Toward A Collection of Cloud Integration Patterns

Daniel Ritter[1] and Stefanie Rinderle-Ma[2]

[1] SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
daniel.ritter@sap.com
[2] University of Vienna, Währingerstraße 29, 1090 Wien, Austria
stefanie.rinderle-ma@univie.ac.at

Abstract. Cloud computing is one of the most exciting IT trends nowadays. It poses several challenges on application integration with respect to, for example, security. In this work we collect and categorize several new integration patterns and pattern solutions with a focus on cloud integration requirements. Their evidence and examples are based on an extensive literature review and a system study of "well-established" open source systems.

Keywords: Business Networks, Cloud Middleware, Enterprise Integration Patterns, Integration Systems.

1 Introduction

In this work we revisit the current collection of *Enterprise Integration Patterns* (EIP) in the context of cloud integration. On the basis of a quantitative analysis of several cloud integration scenarios on a well-established platform (i. e., [14]) , we conducted further qualitative literature and system reviews to collect and categorize additional characteristics. These characteristics are verified by cross-referencing them between the quantitative and the qualitative studies, categorized and formulated as a list of patterns or pattern solutions.

2 Cloud Integration Patterns

In this section we collect and define new integration patterns from the cloud integration domain as extensions to the EIPs [5]. The pattern descriptions are represented in the format in Tab. 63.

The categories we consider in this work are storage in Sect. 2.1, messaging patterns like transformation and routing in Sect. 2.2, security in Sect. 2.3, exception handling Sect. 2.4, monitoring and operations in Sect. 2.5, as well as adapter and endpoint patterns in Sect. 2.6.

2.1 Storage Patterns

In addition to the *Message Store* [5], several vendors identified the need for further storage patterns as in Fig. 1, e. g., the storage of variables, scheduler timings in Sect. 2.5, and persistent patterns like *Aggregator* [5] in a data store (cf. Tab. 1). The store can be accessed using a store accessor (cf. Tab. 2).

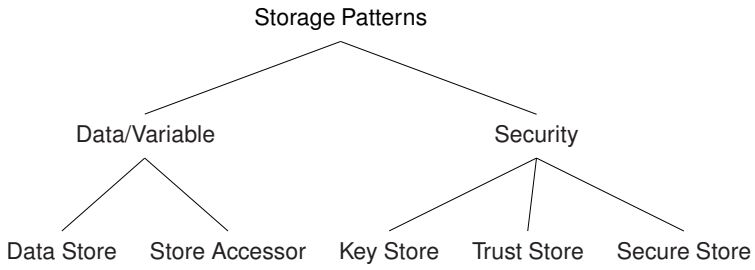

Fig. 1. Storage Patterns.

Table 1. Data Store

Pattern Name	Data Store
Intent	Store arbitrary content explicitly or implicitly from persistent patterns.
Driving Question	How can arbitrary data be stored transient or persistent across instances for access by arbitrary patterns?
Solution	Provide a persistent store, which allows to store variables, value mapping, data of persistent patterns, "in-memory" and transient storage capabilities.
Data Aspects	visibility: local (integration scenario), global (multiple integration scenarios or even solutions), multi-tenant, reliability: redundant, high-available, disaster recoverable; persistent
Variations	e. g., relational column or row store, NoSQL store
Example	Variables, Persistent Scheduler, storage and archiving for legal regulations
Related Patterns	Message Store [5]
Known Uses	partially covered by "Message Store" [6], "Flow reliability" [1], "StoreInKite-Dataset" [2], "DBStorage", "Persist", "Variables" [14]

In addition, several security patterns from Sect. 2.3 require stored secure material like certificates, public/private keys in Tab. 3 or Tab. 4, tokens and user/password in Tab. 5.

Table 2. Store Accessor

Pattern Name	Store Accessor
Intent	Access the data store.
Driving Question	How to access the data store?
Data Aspects	transactions and data access depending on the data store, e. g., query, read, write, delete
Example	Store timings of a Persistent Scheduler, aggregates of an Aggregator and persist (parts of) messages due to legal regulations for archiving.
Related Patterns	Data Store, Message Store [5], Claim Chek [5]
Known Uses	"StoreInKiteDataset (Hadoop)" [2], "DBStorage", "Persist", "Variables" [14]

Table 3. Key Store

Pattern Name	Key Store
Intent	A store that contains private keys, and the certificates with their corresponding public keys.
Driving Question	How can keys and certificates be stored securely?
Solution	Provide a secured storage for security relevant key material, which is accessible from other patterns, but protected from remote access.
Data Aspects	certificates, persistent
Example	Set up and provide access to a Java Key Store from the integration content.
Related Patterns	Trust Store
Known Uses	"JKS" [6], "Key Store" [1], probably [2], implicitly configurable in [14]

Table 4. Trust Store

Pattern Name	Trust Store
Intent	A store that contains certificates from other parties that you expect to communicate with, or from Certificate Authorities that you trust to identify other parties.
Driving Question	How can certificates from other communication partners be stored securely?
Solution	Provide a secured storage for security relevant key material, which is accessible from other patterns. Probably limited access from respective parties is required to store their material.
Data Aspects	certificates, persistent
Related Patterns	Key Store
Known Uses	Trust Store [1]

Table 5. Secure Store

Pattern Name	Secure Store
Intent	A store that contains users and their passwords or secure tokens.
Driving Question	How can users and their passwords or secure tokens be stored securely?
Solution	Secure and reliable storage for User/Password, Token, Expiring Token (cf. Sect. 2.3)
Data Aspects	user/password, token, persistent
Related Patterns	Key Store, Trust Store
Known Uses	"Secure Store" [14]

2.2 Messaging Patterns

We combine the routing and transformation pattern categories from [5] to general messaging patterns for the subsequent discussions.

Routing Patterns For realizing, combined *Scatter-Gather* [5] (e. g., for map-reduce like processing), the multicast is used Tab. 6.

Another new aspect is the communication between instances of several flows within one scenario or even between scenarios on the same platform (i. e., no external call). Therefore, a delegator can be used (cf. Tab. 7). While in the asynchronous case, this can be realized by a messaging system endpoint, e. g., JMS, many vendors offer special mechanisms to avoid an external call.

Table 6. Multicast

Pattern Name	Multicast
Intent	Enable parallel message processing.
Driving Question	How to (statically) enable parallel message processing?
Solution	Send copies of a message to multiple receivers in parallel (statically)
Data Aspects	message creating, read-only, channel cardinality 1:n, non-persistent
Related Patterns	Recipient List [5], Message Dispatcher, Splitter [5]
Known Uses	"Multicast" [6], "Sequential/Parallel Gateway" [14]

Table 7. Delegate

Pattern Name	Delegate
Intent	Exchange messages between several integration scenarios within the same integration system.
Driving Question	How to exchange messages between two or more integration scenarios locally?
Solution	Provide integration system local direct endpoints – preferably using an optimized message exchange format and protocol.
Variations	Asynchronous message exchange via queues, synchronous message exchange via integration system local direct endpoints.
Example	Messaging System endpoint (e. g., JMS, MQTT) for asynchronous messaging, or VM-local or platform local calls for synchrnonous messaging (potentially in an optimized format)
Related Patterns	Channel Adapter [5]
Known Uses	"Direct-VM" endpoint [6], partially covered by "Process Call" [14]

Transformation Patterns The message transformation patterns from [5] are mainly *Content Enricher*, *Content Filter*, and *Claim Check*. The *Message Translator* itself is sketched around the OSI reference model in [5]: data structures, data types, data representation and transport. Following these categories, we

collected patterns as depicted in Fig. 2. The security patterns enable message-level security and are further discussed in Sect. 2.3. The custom processors are theoretically applicable to all categories.

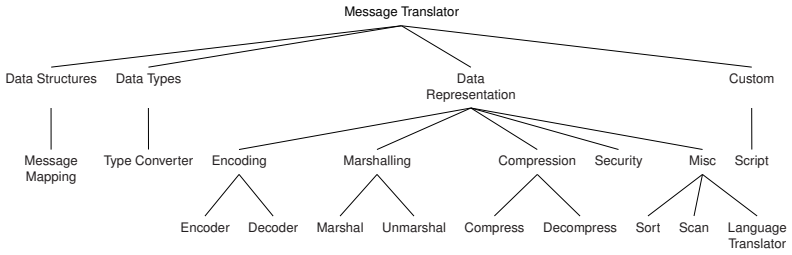

Fig. 2. Message Translator Patterns

The classical *Message Translator* [5] on data structure level is called message mapping and translates standardized data formats between applications. Typical implementations can be found, e. g., as "Morphline Interceptor" in [1] and the "Mapping" component in [14].

In practice, basic converters on data type level (cf. Tab. 8) convert, e. g., between different stream types, reader and writer types, byte arrays and string.

Table 8. Type Converter

Pattern Name	Type Converter
Intent	Convert data types.
Driving Question	How to convert data types?
Solution	Provide special type converters for common types (domain specific).
Example	InputStream to OutputStream, Reader to Writer, ByteArray to String
Related Patterns	Script
Known Uses	"Type Converter" [6], [2], "Morphline Interceptor" [1], implicit in [14]

On the data representation level these converters are complemented by transfer encoders to represent binary data in an ASCII string format (cf. Tab. 9) and decoders (cf. Tab. 10), as well as message protocol marshaller (cf. Tab. 11) and unmarshaller (cf. Tab. 13).

One way to reduce the amount of exchanged data is to compress (cf. Tab. 13) and later decompress the content (cf. Tab. 14).

Some special constructs that were not picked up in general are a special sort pattern (cf. Tab. 15), a scanner (cf. Tab. 16) and an actual language translator (cf. Tab. 17).

Table 9. Encoder

Pattern Name	Encoder
Intent	Represent binary content textually.
Driving Question	How to represent binary content textually?
Solution	Provide standard and custom encoding capabilities for binary formats (domain-specific).
Data Aspects	Binary data encoding
Example	Base16, Base64 (ASCII, MIME content transfer encoding), Radix-64 (OpenPGP)
Related Patterns	Decoder
Known Uses	"Data Format" [6], "Morphline Interceptor" [1], "Base64EncodeContent" [2], "Encoder" [14]

Table 10. Decoder

Pattern Name	Decoder
Intent	Recover encoded binary content.
Driving Question	How to recover encoded binary content?
Solution	Provide standard and custom decoding capabilities for binary data (domain-specific).
Data Aspects	Encoding
Example	Base16, Base64 (ASCII, MIME content transfer encoding), Radix-64 (OpenPGP)
Related Patterns	Encoder
Known Uses	"Data Format" [6], "Morphline Interceptor" [1], "Base64EncodeContent" [2], "Decoder" [14]

Table 11. Marshaller

Pattern Name	Marshaller
Intent	Convert data formats.
Driving Question	How to convert one data format to another?
Solution	Provide standard and custom marshalling capabilities for data formats (domain-specific).
Data Aspects	Encoding
Example	JSON, XML, POJO, SQL
Related Patterns	Unmarshaller
Known Uses	"Data Format" [6], "Morphline Interceptor" [1], "ConvertJSONToSQL" [2], "JsonXMLConverter" [14]

Table 12. Unmarshaller

Pattern Name	Unmarshaller
Intent	Convert data formats.
Driving Question	How to convert one data format to another?
Solution	Provide standard and custom unmarshalling capabilities for data formats (domain-specific).
Data Aspects	Encoding
Example	JSON, XML, POJO, SQL
Related Patterns	Unmarshaller
Known Uses	"Data Format" [6], "Morphline Interceptor" [1], "ConvertJSONToSQL" [2], "JsonXMLConverter" [14]

Table 13. Compress Content

Pattern Name	Compress Content
Intent	Compress message content.
Driving Question	How to compress the message content?
Solution	Provide compression capabilities that compress (parts of) messages.
Data Aspects	Compression algorithm, type
Example	GZIP
Related Patterns	Decompress Content, Message Endpoint, Adapter
Known Uses	"Morphline Interceptor" [1], "Compress Content" [2], "Script" [14]

Table 14. Decompress Content

Pattern Name	Decompress Content
Intent	Decompress message content.
Driving Question	How to decompress message content?
Solution	Provide decompression capabilities that decompress (parts of) messages.
Data Aspects	Decompression algorithm, type
Example	GZIP
Related Patterns	Compress Content, Message Endpoint, Adapter
Known Uses	"Morphline Interceptor" [1], "Compress Content" [2], "Script" [14]

Table 15. Content Sort

Pattern Name	Content Sort
Intent	Sort message content.
Driving Question	How to sort the content of a message?
Solution	Provide configurable sort capabilities that access the message content.
Data Aspects	Algorithms, Comparators
Example	User default comparator in the programming language to sort alpha-numeric content.
Related Patterns	Custom Script, Aggregator
Known Uses	"Sort" [6], "Morphline Interceptor" [1], "Script" [14]

Table 16. Scanner

Pattern Name	Scanner
Intent	Find and replace content.
Driving Question	How to find and replace content?
Solution	Scans the content of a message (for terms that are found in a user-supplied dictionary). If a term is matched, the UTF-8 encoded version of the term will be added to the message.
Data Aspects	optional dictionary
Example	Regex
Related Patterns	Custom Script
Known Uses	"Scan Content" [2], "Morphline Interceptor" [1], "Processor" [6], "Script" [14]

Table 17. Language Translator

Pattern Name	Language Translator
Intent	Translate textual content from one language to another.
Driving Question	How to translate textual content from one language to another?
Solution	Provide standard interface access to natural language translation libraries or services.
Data Aspects	Service bindings, locale
Example	Yandex or Google translator API.
Related Patterns	Custom Script
Known Uses	"Yandex Translator" [2], "Morphline Interceptor" [1], "Processor" [6], "Script" [14]

Besides the specialized translation patterns, language bindings for the execution of custom code are required for flexibility (cf. Tab. 18).

Table 18. Custom Script

Pattern Name	Custom Script
Intent	Provide arbitrary body, header and attachment access and allow modifications.
Driving Question	How to execute custom code during message processing?
Solution	Embed freely programmable language support with access to the message channel and the messages.
Data Aspects	Service bindings, compiled or plain source code (interpreted)
Variations	controlled access by pre-defined, limiting expression language.
Example	Language support for Groovy or Java script with access to the message and other services of the integration system e. g., Store Accessor.
Related Patterns	Content Modifier, Content Enricher
Known Uses	"Morphline Interceptor" [1], "Processor" [6], "Script" [14]

2.3 Security Patterns

The collection of security patterns is grounded our system and literature study on security aspects in integration systems. Subsequently, the security patterns are listed and defined grouped by the security categories shown in Fig. 3.

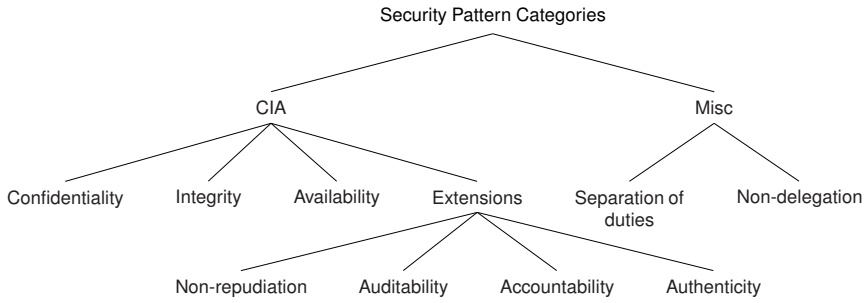

Fig. 3. Security Categories grouped by classical CIA and extensions [17].

Confidentiality Patterns The confidentiality or message privacy patterns collected in Fig. 4 help to ensure that only authorized participants can access integration information like messages, channels, operations and storage. The message Tab. 19, the channel Tab. 20 and the storage Tab. 21 can have the characteristic of having encrypted information (e. g., encryption [7]).

The security-action patterns consist of pair-wise encrypting and decrypting operations (Tab. 22, Tab. 23), endpoints (Tab. 24, Tab. 25), adapters (Tab. 26, Tab. 27) and an encrypting store (Tab. 28).

Integrity and Authenticity These patterns are collected in Fig. 5 and ensure the completeness, accuracy and absence of unauthorized modifications during

Table 19. Encrypted Message

Pattern Name	Encrypted Message
Intent	Rely on a confidential and private piece of information.
Driving Question	How can messages be sent confidential and with data-privacy?
Context	The confidentiality or data-privacy of a message is especially important, when the communication happens via a public network (e. g., cloud integration).
Solution	Message level security; distinguish symmetric, asymmetric; text and categories (categories: channel, message cardinality, input/output, message generating, read/write access).
Data Aspects	text and categories (categories: channel, message cardinality, input/output, message generating, read/write access)
Result	The message is asymmetrically encrypted. The message can only be read by. applying, e. g., a *Decryptor* pattern
Example	PGP, PCKS7
Related Patterns	Message, Encryptor
Known Uses	"PGP Message" [6], "Encrypted Content" [2], implicitly in [14]

Table 20. Encrypted Channel

Pattern Name	Encrypted Channel
Intent	Exchange message over a secure channel.
Driving Question	How can an application send a confidential message such that only the actual receiver can process it?
Solution	Transport level security
Data Aspects	Encrypted transfer, certificates
Example	SSL/TLS, HTTPS
Related Patterns	Encrypted Message
Known Uses	[1], [2], [6], [14]

Table 21. Encrypted Store

Pattern Name	Encrypted Store
Intent	Store messages confidential.
Driving Question	How to store messages confidential?
Solution	Message level security; TODO: stores messages encrypted
Data Aspects	Encrypted content, header and attachments
Example	PGP, PCKS7
Related Patterns	Encryptor, Encrypted Message
Known Uses	Configuration on "DBStorage" [14]

Table 22. Encryptor

Pattern Name	Encryptor
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can messages be sent confidential and with message-privacy as *Encrypted Message*?
Solution	Provide capabilities to encrypt the content, headers and/or attachments of a message.
Data Aspects	in: message, out: encrypted message, key store, public key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Encrypted Message, Decryptor
Known Uses	"Encrypt content" [2], "Encryptor" [14]

Table 23. Decryptor

Pattern Name	Decryptor
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can confidential messages *Encrypted Message* be processed by the actual receiver?
Solution	Provide capabilities to decrypt the content, headers and/or attachments of a message.
Data Aspects	in: encrypted message, out: decrypted message, key store, private key, non-message generating, content modifying.
Example	PGP, PCKS7
Related Patterns	Encryptor
Known Uses	"Decrypt content" [2], "Decryptor" [14]

Table 24. Encrypting Endpoint

Pattern Name	Encrypting Endpoint
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can messages be sent confidential and with message-privacy as *Encrypted Message*?
Solution	Provide capabilities to encrypt the content, headers and/or attachments of a message.
Data Aspects	key store, public key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Encryptor
Known Uses	[14]

Table 25. Decrypting Endpoint

Pattern Name	Decrypting Endpoint
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can confidential messages *Encrypted Message* be processed by the actual receiver?
Solution	Provide capabilities to decrypt the content, headers and/or attachments of a message.
Data Aspects	key store, private key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Decryptor
Known Uses	[14]

Table 26. Encrypting Adapter

Pattern Name	Encrypting Adapter
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can messages be sent confidential and with message-privacy as *Encrypted Message*?
Solution	Provide capabilities to encrypt the content, headers and/or attachments of a message.
Data Aspects	key store, public key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Encryptor
Known Uses	"FileChannel" [1], [14]

Table 27. Decrypting Adapter

Pattern Name	Decrypting Adapter
Intent	When an application sends a confidential message to another participant, its content shall only be read by the receiver.
Driving Question	How can confidential messages *Encrypted Message* be processed by the actual receiver?
Solution	Provide capabilities to decrypt the content, headers and/or attachments of a message.
Data Aspects	key store, private key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Decryptor
Known Uses	[14]

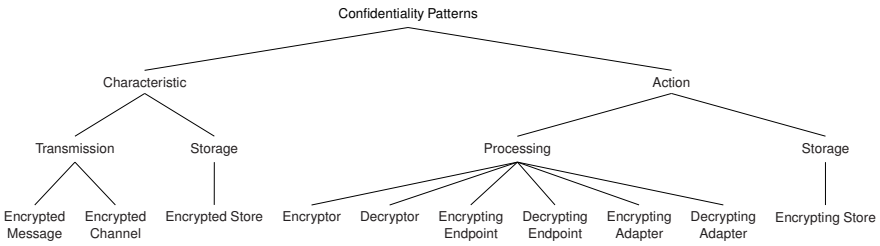

Fig. 4. Confidentiality Patterns grouped by characteristics and actions as well as information states: processing, transmission and storage [8].

Table 28. Encrypting Store

Pattern Name	Encrypting Store
Intent	When a message is stored confidential.
Driving Question	How can messages be stored confidential and with message-privacy as *Encrypted Message*?
Solution	Message level security
Data Aspects	key store, public key, non-message generating, content modifying
Example	PGP, PCKS7
Related Patterns	Encryptor
Known Uses	configurable during "Persist" [14]

message processing and verify the claim of identity. The characteristics of signed and verified messages (Tab. 29, Tab. 30) denote the message level and safe and authenticated channel (Tab. 31, Tab. 33) stand for transport level integrity and authenticity. On storage level, the pattern is called safe store (Tab. 32).

The corresponding action-patterns are the signer and verifier (Tab. 34, Tab. 35).

Table 29. Signed Message

Pattern Name	Signed Message
Intent	Ensuring a message's authenticity, integrity and non-repudiation.
Driving Question	How can a message's authenticity, integrity and non-repudiation be guaranteed?
Solution	Sign a message.
Data Aspects	certificate-based
Related Patterns	Signer
Known Uses	implicitly in [6], [14]

Availability The availability of resources like *Message Store* [5], *Key Store* or *Data Store* are crucial. Table 36 defines the countermeasure as a redundant store pattern.

Table 30. Verified Message

Pattern Name	Verified Message
Intent	Verifying a message's authenticity, integrity and non-repudiation.
Driving Question	How can a message's authenticity, integrity and non-repudiation be verified?
Solution	Verify the signature of a message.
Data Aspects	certificate-based
Related Patterns	Signature Verifier
Known Uses	implicitly in [6], [14]

Table 31. Safe Channel

Pattern Name	Safe Channel
Intent	Ensure integrity on transport level.
Driving Question	How to ensure integrity on transport level.
Solution	Provide integrity on transport level.
Data Aspects	certificate-based
Related Patterns	Signer, Authenticated Channel
Known Uses	"File channel integrity tool" [1], implicitly in [14]

Table 32. Safe Store

Pattern Name	Safe Store
Intent	Ensure integrity on storage level.
Driving Question	How to ensure integrity on storage level.
Solution	Provide integrity on transport level.
Data Aspects	certificate-based
Related Patterns	Signer, Safe Channel
Known Uses	implicitly in [14]

Table 33. Authenticated Channel

Pattern Name	Authenticated Channel
Intent	Ensure authenticity on channel level.
Driving Question	How to ensure authenticity on channel level.
Solution	Ensure authenticity on transport level.
Data Aspects	certificate, user/password, token
Variations	Basic, certificate-based
Example	certificate-based, basic authentication
Related Patterns	Key Store, Trust Store, Secure Store
Known Uses	Kerberos authentication [1], two-way SSL authentication (implicit) [2], [14]

Table 34. Signer

Pattern Name	Signer
Intent	Ensure the authenticity, integrity and non-repudation of the message content.
Driving Question	How can the authenticity of a message be ensured?
Context	Signing messages is especially important, when the communication happens via a public network (e. g., cloud integration).
Solution	The pattern ensures authenticity, integrity and non-repudiation on the message-level through signing the content or parts using security mechanisms like digital signatures or envelopes. The signer cannot deny signing afterwards and any change to the signed message parts can be detected using, e. g., the *Verifier* pattern. *Message Translator* patterns might invalidate the signing.
Data Aspects	in: any message, out: message with signed content or parts; requires private key, key store
Result	The signing is done by asymmetric cryptography that requires for example the creation of a signature using a signing algorithm with the private key stored in a secured key store.
Example	PGP, PCKS7
Related Patterns	Key Store, Trust Store
Known Uses	"Camel Crypto" [6], "Signer" [14]

Table 35. Signature Verifier

Pattern Name	Signature Verifier
Intent	Verify a message's authenticity, integrity and non-repudiation
Driving Question	How to verify a message's authenticity, integrity and non-repudiation?
Solution	Provide certificate-based verification on message level.
Data Aspects	in: Signed Message, out: Signed Message (read-only); exception: verification failed; requires access: certificate, Key Store, non-message generating, read-only
Variations	Signature Verifier, XMLSignatureVerifier
Related Patterns	Signer, Signed Message, Key Store, Trust Store
Known Uses	"Verifier" [14]

Table 36. Redundant Store

Pattern Name	Redundant Store
Intent	Ensure availability of storage resources.
Driving Question	How to ensure availability of storage resources?
Solution	Provide redundant hardware, service (probably even high-available and disaster recoverable)
Data Aspects	availability level: HA, DR
Variations	Failover, HA, DR
Related Patterns	Message Store, Data Store, Key Store, Trust Store, Secure Store
Known Uses	implicitly in [14]

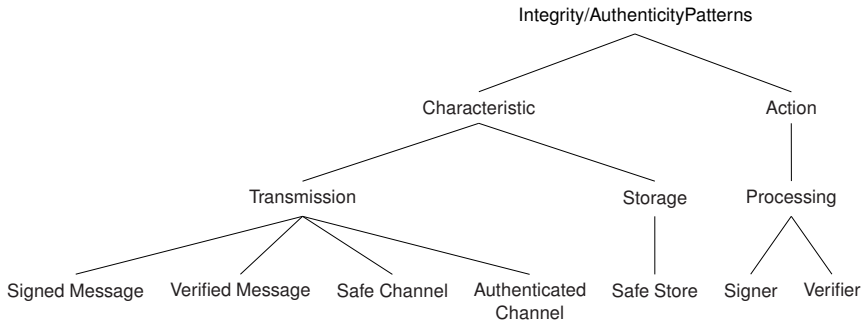

Fig. 5. Integrity and Authenticity patterns grouped by characteristics and actions as well as information states: processing, transmission and storage [8].

Non-repudiation / Auditability / Accountability Being able to prove which event happenend during message processing when and with which privileges (role) and by whom (user) is important for detecting security issues as well as using the information for metering.

Table 37. Audit Log

Pattern Name	Audit Log
Intent	Keep a record of those events you consider relevant for (security) auditing: accountability, non-repudiation, auditability
Driving Question	How to keep a record of those events considered relevant for (security) auditing.
Solution	Provide a secure logging capability, which tracks events and (especially) all its own configurations.
Data Aspects	Log entries
Example	Record event-based endpoints, unsuccessful message send events, changes to the audit log.
Related Patterns	Monitor
Known Uses	implicitly in [14]

Authorization, Non-Delegation

Table 38. Token

Pattern Name	Token
Intent	Grant role-based access to a user.
Driving Question	How to grant role-based access to a user?
Solution	Provide secure token, which is passed as part of each during conversation.
Example	OAuth
Related Patterns	Secure Store
Known Uses	"GetTwitter" [2], "Secure Paramters" in SAP HCI ADK [14]

Table 39. Expiring Token

Pattern Name	Expiring Token
Intent	Improve security for role-based access of users via Tokens.
Driving Question	How to improve security, when granting role-based access to a user via Tokens?
Solution	Reduce the validity of a Token through expiration.
Data Aspects	Token
Example	OAuth
Related Patterns	Secure Store
Known Uses	"GetTwitter" [2], "Secure Paramters" in SAP HCI ADK [14]

Table 40. Refresh Token

Pattern Name	Refresh Token
Intent	Deal with expiring tokens.
Driving Question	How to deal with expiring tokens?
Solution	Allow to (automatically) re-negotiate expiring tokens?
Data Aspects	Expiring Token
Example	OAuth
Related Patterns	Secure Store
Known Uses	"GetTwitter" [2], "Secure Paramters" in SAP HCI ADK [14]

Table 41. Token-based Authorizer

Pattern Name	Token-based Authorizer
Intent	Validate token-based authorizations.
Driving Question	How to validate token-based authorizations?
Solution	Provide authorization checks for messages with tokens.
Data Aspects	Secure Tokens
Example	OAuth
Related Patterns	Secure Store, Adapter, Message Endpoint
Known Uses	Pluggable Authorization [2], SAP Social Media ADK adapters [14]

Table 42. Principle Propagation

Pattern Name	Principle Propagation
Intent	
Driving Question	How to delegate authorizations to a third-party?
Solution	Allow authorization forwarding.
Data Aspects	Token
Related Patterns	Token
Known Uses	"Principle Propagation" [15]

2.4 Exception Patterns

The exception patterns were introduced and analyzed in [13], thus summarized in Fig. 6 and not further discussed:

- Failover Router
- Compensation Sphere
- Exception Sphere
- Validate Message
- Timeout Operation
- Message Redelivery on Exception
- Delayed Redelivery
- Skip Operation
- Stop Operation Local
- Stop Operation All
- Pause Operation
- Throw
- Raise Incident (similar to monitoring pattern Sect. 2.5)
- Catch Selective
- Catch All

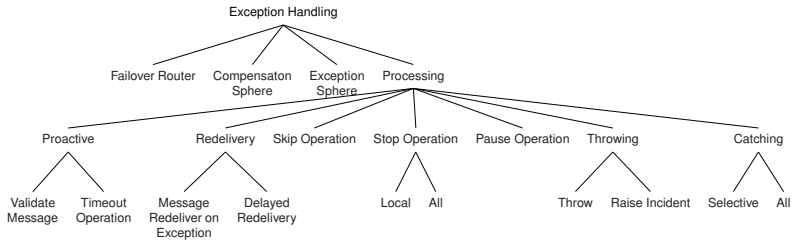

Fig. 6. Exception Handling Patterns.

2.5 Monitoring and Operation Patterns

The current EIP cover basic monitoring and operation patterns [5]: *Control Bus* for administrating the messaging system, *Wire Tab* for routing message copy e. g., to a *Message Store* for monitoring, *Message History* for provenance, and *Smart Proxies* for asynchronous message tracking.

In addition, there are more aspects to monitoring and operations that can be found in current integration system implementations. Some of them are collected subsequently (others like the circuit breaker Sect. 2.6 can be found in related sections). Figure 7 gives an overview of the collected monitoring patterns.

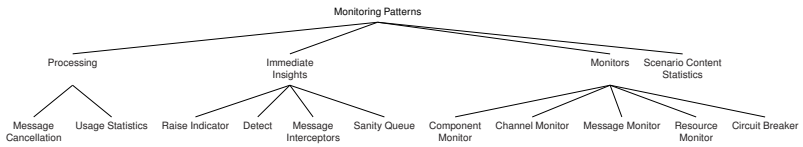

Fig. 7. Monitoring Patterns.

The processing of messages might require their cancellation (cf. Tab. 43), e. g., based on defined usage statistics (cf. Tab. 44)).

Table 43. Message Cancellation

Pattern Name	Message Cancellation
Intent	Cancel the processing of a message.
Driving Question	How to cancel the processing of a message?
Solution	Define a condition under which the message will not be processed further.
Data Aspects	condition
Related Patterns	Message Expiration [5], Stop Operation (Local,All) [13], Validate Message [13]
Known Uses	[14]

Table 44. Usage Statistics

Pattern Name	Usage Statistics
Intent	A measure for the usage of components.
Driving Question	How to measure the usage of components during message processing?
Solution	Define performance indicators for components.
Data Aspects	usage indicators, persistent
Related Patterns	Scenario Content Statistics
Known Uses	[14]

While the whole message processing and failure handling shall be automatic, there might be situations (e. g., due to business semantics), which require raising an indication (cf. Tab. 45) for manual post-processing or information. Sometimes this can be preceded by the detection of a special situation (cf. Tab. 46). The information can be collected by interceptors (cf. Tab. 47) and provided asynchronously by a sanity queue (cf. Tab. 48).

Table 45. Raise Indicator

Pattern Name	Raise Indicator
Intent	Inform about an important event during message processing.
Driving Question	How to inform about an important event during message processing?
Solution	Raise an indicator.
Data Aspects	persistent
Example	Send e-mail, Monitor
Related Patterns	Detect
Known Uses	“Indicator” [2], “Alert” [14]

Table 46. Detect

Pattern Name	Detect
Intent	Detect inactive, active components like channel or pattern/activity.
Driving Question	How to detect inactive components like channels or patterns?
Solution	Send out an alert when a component did not have any data for a specified amount of time.
Data Aspects	conditions
Related Patterns	Usage statistics, Channel Monitor, Component Monitor
Known Uses	“Monitor Activity” [2]

Table 47. Message Interceptor

Pattern Name	Message Interceptor
Intent	Gather usage statistics on channel level.
Driving Question	How to gather usage statistics on channel level?
Solution	Inject a configurable “listener” between components.
Data Aspects	conditions, read-only, non-message creating
Related Patterns	Usage statistics, Wire Tap
Known Uses	“Interceptor” [6], “Interceptor” [1]

The respective events and incidents can be tracked by monitors, e. g., for components (cf. Tab. 49), channels (cf. Tab. 50) and messages (cf. Tab. 51). In

Table 48. Sanity Queue

Pattern Name	Sanity Queue
Intent	Stay informed about a system's sanity.
Driving Question	How to stay informed about the system's sanity?
Solution	Create sanity queues for important aspects and register on events.
Data Aspects	queues, persistent
Related Patterns	Wire Tap
Known Uses	JMS sanity check

addition, the system resource information (cf. Tab. 52) as well as the overview of all open circuits (cf. Tab. 56) might be of interest.

Table 49. Component Monitor

Pattern Name	Component Monitor
Intent	Measure usage statistics and behavior of a component.
Driving Question	How to measure usage statistics and behavior of a component?
Solution	Monitor configurable characteristics of a component and store them persistently as statistics.
Data Aspects	Configurations, statistical records, KPIs, persistent
Example	Monitor throughput, exceptions and raise indicator or load balance, if KPIs are not fulfilled.
Related Patterns	Stop Operation, Pause Operation (both from [13]), Failover Router [13], Data Store, Smart Proxy [5]
Known Uses	"Adapter Monitor" [9], conceptually supprted by [11]

For analytical reasons, it might be important to have statistics on the used components and adapters (cf. Tab. 53).

The operational aspects mostly come from reliability and distributed system requirements as shown in Fig. 8.

Upon them are the persistent scheduling (cf. Tab. 54) and the persistent cluster lock (cf. Tab. 54).

Table 50. Channel Monitor

Pattern Name	Channel Monitor
Intent	Measure communication with endpoints.
Driving Question	How to measure success, failures (exceptions thrown by endpoint), timeouts for requests?
Solution	Monitor configurable characteristics of the communication with endpoints and store them persistently as statistics.
Data Aspects	Configurations, statistical records, KPIs, persistent
Example	Monitor timeout to endpoints and decide to apply a Circuit Breaker pattern.
Related Patterns	Circuit Breaker, Timeout Synchronous Request, Failover Router [13], Command, Data Store, Smart Proxy [5]
Known Uses	"Adapter Monitor" [9], conceptually supported by [11]

Table 51. Message Monitor

Pattern Name	Message Monitor
Intent	Monitor the states of all processed messages.
Driving Question	How to monitor the states of all currently processed messages?
Solution	Provide a status monitor that shows the Message History.
Data Aspects	message provenance data
Related Patterns	Message History
Known Uses	"Message Processing Log"[14]

Table 52. Resource Monitor

Pattern Name	Resource Monitor
Intent	Monitor the system resources.
Driving Question	How to monitor the system resources and react on critical situations?
Solution	Provide a status monitor that shows the system's recources and allow the definition of thresholds for raising indicators.
Data Aspects	resource statistics
Example	Memory consumption, CPU and network load
Related Patterns	Raise indicator
Known Uses	"MonitorDiskUsage", "MonitorMemory" [2]

Table 53. Scenario Content Statistics

Pattern Name	Scenario Content Statistics
Intent	A measure for the usage of content.
Driving Question	How to measure the usage of content within integration scenarios?
Solution	Define performance indicators for the usage of content.
Data Aspects	content statistics
Example	Number of used adapters and their configuration
Related Patterns	Usage statistics
Known Uses	[14]

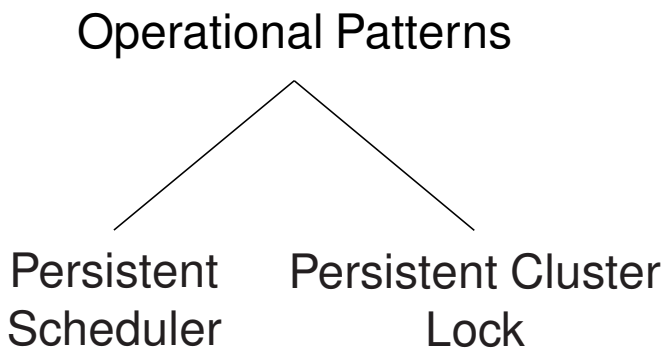

Fig. 8. Operational Patterns.

Table 54. (Persistent) Scheduler

Pattern Name	(Persistent) Scheduler
Intent	Start from last message or action after restart.
Driving Question	How to start scheduling from the last message or action after system restart?
Solution	Persistently store the state of the scheduler.
Data Aspects	persistent
Related Patterns	Data Store, Polling Consumer
Known Uses	[6], [14], conceptually supported in [3]

2.6 Endpoint and Adapter Patterns

Many of the channel access and endpoint patterns for connecting applications with messaging systems (i. e., *Message Channel*, *Channel Adapter*, *Message Endpoint*, *Durable Subscriber*, *Channel Purger*, *Transactional Client*, *Selective Consumer*, *Polling Consumer*, *Message Dispatcher*, *Event-Driven Consumer*), and for involving existing systems in the message exchange (i. e., *Envelope Wrapper*, *Messaging Bridge*) are covered by [5].

In addition, [12] collected some more integration adapter and quality of service (QoS) related patterns, shown in Fig. 9, which are not further discussed here:

- Commutative Endpoint
- Redelivering Endpoint
- Adapter Flow
- Synch/Asynch Bridge
- Asynch/Synch Bridge
- Protocol Switch

Table 55. (Persistent) Cluster Lock

Pattern Name	(Persistent) Cluster Lock
Intent	Prevent several integration scenarios from processing the same information concurrently.
Driving Question	How to prevent several integration scenarios from processing the same information concurrently?
Solution	Set persistent lock.
Data Aspects	persitent
Related Patterns	Data Store, Competing Consumers [5]
Known Uses	[14]

- Cross scenario Processing
- Cross tenant Processing
- Best Effort Processing
- At least once Processing
- At most once Processing
- Exactly once Processing
- Exactly once in order Processing

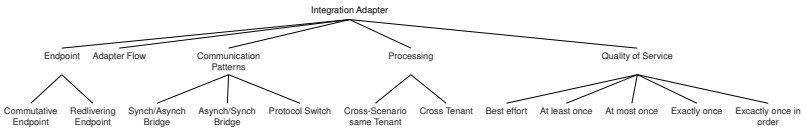

Fig. 9. Integration Adapter and QoS Patterns.

Recent practical advances for cloud-based *Microservice* architectures [10], e. g., in the area of fault-tolerance like *Hystrix* [9], can be mapped to integration systems and summarized to general patterns as shown in Fig. 10.

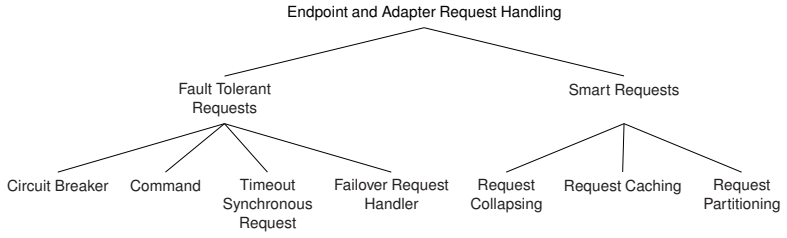

Fig. 10. Integration Adapter and Endpoint Request Patterns.

Thereby the related aspects of fault-tolerance and smart requests are differentiated. For fault-tolerance, one wants to make request more tangible (cf. Tab. 57), to penalize long running requests (cf. Tab. 56), break them by timeouts (cf. Tab. 58) and find alternative endpoints that can answer the requests instead (cf. Tab. 59).

Table 56. Circuit Breaker

Pattern Name	Circuit Breaker
Intent	Stop all requests to a particular endpoint/application for a period of time
Driving Question	How to stop all requests to a particular endpoint/application for a period of time?
Context	Communication with endpoints/applications
Solution	Provide capabilities to monitor the error percentage, when communicating with a particular endpoint and automatically or manually prevent its invocation for a period of time.
Data Aspects	Timing, monitoring statistics
Result	Reduces number of failed message exchanges.
Related Patterns	Electronic circuits
Known Uses	"Circuit Breaker" [9], conceptually from [11]

Table 57. Command

Pattern Name	Command
Intent	Make remote calls tangible.
Driving Question	How to make remote calls tangible?
Solution	Wrap remote calls into one command context.
Result	The command context allows to, e. g., timeout and monitor a request.
Related Patterns	Similar to "Adapter Flow" in [12], Request Collapsing
Known Uses	"Reactive Command" [9]

The smart handling of requests helps to stabilize the system as well. However, the main focus lies on an intelligent way of requesting information from endpoints, e. g., by combining several (cross-scenario) requests to one (cf. Tab. 60), or leveraging already recently requested information for multiple subsequent requests (cf. Tab. 61) or separating request aspects to multiple requesters (cf. Tab. 62).

Table 58. Timeout Synchronous Request

Pattern Name	Timeout Synchronous Request
Intent	Timeout remote calls that take longer than a configured threshold.
Driving Question	How to timeout remote calls that take longer than a configured threshold?
Context	Long blocking requests destabilize the scenario and potentially the integration system.
Solution	Apply a timeout to each synchronous request.
Data Aspects	Timing
Result	Unblocks the scenario and system.
Related Patterns	Command, "Delayed Redelivery", "Stop Operation" both from [13], Message Expiration [5]
Known Uses	"Timeout" [9], conceptually from [11]

Table 59. Failover Request Handler

Pattern Name	Failover Request Handler
Intent	Constructively handling failed requests.
Driving Question	How to handle failed or timed-out requests or short-circuits constructively?
Solution	Perform fallback logic when a request fails, is rejected, times-out or short-circuits.
Data Aspects	Configuration of failover channels or endpoints/adapters
Variations	Applicable to endpoints and adapters.
Example	If there is a communication error with the currently selected endpoint, then the request handler automatically fails-over to the next endpoint in the list.
Related Patterns	Command, Failover Router [13]
Known Uses	"Failover Client" [1]

Table 60. Request Collapsing

Pattern Name	Request Collapsing
Intent	Reduce the number of endpoint calls.
Driving Question	How to reduce the number of endpoint calls?
Solution	Collapse multiple requests into a single endpoint dependency call.
Data Aspects	Data aggregation, selection and projection handling
Variations	Applicable to endpoints and adapters.
Related Patterns	Command, similar to "Adapter Flow" in [12]
Known Uses	"Request Collapsing" [9], conceptually supported by [16] for database access from business process engines.

Table 61. Request Caching

Pattern Name	Request Caching
Intent	Reduce the amount of duplicate requests to the same endpoint.
Driving Question	How to reduce the amount of duplicate requests (from several scenarios) to the same endpoint?
Context	Several integration scenarios might request the same information from the same endpoint (concurrently).
Solution	Cache request contexts/keys to connect only once and feed many integration scenarios with the required information.
Data Aspects	Data aggregation, selection and projection handling, (persistent) caching, request correlation
Variations	Applicable to endpoints and adapters.
Related Patterns	Command, Data Store
Known Uses	similar to "Request Caching" in [9], conceptually supported by [16] for database access from business process engines.

Table 62. Request Partitioning

Pattern Name	Request Partitioning
Intent	Isolate request dependencies and limit concurrent access to them.
Driving Question	How to isolate requests and limit concurrent access to any of them?
Solution	Manually or automatically isolate request dependencies.
Data Aspects	Split queries (where possible) and merge results
Result	Partitions request failures as well for a more robust system: by partitioning requests to endpoints, errors are confined to one request aspect (e. g., get business partner header data) as opposed to cancelling the entire request (e. g., business partner header and associated documents).
Variations	The partitions can be hardware redundancy, binding certain processes to certain CPUs, segmenting different areas of business functionality to different server farms, or partitioning requests into different information aspect groups for different functionality.
Related Patterns	Bulkhead, Command, Request Collapsing
Known Uses	"Isolation" in [9], conceptually similar to "Bulkhead" from [11]

References

1. Apache Foundation. Apache Flume. https://flume.apache.org/, 2015.
2. Apache Foundation. Apache Nifi. https://nifi.apache.org/, 2015.
3. A. Böhm. *Building Scalable, Distributed Applications with Declarative Messaging.* PhD thesis, University of Mannheim, 2010.
4. C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter. *Cloud Computing Patterns - Fundamentals to Design, Build, and Manage Cloud Applications.* Springer, 2014.
5. G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
6. C. Ibsen and J. Anstey. *Camel in Action.* Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.
7. M. Leitner, S. Schefer-Wenzl, S. Rinderle-Ma, and M. Strembeck. An experimental study on the design and modeling of security concepts in business processes. In *The Practice of Enterprise Modeling - 6th IFIP WG 8.1 Working Conference, PoEM 2013, Riga, Latvia, November 6-7, 2013, Proceedings*, pages 236–250, 2013.
8. J. McCumber. Information systems security: A comprehensive model. In *Proceedings of the 14th National Computer Security Conference*, 1991.
9. Netflix. Hystrix. https://github.com/Netflix/Hystrix/wiki, 2015.
10. S. Newman. *Building Microservices: Designing Fine-Grained Systems.* O'Reilly, Beijing, 2015.
11. M. T. Nygard. *Release It! Design and Deploy Production-Ready Software.* Pragmatic Bookshelf, Raleigh, NC, 2007.
12. D. Ritter and M. Holzleitner. Integration adapter modeling. In *Advanced Information Systems Engineering*, pages 468–482. Springer, 2015.
13. D. Ritter and J. Sosulski. Modeling exception flows in integration systems. In *18th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2014, Ulm, Germany, September 1-5, 2014*, pages 12–21, 2014.
14. SAP SE. SAP HANA Clold Integration. https://cloudintegration.hana.ondemand.com, 2015.
15. SAP SE. SAP Process Integration. http://help.sap.com/nwpi, 2015.
16. M. Vrhovnik. *Optimierung datenintensiver Workflows: Konzepte und Realisierung eines heuristischen, regelbasierten Optimierers.* PhD thesis, Universitt Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2011.
17. M. E. Whitman and H. J. Mattord. *Principles of Information Security.* Course Technology Press, Boston, MA, United States, 3rd edition, 2007.

List of Patterns

A Pattern Format

In this section the pattern format used in this paper is described. The format, shown in Tab. 63, is defined similar to existing pattern formats of the EIP [5] and related pattern descriptions (e. g., from cloud computing [4]).

Table 63. Pattern Format Template

Pattern Name	Name
Intent	at the beginning of each pattern, its purpose and goal is shortly stated, to describe what the solution represented by the pattern contains.
Driving Question	This question captures the problem that is answered by the pattern. Stating this question at the beginning allows readers to identify if the pattern fits the problem they have in a concrete use case.
Context (optional)	This section describes the environment and forces leading to the problem solved by the pattern. It also may describe why naive solutions can be unsuccessful or suboptimal. Other patterns may be referenced here.
Solution	The solution section briefly states how the pattern solves the problem raised by the driving question. It is kept brief, because readers shall be enabled to quickly read the intent, question, and solution sections to get an idea what the pattern is doing in detail. The solution section is commonly closed with a sketch depicting the architecture of the solution.
Data Aspects (optional)	All message, pattern content and configuration aspects are described in this section.
Result (optional)	In this section, the solution is elaborated in greater detail. The architecture proposed by the sketch is described and the behavior of the application after implementation of the pattern is discussed. New challenges that may arise after a pattern has been applied may also be included here, together with references to other patterns addressing these new challenges.
Variations (optional)	Often, patterns can be applied in slightly different forms. If the differences of these variations are not significant enough to justify their description in a separate pattern, they are covered in the variations section.
Example (optional)	This section gives an illustrative example of the pattern.
Related Patterns	Several patterns are often applied together as they are solving related problems, but the application of one pattern may also exclude other patterns from being applicable. These interrelations of patterns are described in this section. It, therefore, forms the structure of the integration pattern language and guides readers through the set of patterns.
Known Uses	Existing applications implementing a pattern, products offering a pattern or supporting its implementation, are covered here exemplarily.