1 Legacy AMQP Exchange Binding Support

Versions of AMQP prior to 1.0 prescribed a model of Exchanges and Queues, where Queues were bound to Exchanges with a binding key whose meaning depended upon the type of the Exchange. In order to allow a consistent mechanism for addressing legacy AMQP Exchanges over AMQP 1.0 the following filter types are defined. Use of an Exchange as an address for a Source thus can be seen as equivalent to constructing exclusive queues bound to an Exchange in legacy AMQP versions.

Containers which support the filters that are defined in this section should advertise the capability *APACHE.ORG:LEGACY_AMQP_EXCHANGE_FILTERS* in their connection capabilities when sending the open performative, and MUST provide this capability on sources supporting these filter types.

1.1 Legacy AMQP Direct Binding

```
<type name="legacy-amqp-direct-binding" class="restricted" source="string" provides="filter">
    <descriptor name="apache.org:legacy-amqp-direct-binding:string" code="0x0000468C:0x00000000"/>
</type>
```

The legacy-amqp-direct-binding filter consists of a described string value. The filter matches a message if and only if the described string value exactly matches the subject field of the Properties section of the message being evaluated. If the message has no Properties section, or if the subject field of the Properties section is not set, then the legacy-amqp-direct-binding filter does not match.

1.2 Legacy AMQP Topic Binding

```
<type name="legacy-amqp-topic-binding" class="restricted" source="string" provides="filter">
    <descriptor name="apache.org:legacy-amqp-topic-binding:string" code="0x0000468C:0x00000001"/>
</type>
```

The legacy-amqp-topic-binding filter consists of a described string value. The value value described by the type is interpreted as a pattern to match against the subject field of the Properties section of the message being evaluated.

- The pattern is formed using zero or more tokens, with each token delimited by the "." character. The tokens "#" and "*" have special meanings.
- The token consisting of the single character "*" matches a single word in the subject field.
- The token consisting of the single character "#" matches zero or more words in the subject field.

Thus the filter value "*.stock.#" would match the subjects "usd.stock" and "eur.stock.db" but not "stock.nasdaq".

If the message has no Properties section, or if the subject field of the Properties section is not set, then the legacy-amqp-topic-binding filter matches only if the value of the filter is a single "#".

1.3 Legacy AMQP Headers Binding

```
<type name="legacy-amqp-headers-binding" class="restricted" source="map" provides="filter">
    <descriptor name="apache.org:legacy-amqp-headers-binding:map" code="0x0000468C:0x00000002"/>
</type>
```

The legacy-amqp-headers-binding filter consists of a described map value. The map value described by the type is interpreted as a pattern to match against the application-properties section of the message being evaluated. The map has the same restriction as the application-properties section, namely that the keys of this are restricted to be of type string (which excludes the possibility of a null key) and the values are restricted to be of simple types only, that is, excluding map, list, and array types.

The key “x-match” in the described map has special meaning. This key MUST map to the symbolic value “any” or the symbolic value “all” within the described map. All other keys which begin “x-” MUST be ignored by the source when evaluating. If the value for “x-match” is “all” then all other valid key-value pairs in the map MUST match with an entry with the same key in the application-properties section. If the value for “x-match” is “any” then the filter will accept the message if at least one key-value pair matches the equivalent key value pair in the application-properties section.

A key-value pair in the filter’s map matches a key-value pair in the application-properties section if the keys are identical (including the same type), or if the value in the filter map for the key is null.

2 Java Message Service Support

The Java Message Service defines two types of filtering of messages: firstly the ability to exclude from a subscription messages sent by the same connection, secondly a more general filtering syntax known as "selectors" based on an SQL like syntax.

Containers which support the filters that are defined in this section should advertise the capability *APACHE.ORG:JMS_FILTERS* in their connection capabilities when sending the open performative, and MUST provide this capability on sources supporting these filter types.

2.1 Jms No Local Filter

```
<type name="jms-no-local-filter" class="composite" source="list" provides="filter">
    <descriptor name="apache.org:jms-no-local-filter:list" code="0x0000468C:0x00000003"/>
</type>
```

A message will be accepted by the jms-no-local-filter if and only if the message was originally sent to the container of the source on a separate connection from that which is currently receiving from the source.

2.2 Jms Selector Filter

```
<type name="jms-selector-filter" class="restricted" source="string" provides="filter">
    <descriptor name="apache.org:jms-selector-filter:string" code="0x0000468C:0x00000004"/>
</type>
```

The Java Message Service "selector" defines an SQL like syntax for filtering messages. The selector filters based on the values of "headers" and "properties". The defined JMS headers can be mapped to equivalent fields within the AMQP message sections:

```
JMS Property Name | AMQP 1.0 Field
==================|================================================
JMSCorrelationID  | correlation-id field of properties section
JMSDeliveryMode   | durable field of header section
JMSDestination    | to field of the properties section
JMSExpiration     | absolute-expiry-time of properties section
JMSMessageID      | message-id of properties section
JMSPriority       | priority field of header section
JMSRedelivered    | delivery-count > 0 in header section
JMSReplyTo        | reply-to in properties section
JMSTimestamp      | creation-time of properties section
JMSType           | annotation x-opt-jms-type in message-annotations section
```

Figure 1: Mapping JMS Headers to AMQP fields

The "properties" of the JMS message are equivalent to the AMQP application-properties section. Thus a reference to a property Foo in a message selector would be evaluated as the value associated with the key "Foo" (if present) in the application-properties section.

The operands of the JMS selector are defined in terms of the types available within JMS, When evaluated against the application properties section, the values within that section MUST be evaluated according to the following type mapping.

AMQP Type	JMS Selector Type
null	null
boolean	boolean
ubyte	short
ushort	int
uint	long
ulong	long
byte	byte
short	short
int	int
long	long
float	float
double	double
decimal32	double
decimal64	double
decimal128	double
char	char
timestamp	long
uuid	byte[16]
binary	byte[]
string	String
symbol	String

Figure 2: Mapping AMQP types to JMS types

3 Logical Operators

The filters field of the source allows for a number of filters to be associated with a source, with a message having to meet the filtering criteria of all filters in order to be passed through. This mechanism allows for a primitive notion of combining filters, but only in logical conjunction. In order to build more complex filters it is useful to allow for a full set of logical combinations.

Containers which support the filters that are defined in this section should advertise the capability *APACHE.ORG:LOGIC_FILTERS* in their connection capabilities when sending the open performative, and MUST provide this capability on sources supporting these filter types.

3.1 Or Filter

```
<type name="or-filter" class="restricted" source="list" provides="filter">
    <descriptor name="apache.org:or-filter:list" code="0x0000468C:0x00000005"/>
</type>
```

The or-filter, as its name implies, acts as a logical disjunction of multiple filters. That is the filter accepts a message if at least one of the filters within its list accept the message.

3.2 And Filter

```
<type name="and-filter" class="restricted" source="list" provides="filter">
    <descriptor name="apache.org:and-filter:list" code="0x0000468C:0x00000006"/>
</type>
```

The and-filter, as its name implies, acts as a logical conjunction of multiple filters. That is the filter accepts a message if and only if all of the filters within its list accept the message.

3.3 Not Filter

```
<type name="not-filter" class="composite" source="list" provides="filter">
    <descriptor name="apache.org:not-filter:list" code="0x0000468C:0x00000007"/>
    <field name="filter" type="*" requires="filter" mandatory="true"/>
</type>
```

The not-filter, as its name implies, acts as a logical negation of the nested filter. That is the filter accepts a message if and only the filter it wraps does not accept the message

Field Details

`filter` *the filter to be negated*