used in many SWIFT solutions, including Funds, Exceptions and Investigations, and Collateral Management. It is also used for market initiatives such as TARGET2, SEPA, or TARGET2-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 ISO 20022 methodology.

The Alliance Add-on must support the InterAct messaging service to cover the needs of all SWIFT solutions and market initiatives. The Alliance Add-on must also strictly adhere to the SWIFTNet Messaging Operations Guide (available on the User Handbook).

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

7.4 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 the FileAct messaging service. The Alliance Add-on must also strictly adhere to the SWIFTNet Messaging Operations Guide (available on the User Handbook).

Label requirement	Reference number 9	Optional
The Alliance Add-on must support the FileAct messaging service and adhere to the latest release of the SWIFTNet Messaging Operations Guide.		

8 Alliance Access Integration

8.1 Overview of 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, journalising, and intervention history).

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

Label requirement	Reference number 10	Mandatory
The architectural design must be provided to SWIFT for review and advice. The requirements must be assessed during the design phase and the development phase. SWIFT will verify (at SWIFT headquarters) some of these requirements in the product source code and the documentation during the formal presentation of the vendor's product for the Add-on validation. The product must be compliant with the latest versions of Alliance Access software, and must follow the release cycle of Alliance Access. In particular, the product must 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).		

SWIFTNet Release 7.2

Release 7.2 support is a mandatory requirement in 2018.

More details on the SWIFTNet Release 7.2 can be found on www.swift.com:

- Release 7.2
- User Handbook

Label requirement	Reference number 11	Mandatory
The Alliance Add-on must support Alliance Access 7.2.		

Label requirement	Reference number 12	Mandatory
The Alliance Add-on should integrate with the updated 64-bit libraries, and be recompiled with 64-bit.		

8.2 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 compliance statements in the Alliance Developers Toolkit Developer and Reference Guides:

1	The design and software code must be compliant with the Alliance and Alliance Developers Toolkit conventions on Alliance Developers Toolkit coding.
2	The Add-on component names must 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 must be designed 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 must be provided as an integrated part of the installation process to register the Add-on component to Alliance Access.
5	The component directory structure must be defined according to the Developer Guide, to enable the application to run on an operational Alliance site (for example, bin, $ARCH), and to build correct Alliance Developers Toolkit installation media.
6	SWIFT recommends storing configuration files under the **ADK_DIR** 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 must start with the Add-on component name.
8	The Add-on component data files must be different from the ones used by Alliance and Alliance Developers Toolkit.
9	Reserved symbol names used by Alliance must not be used, as this can lead to unexpected behaviour of the faulty programme even if its compilation succeeds. A list of all the reserved symbol names is provided in Appendix A of the *Developing SWIFT Alliance Access Plug-ins with ADK*. For more information, go to www.swift.com.
10	All entity variables used in programmes must be initialised before being used.
11	Automatic re-launch after crash must be configured through the fields **exsa_num_retries** and **exsa_retry_period**
12	All Add-on processes must register themselves with Alliance Access using the appropriate API **PcsRegister**. A registration script must be provided for this purpose.
13	Significant events related to the start and the stop of the Add-on and to any message processing must 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 pops up on specified operator screens to display alarm information. Journal events 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: • **Process**: for process start/stop and general information • **Message**: for message processing • **Communication**: if the component provides a link to an external system • **Data**: for internal data handling, like configuration files
16	At installation time, each application component with a GUI must register an **appl** record that will define its icon. For more information, see the Process Control section of the *Alliance Developers Toolkit Developer Guide*.
17	If a GUI needs to exhibit usage restrictions, then it must register a **func** record for every restriction function. If the functions need parameter values, then a **perm** record must be registered for every permission.
18	All Alliance Developers Toolkit API return codes must be trapped and an appropriate message must be logged in the event journal or in the trace file.
19	Interventions must be written in the message history each time a significant event happens to the message (for example, message checked, modified, approved)
20	A message appendix must be created when a message enters or leaves Alliance Access through the component (that is, links to an external system).
21	If a message is modified (Message enrich/repair Add-on), then the original text must be kept in a free-text intervention with a clear indication of what has been changed/added/removed.
22	The Add-on must follow the Alliance Access release cycle. Vendors must at least re-link, re-package, and re-test the Add-on product within three months of receiving a new Alliance Access release. From time to time, code updates must be performed as indicated in the Alliance Release Letter. This applies to both major and minor Alliance Access releases.
23	The add-on must be provided on all platforms supported by Alliance Access (that is, AIX, Linux, Oracle Solaris, and Windows).
24	Dynamic files **must not** be located in the bin or install directory of the ADK component. Placing such files in the bin will result in an integrity verification violation and will be reported by the Integrity Verification Tool (IVT).

Label requirement	**Reference number 13**	**Mandatory**
The Alliance Developers Toolkit component must be compliant with the Alliance Developers Toolkit Developer and Reference Guides.		

8.3 Alliance Access Message Recovery

1	Processes that use the API to update the database must have a recovery process available to deal with API failure (return code ADK_FAILURE).
2	Message processing functions of the Alliance Add-on must be designed with the assumption that it is potentially being started after the crash of an earlier message processing function process. The 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 must 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 must 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 must 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. The recovery process can then identify how much of the normal processing has been performed and possibly activate a roll-back or a process completion.
6	The recovery process must either undo (roll-back) what was done by the normal processing or finish what was left unfinished. In particular, every message must be unreserved at the end of the recovery process.

Label requirement	**Reference number 14**	**Mandatory**
The Alliance Developers Toolkit component must develop a recovery procedure as a follow-up to an Alliance Access failure. This procedure must ensure that no message was lost or resulted in an unknown status.		

8.4 Alliance Access Add-on Configuration and Naming Conventions

To facilitate customer support, 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.

1	Alliance Developers Toolkit server (Component name (four capital letters). Server names must ends with an "S".
2	Alliance Developers Toolkit client (Component name (four 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 Message Journal entries must all start with the component name. This enables easy tracking of Add-on behaviour.
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 15**	**Mandatory**
A clear description of the Alliance Developers Toolkit components must be provided to SWIFT to facilitate customer support.		

8.5 Alliance Access User Documentation

A user and installation manual must be provided together with the Add-on product. The following information must be made available in the product documentation. The aim is to ensure smooth installation, configuration, and daily usage at the customer's 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 the 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. Customise routing point 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 16	Mandatory
A complete user and installation manual must be provided to SWIFT together with every Add-on product.		

8.6 Security Conformance Requirements

Introduction

Cyber-attacks are growing in number and sophistication, and attackers are focusing more deeply on financial institutions. The current cyber-threat environment makes it clear that both SWIFT and its customers must remain vigilant and proactive over the long term. While customers are responsible for protecting their own environments, SWIFT's Customer Security Programme (CSP) has been established to support customers in the fight against cyber-fraud. It introduces a common set of mandatory and advisory security standards that will help foster a more secure financial ecosystem.

The security requirements applicable to Alliance Add-ons are extracted from the SWIFT Customer Security Controls Framework.

Security Conformance Statement

The Security Conformance Statement will include the following controls.

Alliance Add-on providers are expected to comply with the following four controls.

Control	Control objective
1.1 - SWIFT Environment Protection	Ensure the protection of the user's local SWIFT infrastructure from potentially compromised elements of the general IT environment and external environment.
2.1 - Internal Data Flow Security	Ensure the confidentiality, integrity, and authenticity of data flows between local SWIFT-related applications and their link to the operator PC.
2.4A - Back-office Data Flow Security (applicable if ADK also connects to a back office system)	Ensure the confidentiality, integrity, and mutual authenticity of data flows between back office (or middleware) applications and connecting SWIFT infrastructure components.
2.5A - External Transmission Data Protection	Protect the confidentiality of SWIFT-related data transmitted and residing outside of the secure zone.

Because Add-on products fall within the scope of the CSP framework, providers also have an obligation to help their customers comply with not only the previously mentioned four controls, but also with those that these products indirectly support or play a role in. Providers should consider ways in which to support the following controls through the functionality of their add-ons:

Control	Control objective
2.2 – Security Updates	Minimize the occurrence of known technical vulnerabilities within the local SWIFT infrastructure by ensuring vendor support, applying mandatory software updates, and applying timely security updates aligned to the assessed risk.
6.4 – Logging and Monitoring	Record security events and detect anomalous actions and operations within the local SWIFT environment.

For more information about these controls, see the SWIFT Customer Security Controls Framework.

A Summary of Label Requirements

A.1 Label Requirements

Label requirement	Reference number 1	Mandatory
The Alliance Add-on must support the MT and MX related to the solution in question, as listed on the User Handbook. Message support implies the capacity to capture messages for many categories and types, treat them as appropriate, and route them within Access, according to the Add-on treatment result		

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

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

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.		

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

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

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

Label requirement	Reference number 8	Optional
The Alliance Add-on must support the InterAct messaging services and adhere to the latest release of the *SWIFTNet Messaging Operations Guide*.		

Label requirement	Reference number 9	Optional
The Alliance Add-on must support the FileAct messaging service and adhere to the latest release of the SWIFTNet Messaging Operations Guide.		

Label requirement	Reference number 10	Mandatory
The architectural design must be provided to SWIFT for review and advice. The requirements must be assessed during the design phase and the development phase. SWIFT will verify (at SWIFT headquarters) some of these requirements in the product source code and the documentation during the formal presentation of the vendor's product for the Add-on validation. The product must be compliant with the latest versions of Alliance Access software, and must follow the release cycle of Alliance Access. In particular, the product must 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).		

Label requirement	Reference number 11	Mandatory
The Alliance Add-on must support Alliance Access 7.2.		

Label requirement	Reference number 12	Mandatory
The Alliance Add-on should integrate with the updated 64-bit libraries, and be recompiled with 64-bit.		

Label requirement	Reference number 13	Mandatory
The Alliance Developers Toolkit component must be compliant with the Alliance Developers Toolkit Developer and Reference Guides.		

Label requirement	Reference number 14	Mandatory
The Alliance Developers Toolkit component must develop a recovery procedure as a follow-up to an Alliance Access failure. This procedure must ensure that no message was lost or resulted in an unknown status.		

Label requirement	Reference number 15	Mandatory
A clear description of the Alliance Developers Toolkit components must be provided to SWIFT to facilitate customer support.		

Label requirement	Reference number 16	Mandatory
A complete user and installation manual must be provided to SWIFT together with every Add-on product.		

Legal Notices