

SMPTE RECOMMENDED PRACTICE

BXF SDK Documentation



Table of Contents

1	Scope	4
2	Conformance Notation	4
3	Normative References	5
4	Terms and Definitions	5
4.1	Traffic System (TS)	5
4.2	Automation System (AS)	6
4.3	Traffic Agent (TA)	6
4.4	Automation Agent (AA)	6
4.5	Associate System	6
4.6	Agent	6
4.7	Partner	6
4.8	Validator	6
4.9	SDKBox	6
5	The SDK Structure	7
5.1	With external transport level	7
5.2	With embedded transport adaptor	9
5.3	Agents	9
5.4	Extended Agent	12
5.5	Transport Adapter (HTTP Post Adapter)	15
5.6	BXF Validator	15
5.7	Loggers	16
5.8	Exception Handling	20
5.1	SDKBox	20
5.2	Assembly	22
5.3	Configuration	24
5.4	BXFMessageLogger	24
5.5	Error Logger	24
5.6	User Interface Tool	25
6	SDK Types	26
6.1	SDKEvent	26
6.2	SDKEventAsRun	29

6.3	SDKSchedule	33
6.4	SDKAsRunList	36
6.5	SDKDubTransfer	38
6.6	SDKPurgeTransfer	40
6.7	SDKDubList	42
6.8	SDKPurgeList	43
6.9	SDKException	44
Annex A	SDK Internal Types (informative)	46

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 34CS

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

This section is entirely informative and does not form an integral part of this Engineering Document.

When the BFX suite of documents (SMPTE 2021-x) was first published, interoperability issues were sometimes encountered between systems because of different communication schemes, protocols and security constraints.

This Recommended Practice aims to solve these problems, specifically between Traffic and Automation systems. Automatic serialization classes are utilized for working with the BFX xml-based messages, which provides a baseline solution for the system interoperability.

The BFX SDK supports 3 types of transmitted objects between Traffic System and Automation:

- 1) Native BFX xml-based messages
- 2) Deserialized generic objects
- 3) Automation-friendly objects – an event and a list of events

An implementer can select any type to exchange data between the Traffic system and Automation system.

What follows is a basic model of the BXF SDK, which includes three layers between Traffic System and Automation System:

1. Traffic SDK Endpoint
2. Messages Bus (Mule ESB). The Mule ESB is utilized as the transport bus because of its simplicity and flexibility.
3. Automation SDK Endpoint

See figs below.

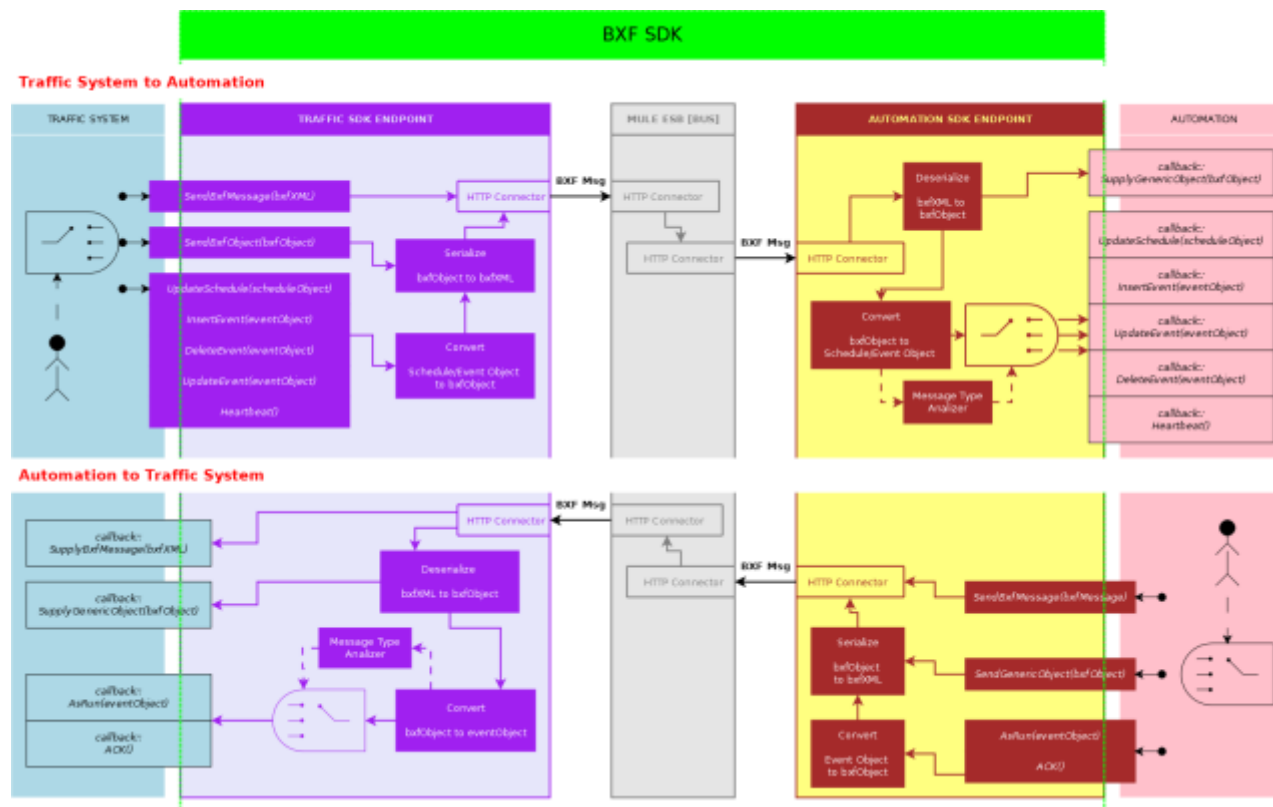


Fig 1. Basic Model

Direction	Traffic System → Automation	Automation → Traffic System
Module		
TRAFFIC SDK ENDPOINT	A collection of procedures to report a traffic schedule list changes	Callback procedures to send messages from Automation

BUS (Mule ESB)	Redirect xml messages from traffic endpoint to automation endpoint	Redirect xml messages from automation endpoint to traffic endpoint
AUTOMATION SDK ENDPOINT	Callback procedures from Automation to supply data from traffic system	A collection of procedures to report a automation play list changes

Fig 2. High-level API solutions

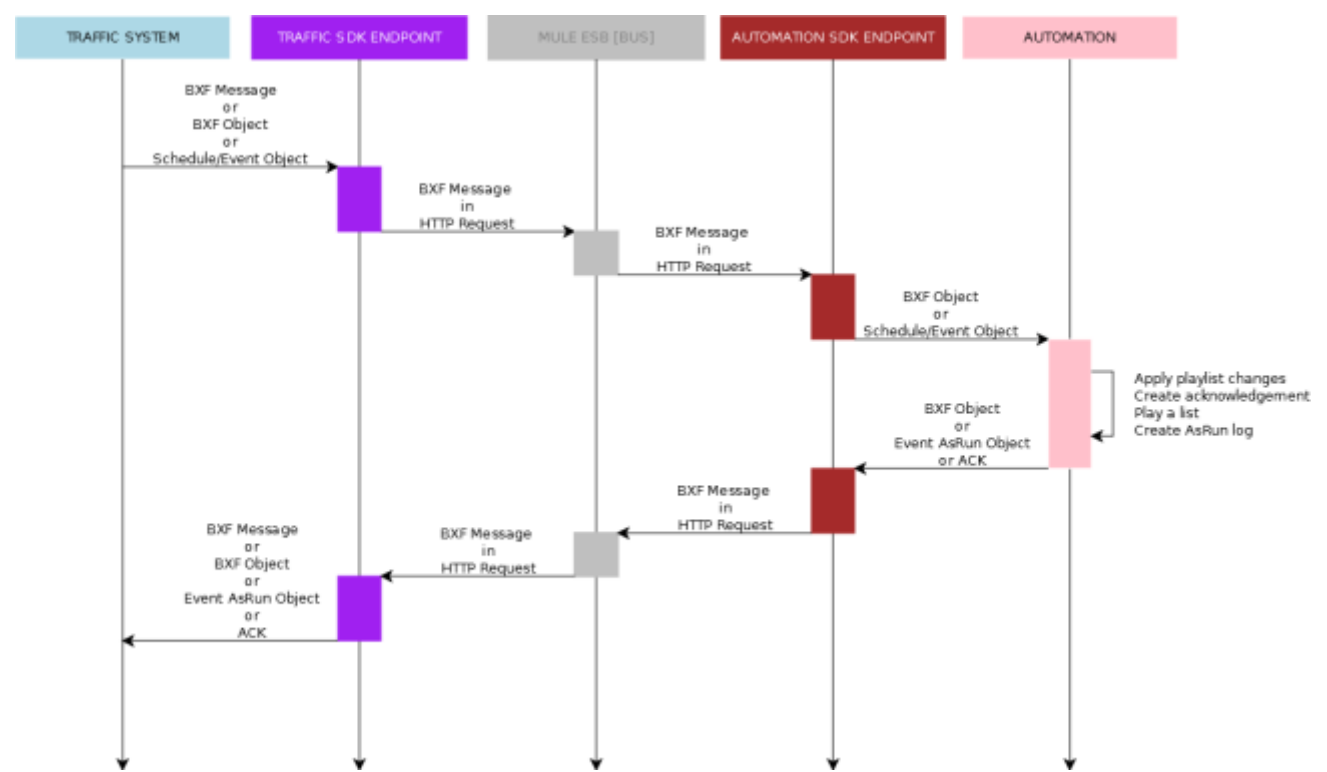


Fig. 3 Basic interaction workflow

1 Scope

This Recommended Practice provides documentation of the BXF SDK, a set of DLLs provided alongside this document, intended to facilitate improved interoperability among Traffic and Automation Systems.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful

to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; then formal languages; then figures; and then any other language forms.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this engineering document. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this engineering document are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

There are no normative references

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

4.1 Traffic System (TS)

A system used in the construction of playout schedules.

4.2 Automation System (AS)

A system used in the on-air execution of playout schedules.

4.3 Traffic Agent (TA)

An SDK module processing on TS side.

4.4 Automation Agent (AA)

An SDK module processing on AS side.

4.5 Associate System

TS for TA and is AS for AA

4.6 Agent

Can refer to a Traffic Agent or Automation Agent.

4.7 Partner

Contra-side system or agent : Traffic Agent or Traffic System for a corresponding Automation Agent, or an Automation Agent or Automation System for a corresponding Traffic Agent.

4.8 Validator

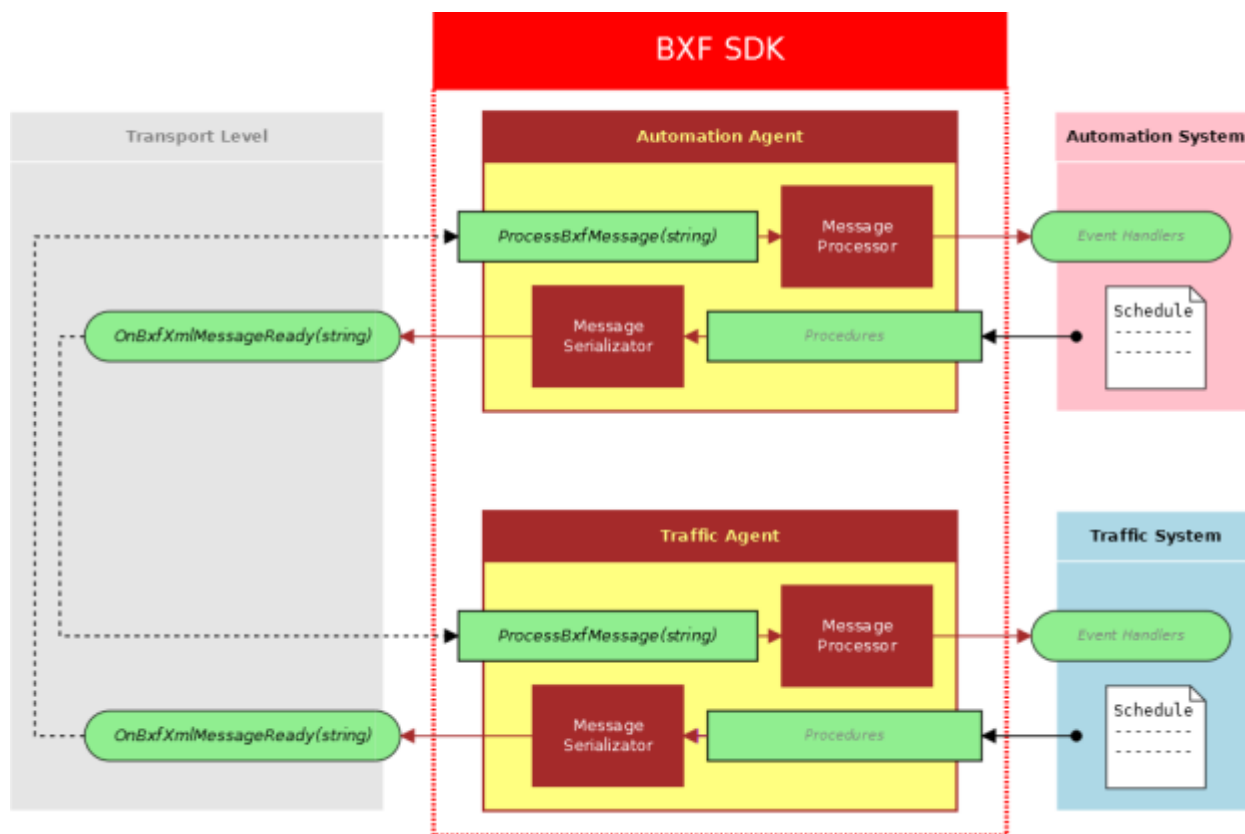
An SDK module used for validating BXF XML messages against xsd schema

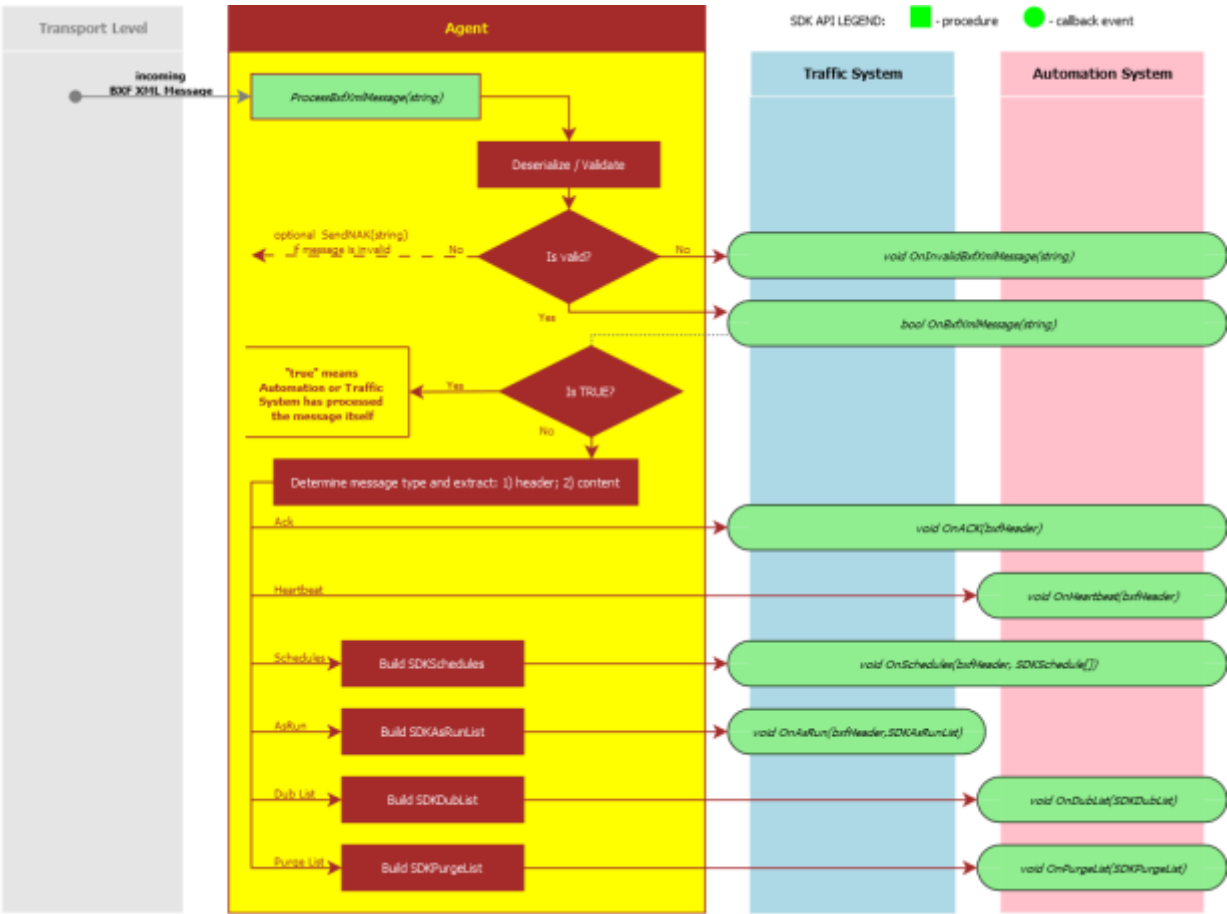
4.9 SDKBox

An SDK high level module used to hold and manage SDK modules (Transport Adapters, Agent, Validator)

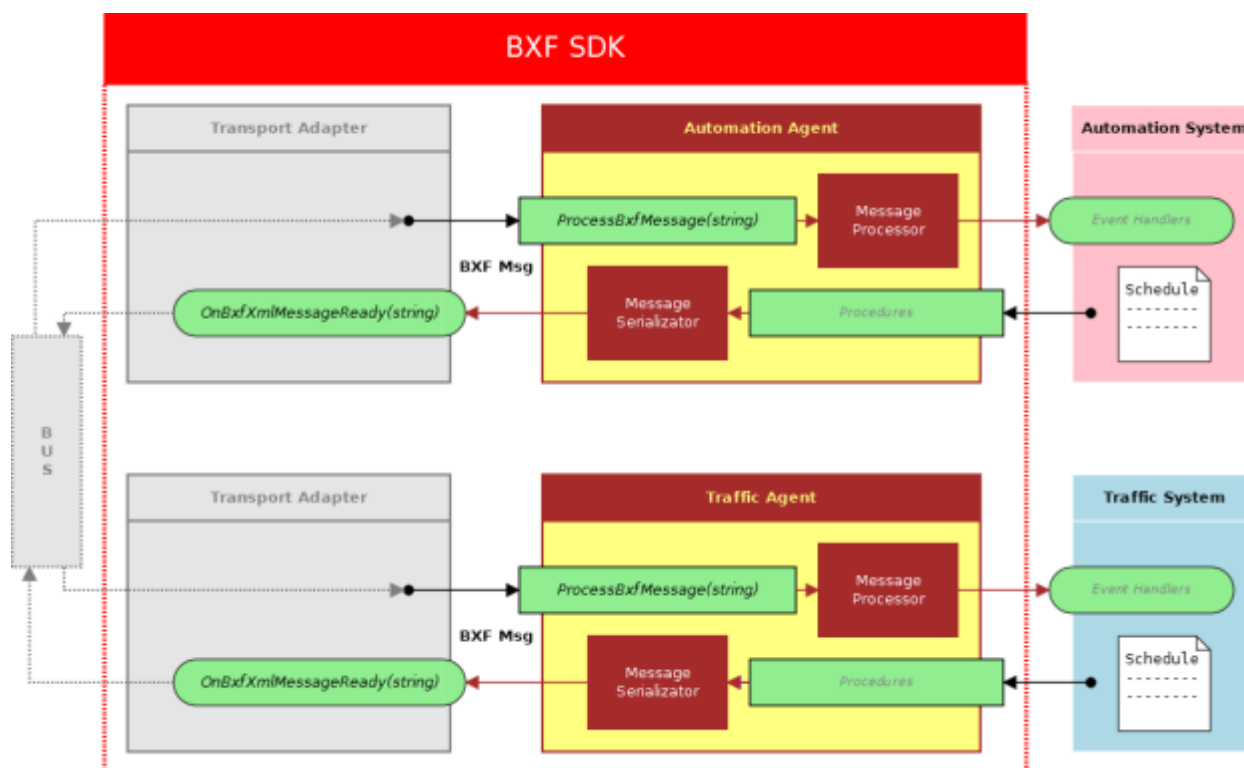
5 The SDK Structure

5.1 With external transport level





5.2 With embedded transport adaptor

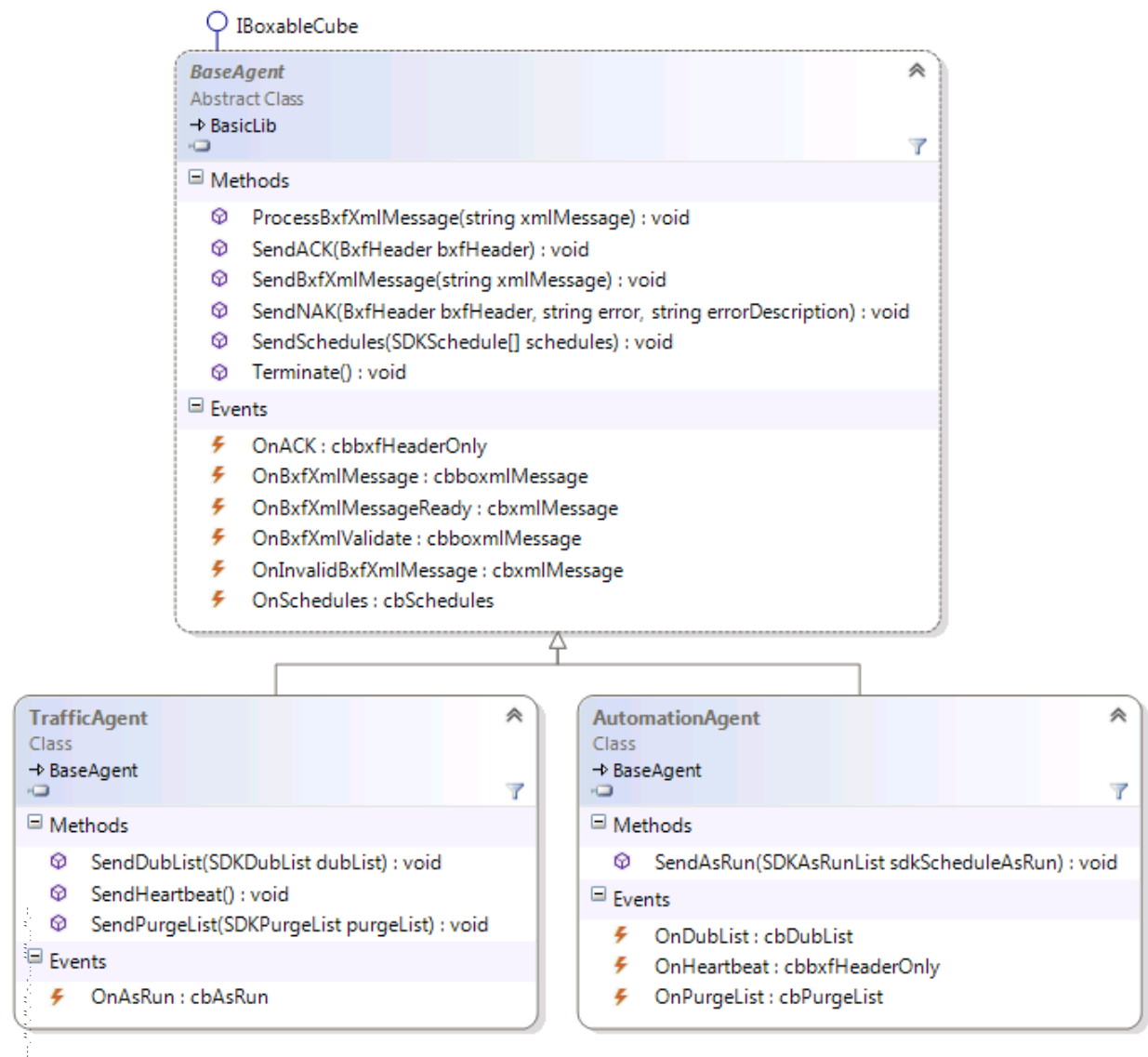


5.3 Agents

Traffic/Automation Agent – BFX SDK module which interacts with Traffic/Automation System.

The Agents:

- receive bxf messages from transport level (Transport Adapter or another implementation)
- validate and deserialize bxf message
- call event handlers from Traffic System and provide received data as
 - a) original raw xml string
 - b) deserialized BFX object
 - c) simplified SDK object for events and schedules (SDKEvent/SDKSchedule types)



Method:	Terminate()	Safe terminating
To Partner:		
Event:	OnBxfXmlMessageReady	Occurs when an agent has got bxf message and is ready to give it to a Partner
Method:	ProcessBxfXmlMessage(string xmlMessage)	Receives and processes the message from a Partner.
To Associate System:		
Events:	OnBxfXmlMessage	Occurs when an agent has got raw BXF message and is ready to give

		one to its Associate System. Event handler (callback function) should return boolean value: <i>true</i> – if Associated System processes BFX message itself, <i>false</i> – if Associated System does not process BFX message.
	OnBfxMessageValidate	Occurs when an agent needs to validate BFX XML message. Validator event handler (callback function) should return boolean value: <i>true</i> – if BFX message is valid, <i>false</i> – otherwise.
	OnInvalidBfxMessage	Occurs when an agent determines that received BFX message is not correct.
	OnACK	Occurs when an agent receives ACK BFX message from a partner
	OnSchedule	Occurs when an agent receives a schedule from a partner
Methods:	SendBfxXmlMessage(string xmlMessage)	Sends raw BFX XML string <i>xmlMessage</i> from Associate System. Return nothing.
	SendSchedules(SDKSchedule[] schedules)	Sends array of schedules in SDKSchedule format from Associate System Returns nothing.
	SendACK(BfxHeader bfxHeader)	Sends Acknowledgement from Associated System for received message with <i>bfxHeader</i> header. Returns nothing.
	SendNAK(BfxHeader bfxHeader, string error, string errorDescription)	Sends Error Acknowledgement from Associated System for received message with <i>bfxHeader</i> header. <i>error</i> – short error definition <i>errorDescription</i> – detailed error description Returns nothing.

Traffic Agent only

Event:	OnAsRun	Occurs when a TA receives AsRun BFX message.
Method:	SendHeartbeat	Creates and sends heartbeat BFX message to Partner.
	SendDubList(SDKDubList dubList)	Sends Duplication content transfer message
	SendPurgeList(SDKPurgeList purgeList)	Sends Purge content transfer message

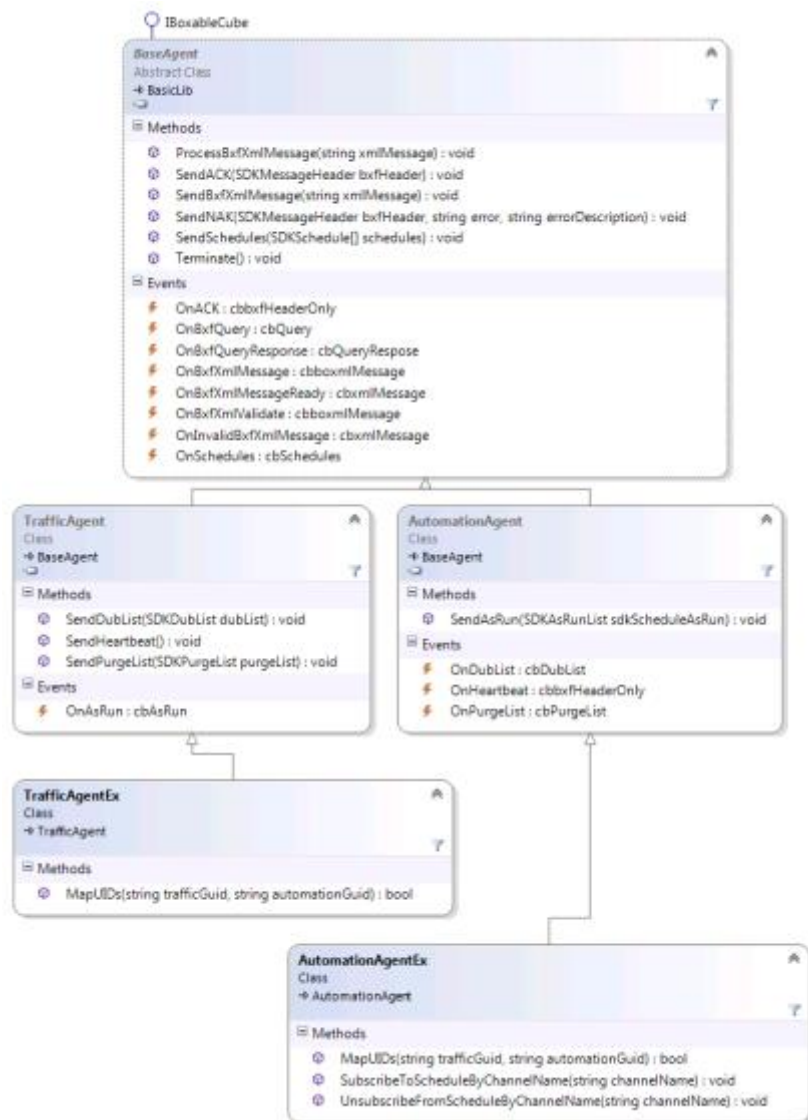
Automation Agent only

Event:	OnHeartbeat	Occurs when AA receives heartbeat BFX message.
	OnDubList	Occurs when an automation receives a duplication content transfer list from traffic system
	OnPurgeList	Occurs when an automation receives a purge content transfer list from traffic system

Method: SendAsRun(SDKAsRunList
asRunList, bool complete)

Sends AsRun list in SDKAsRunList format to Partner,
If *complete* is *false*, AsRunList items are in short basic form, *otherwise* in
extended complete form
Return nothing.

5.4 Extended Agent



Extra features which are implemented in extended version:

1. Event GUID substitution
2. Subscription to selected channel

Extended Traffic Agent

Method:	MapUID(string trafficGuid, string automationGuid)	Map event traffic ID to automation ID
----------------	--	--

Extended Automation Agent

Method:	MapUID(string trafficGuid, string automationGuid)	Map event automation ID to traffic ID
	SubscribeToScheduleByChannelName (string channelName)	Subscribe an agent to process schedules for selected channels only
	UnsubscribeFromScheduleByChannelName (string channelName)	Remove subscription to channel

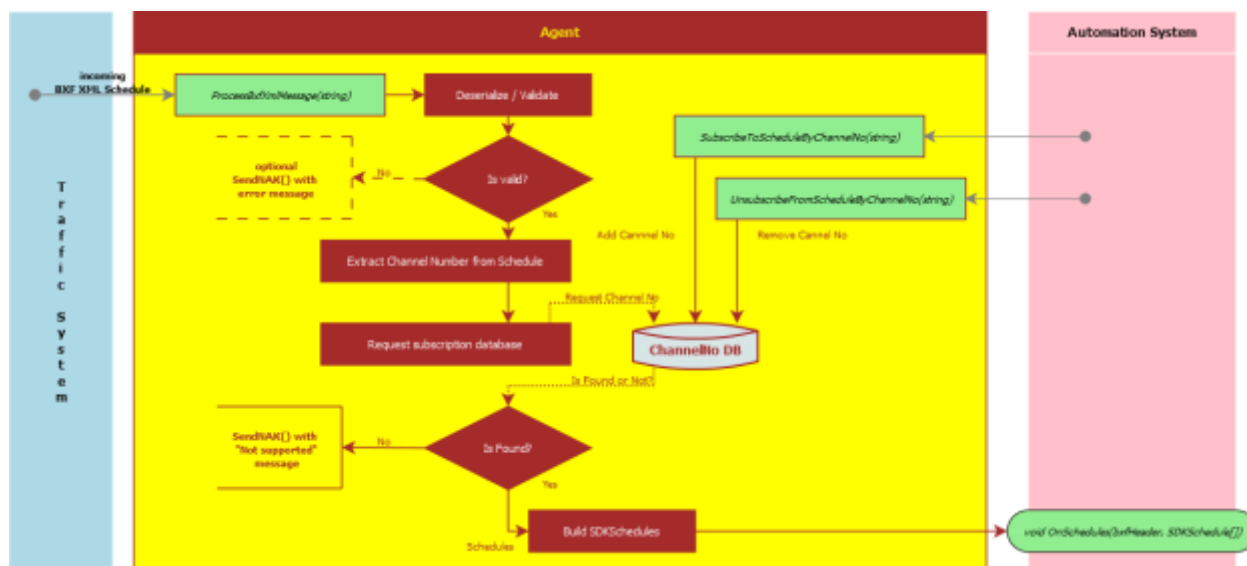


Fig. 1 Schedule subscription API

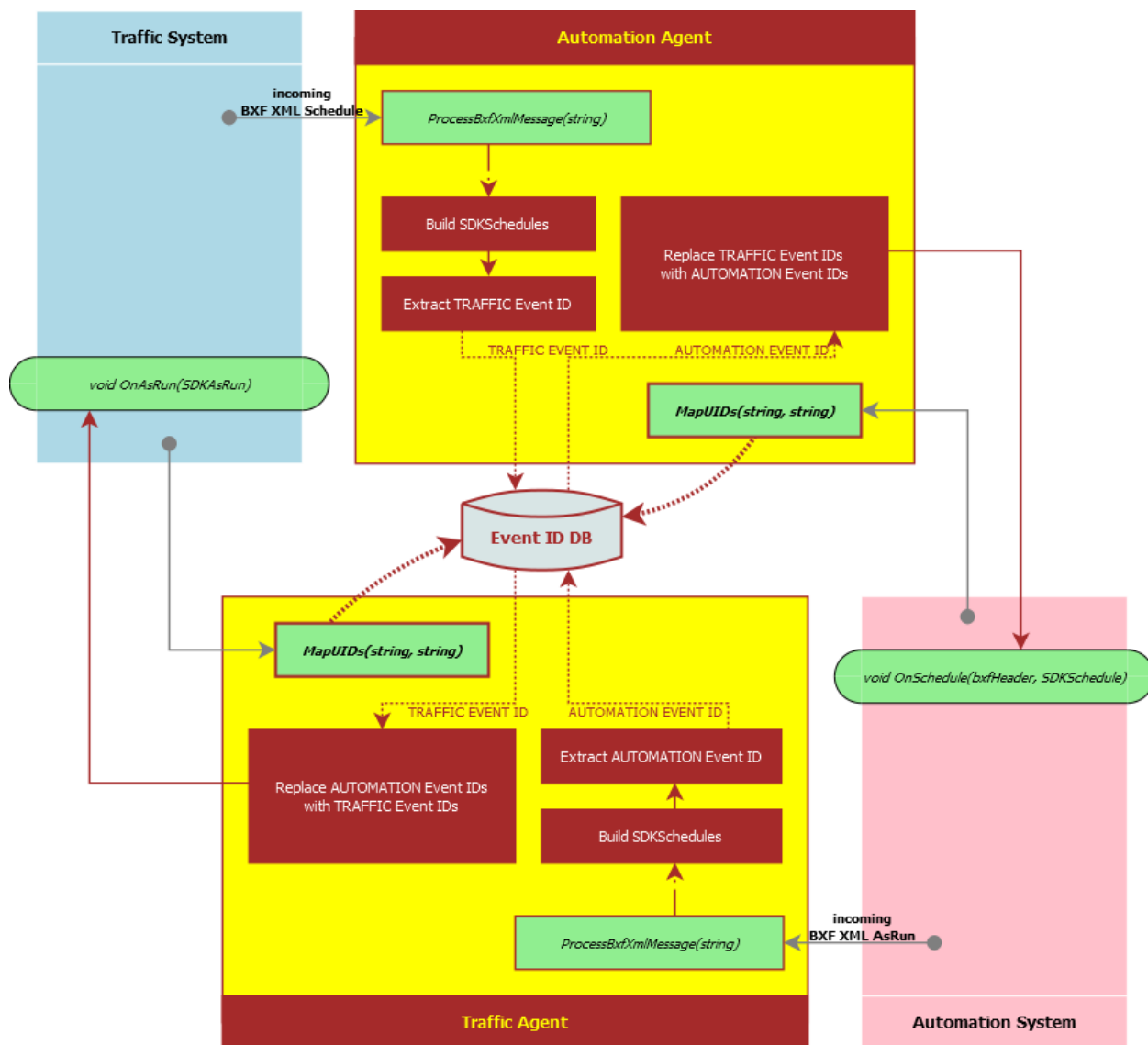


Fig. 2 Event ID mapping and replacement

If the Automation System and Traffic System generate their own UIDs for the same event, then the SDK can substitute IDs in an agent module while processing a schedule and an asrun list. When an agent receives a schedule or an asrun list, it requests a special database (guidmap.db) for alternate event uid.

MapUID (string trafficGuid, string automationGuid) is a procedure to add a new pair of trafficId and automationId into the database for further usage.

5.5 Transport Adapter (HTTP Post Adapter)

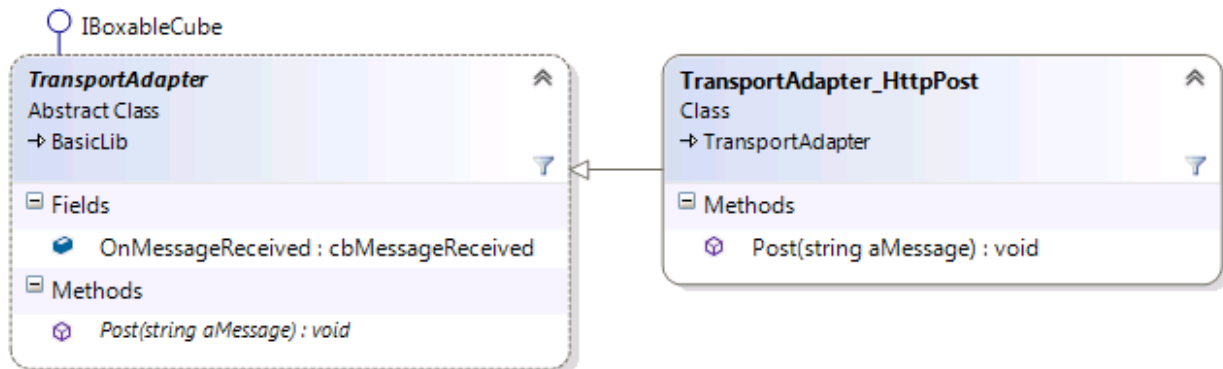


Fig. 3 TransportAdapter and TransportAdapter_HttpPost types

Event:	OnMessageReceived	Occurs when the adapter receives a string message from a Partner
Method:	Post(string aMessage)	Posts string message to a Partner. Returns nothing

5.6 BXF Validator

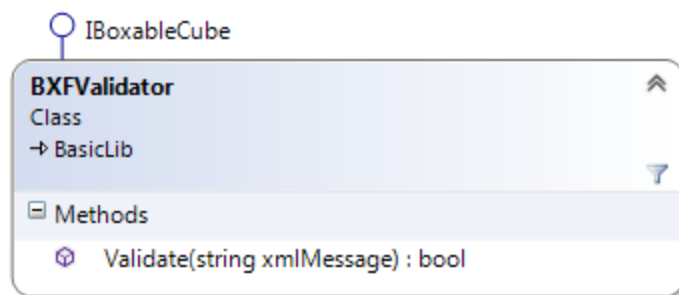


Fig. 4 BXFValidator type

Method:	Validate(string xmlMessage)	Validates XML string message against to BXF XSD schema. See Configuration section. Returns: <i>true</i> - if the <i>xmlMessage</i> is valid xml document
----------------	------------------------------------	---

		<i>false</i> - otherwise
--	--	--------------------------

5.7 Loggers

5.7.1 BXF Message Logger

BxfMessageLogger collects incoming (to SDK) and outgoing (from SDK) BXF messages to the Log/BxfMessages/YYYY-MM-DD folder (where YYYY-MM-DD is the date when a message has been processed).

Each message is saved as a file with a x-hh-mm-ss-nnn.xml name

Where:

x – prefix “in” for incoming messages and “out” for outgoing messages

hh-mm-ss-nnn – time when the message has been processed, nnn is msec.

BXFMessageLogger is implemented as a singleton.