

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 1 of 65

SiLA Device Control & Data Interface Specification

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 2 of 65

Changes

Last storage: 28-Mar-14 12:03

Datum	Version	Reason	Description	Author
26-Feb-2009	0.0-001	First edit		Bepa
03-Mar-2009	0.0-002		Include discussion result from meeting on 27-Feb-2009	Bepa
04-Mar-2009	0.0-003		General Edit	Bepa
19-Mar-2009	0.0-004		Include discussion result from meeting on 17-Mar-2009	Bepa
27-Apr-2009	0.0-008		Include discussion result of security meeting	Bepa
07-May-2009	0.0-009		Include discussion result of performance meeting	Bepa
19-May-2009	0.00-010		Include changes of H. Baer (Section 2, Network Interface and Performance)	Bepa
19-May-2009	0.00-011		Include changes of R. Hochstrasser.	Bepa
20-May-2009	0.00-012		Include state machine	Bepa
25-May-2009	0.00-013		Prepare meeting 26.05.2009	Bepa
26-May-2009	0.00-014		Include feedback from Meeting	Bepa
08-Jun-2009	0.00-015		Last changes before meeting 09.06.2009	Bepa
09-Jun-2009	0.00-016		Include feedback of discussion at 09.06.2009	Bepa
11-Jun-2009	0.17	Import into Aligned Element	Document imported in Aligned Element for proper history tracking and traceability. Input consolidated and considered from Remo H. and Jörg J.	CKA
12-Jun-2009	0.18	Feedback of Interface Workgroup members	Further Input of Jakub B. and Henning B. considered and formatting improved	CKA
15-Jun-2009	0.19	Ready for design review	Document reviewed and corrected on formal and language aspects.	DSP
22-Jul-2009	0.20	Feedback of design review	Feedback of the design review considered according to the decisions made on the interface workgroup meeting of 21.07.09	CKA
14-Aug-2009	0.21	Interface Workgroup meeting	Further input added of the interface workgroup meeting from 11-Aug-2009	CKA
03-Sep-2009	0.22	Feedback of the Interface Workgroup meeting	SP 88, SP 94 and SP 89 Table Return Values: «All commands aborted» removed. A full change history is available in the requirements management tool.	CKA
03-Sep-2009	1.0	Release Version	Document Version changed from 0.22 to 1.0	CKA
16-Nov-2009	1.01	Include Tickets and concerns from InterfaceWG		RHO
23-Nov-2009	1.02	Include new StateMachine		BPA / RHO
02-Dez-2009	1.0.3	contet in Aligned Element updated for	Summary Tag added in several documentation sections	CKA

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 3 of 65

		porper traceability	Register and UnRegister Command moved from chapter 9.2 to Chapter 9.1	
08-Dez-2009	1.0.4	Part II included	Chapter 9.2 reordered Part II (Data Classes) included	HBA/CKA
17-Dez-2009	1.0.5	Chapter 3.5	State machine revised	CKA
18-Dez-2009	1.0.6	Chapter 8 and 3.5	Changes Henning added / State machine figures updated	CKA
18-Dez-2009	1.0.7	Input from Henning	-	CKA
22-Dez-2009	1.0.8	Input from Henning	-	CKA
07-Jan-2010	1.0.9	GIS core Team inputs	Direction as Command parameter removed New Reference layout Event commands separated State Machine update (draft) Formatting improved / grammar and spelling corrected	CKA
11-Jan-2010	1.0.10	GIS core Team inputs	Update Chapter 3.5	CKA
11-Jan-2010	1.0.11	GIS core Team inputs	Return Value1 in Section 8.1 replaced by 2 and 3	CKA
12-Jan-2010	1.0.12	GIS core Team inputs	State Machine updated Formatting and grammar improved	CKA
13-Jan-2010	1.0.13	GIS core Team inputs	Formatting and grammar corrected	CKA
19-Jan-2010	1.0.14	Feedback GIS Meeting 19.01.09	SubState inError added State digramms updated	CKA
16-Feb-2010	1.0.15	Review Input / GIS Meeting 16.02.09	Ticket 41 ... 67	CKA
01-Mar-2010	1.0.16	Review Input	Ticket 41 ... 67	CKA
02-Mar-2010	1.0.17	Input during GIS core Team Meeting of 02.03.2010	-	CKA
11-Mar-2010	1.0.18	Ticket #41, #81, #87	-	CKA
16-Mar-2010	1.0.19	Input during GIS Meeting of 16.03.2010	Spelling, grammar and formatting improved	CKA
16-Apr-2010	1.0.20	Input during GIS Meeting of 13.04.2010	Numerous tickets considered	CKA
29-Apr-2010	1.0.21	Review Feedback	-	CKA
30-Apr-2010	1.1.00	Units added	Release candidate	CKA
30-Apr-2010	1.1.01	Changeset 215	Release candidate	CKA
18-Mar-2011	1.1.02	Several tickets in trac	Suggestions for v1.2	HBA
17-Jan-2012	1.2.01	Final 1.2 version	Final version, resetting state, new _1.2.xsd schemas, etc.	HBA
29-May-2013	1.3.00	New AutoIP, ServiceDiscovery, SiLA domain, and elimination of Nullable Column	Implementation of ChangeRequests	FPE
30-May-2013	1.3.01	Merge to 1.3	Wrong basis used (1.2.00), fixed.	FPE
18-Jun-2013	1.3.02	Review comments arrived	Fixtures and additional improvements as per Reviews	FPE
20-Jun-2013	1.3.03	Further minor reviews on syntax	Additional Fixtures	FPE
11-Oct-2013	1.3.04	Fixes on some text, references of documents updated and WSDL example added	Possibility to have information about parallelism and queues in WSDL	FPE

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 4 of 65


24-Oct-2013	1.3.05	Syntax fixes and clarifications	Comments from Bernd fixed	FPE
29-Oct-2013	1.3.06	WSDL Examples added	Examples of WSDL and ER in references section added	FPE
30-Oct-2013	1.3.07	Final version ready for Balloting	Left agreed sections and eliminated not agreed ones	FPE
13-Nov-2013	1.3.08	Yellow highlights removed	Clarification about SOAP-Import is agreed and clarification about UTF-8 stream configuration is platform dependent	FPE

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 5 of 65

1	Scope of Document.....	7
1.1	Notational conventions.....	7
2	Architecture.....	7
2.1	SiLA Service Provider.....	9
2.1.1	Commands.....	9
2.1.2	Events.....	14
2.1.3	Error Handling.....	14
2.1.4	Simulation.....	15
2.1.5	Service Configuration.....	15
2.1.6	Network Interface.....	16
2.1.7	SiLA Web Service Documentation Extensions.....	17
2.2	SiLA Service Consumer.....	21
2.2.1	SiLA Event Receiver.....	21
2.2.2	Configuration.....	26
3	Interaction between Service Provider and Consumer.....	27
3.1	Device Identification.....	27
3.2	Device Reservation.....	27
3.3	Mandatory Commands.....	28
3.4	Communication Pattern.....	29
3.4.1	Synchronous Command Invocation.....	29
3.4.2	Asynchronous Command Invocation.....	30
3.4.3	Error Handling.....	33
3.5	State Machine.....	35
3.5.1	States.....	40
3.5.2	Sub-States.....	41
3.5.3	State Machine Events.....	42
4	Security.....	43
4.1	SiLA and HTTPS.....	43
4.1.1	Creation of Certificates.....	43
4.1.2	Usage of Certificates.....	44
5	SiLA Integration.....	45
6	Comments.....	47
6.1	Web Service Discovery.....	47
6.1.1	Get the IP Address.....	48
6.1.2	Build the URI.....	48
6.1.3	Discover Functionality.....	48
6.2	Performance.....	48
6.2.1	Transmission Time.....	49
6.2.2	Memory usage.....	49
6.2.3	CPU Load.....	49
7	References.....	50

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 6 of 65

8	Appendix.....	51
8.1	Mandatory Command description.....	51
8.1.1	Abort	52
8.1.2	DoContinue	53
8.1.3	GetDeviceIdentification	53
8.1.4	GetStatus	54
8.1.5	Initialize	56
8.1.6	LockDevice	57
8.1.7	Pause.....	59
8.1.8	Reset	60
8.1.9	UnlockDevice	61
8.2	Event Command Description	62
8.2.1	DataEvent	62
8.2.2	ErrorEvent.....	62
8.2.3	ResponseEvent.....	63
8.2.4	StatusEvent.....	64
8.3	Common Return Codes	65

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 7 of 65

1 Scope of Document

ID:SI 2; Rev: 15

This document describes the interface between a PMS and devices compliant with the SiLA standard. Apart from event data that shall be transmitted, the interface consists of Ethernet as hardware, Web Services as service infrastructure, and a behavior described by a state machine. The choice of Web Services as service infrastructure is the technical implementation layer.

Section 2 introduces the communication architecture. Section 3 explains the interaction between the components. It contains typical communication sequences and the state machine of a SiLA Device. Section 4 describes security aspects. Section 5 explains the SiLA integration levels illustrated by some examples. Section 6 contains some facts which are not normative but are important for understanding this specification. Mandatory commands and common return codes can be found in section 8. A list of device classes is available in the SiLA GIS Appendix [GISAPP] document.

1.1 Notational conventions

ID:SI 3; Rev: 12

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

References to documents, listed in section 7, are enclosed in square brackets like [XSD].

SiLA defined terms will be marked by *cursive letters*. They are defined in the global SiLA Glossary [**Error! Reference source not found.**].

Figures are shown mostly as UML diagrams. Because they should be understandable without deep UML knowledge the diagrams may be incomplete in some details.

Any message that is supposed to be human readable has to be available in the American English language, which is the default. A manufacturer, who wants to offer localization, has to use the command that is described in the SiLA Command Dictionary.

2 Architecture

ID: SP 56; Rev: 7+

The interface between the SiLA Devices and a PMS shall be realized using Web Services. HTTP shall be used as transfer protocol, HTTPS for enhanced security requirements, respectively. It is advised to use HTTP compression with gzip. SiLA supports only UTF-8 encoding in the payload (platform specific needs like usage of BOM, should be taken care of).

ID:SI 4; Rev: 15

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 8 of 65

SiLA Devices shall provide their functionality to the PMS by means of Web Services. Therefore devices shall behave as “*SiLA Service Provider*” and the PMS as “*SiLA Service Consumer*”. The “*SiLA Service Provider Interface*” is made up by all the commands implemented by the *SiLA Service Provider*.

The *SiLA Service Provider* offers two commands in “synchronous” manner, namely the GetStatus and the GetDeviceIdentification command. All other commands are offered in “asynchronous” manner.

In case of the “synchronous” commands GetStatus and GetDeviceIdentification, the *SiLA Service Consumer* waits for the answer of the *SiLA Service Provider*, which will only be sent when the command is finished.

In the asynchronous case this means that the *SiLA Service Provider* sends an immediate reply to the *SiLA Service Consumer* indicating that it is working on the command. After completion of the command execution, it sends an event to the *SiLA Service Consumer*. Therefore the *SiLA Service Consumer* must implement a so called *SiLA Event Receiver*. Additional events for data transfer, error messages and status messages are described in more detail in section 2.2.1.

Figure 1 shows the relations between the modules mentioned.

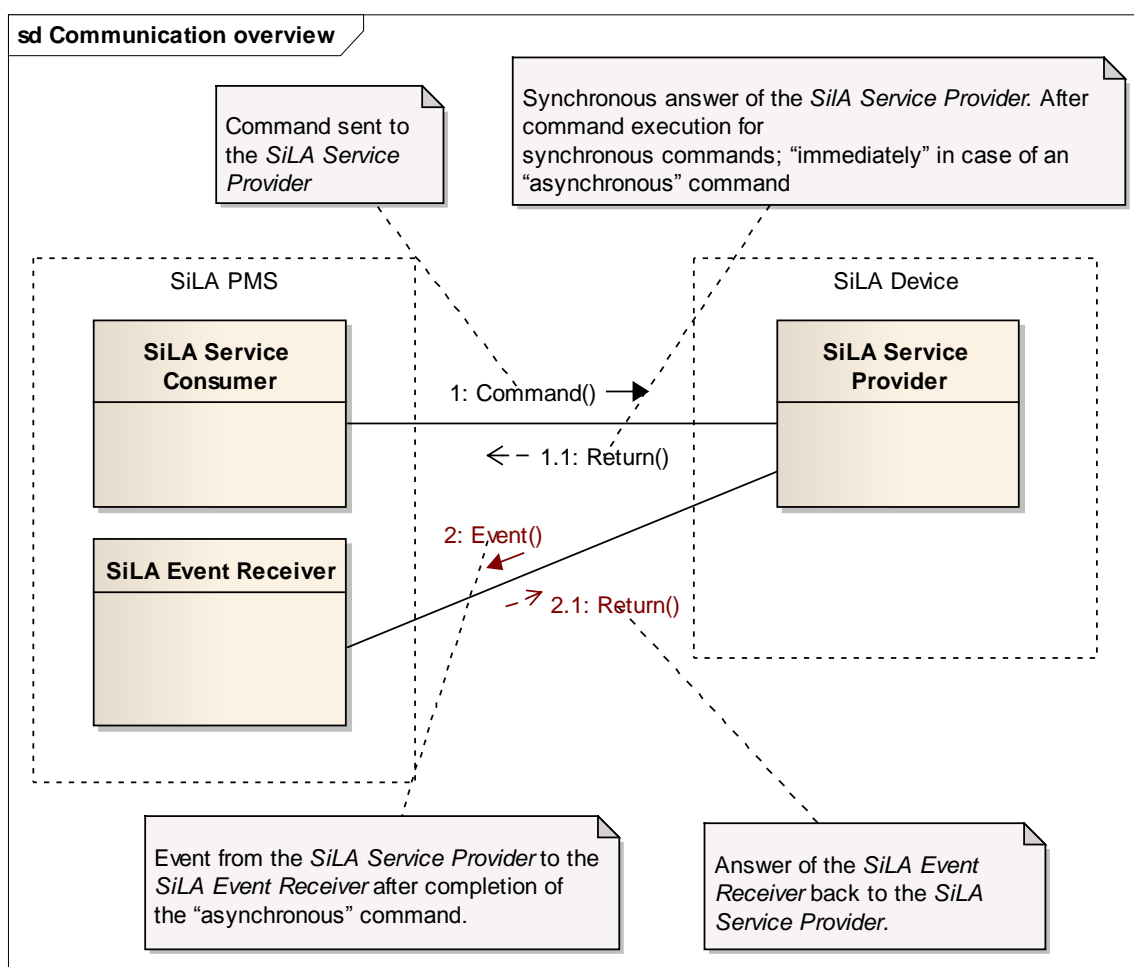



Figure 1: Interaction of the main modules

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 9 of 65

The PMS and all controlled SiLA Devices comprise the so called *SiLA System*.

2.1 *SiLA Service Provider*

ID:SI 5; Rev: 7

All entities, which provide the functionality of a defined Device Class [GISAPP], are called *SiLA Service Providers*. Primarily these entities are the laboratory devices.

The functionality will be provided by the *SiLA Service Provider Interface*. It consists of commands, return values, and events, which are described in the following sections.

2.1.1 **Commands**

ID:SI 6; Rev: 19

This section describes the rules for commands sent to the *SiLA Service Provider*.

The **Mandatory Commands** **MUST** be implemented by all *SiLA Service Providers* in all device classes. They are used to enable the transitions in the state machine and for some additional purposes e.g. status information.

The **Required Commands** describe commands that belong to a device class. They **MUST** be implemented for all *SiLA Service Providers* belonging to this device class (the same commands can be required in different device classes).

The **Optional Commands** express commands that **MAY** be implemented by *SiLA Service Providers* in the device class.

Additional features, specific to a particular device, **SHALL** be implemented as **Specific Commands**.

The **Common Command Set** of a device class is made up of the mandatory, the required and the optional commands. Exchangeability of two devices is only clear-cut if also the same optional commands are implemented in both devices.

The *Common Command Set* and the *Specific Commands* are implemented by Web Service methods. There are no additional web service methods allowed at the same URI. The distinction between these two kinds of commands is given by an attribute of the SiLA Web Service documentation extensions. A more detailed description can be found in section 2.1.7.

Every command name **MUST** only use characters, which are allowed for XML-Names [XSD].

The rules defined in this document **MUST** be fulfilled by any command of the *Common Command Set* and by any *Specific Command*.

Results that are available after executing a command **MUST** be returned by a common return value, which is described in more detail in section 2.1.1.3. Additional data **MAY** be

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 10 of 65

transferred by Events (see section 2.1.2). An asynchronous command MUST only use input parameters.

2.1.1.1 Command Parameters

ID: SP 58; Rev: 22

SiLA Command Parameters are implemented by the parameters of the Web Service methods.

Every parameter name MUST only use characters which are legal in XML-Names [XSD]. Terms potentially colliding with reserved words in the programming languages C, C#, C++, Java, and VBA MUST NOT be used as parameter names. Additionally the guidelines [Guide] specify more concrete information about names and units of parameters.

All parameters have the direction <In>, except some parameters of the (synchronous) commands GetStatus and GetDeviceIdentification. Out parameters must be included in the responseData parameter of the ResponseEvent (see chapter 2.2.1.3)

There MAY be command parameters, which can be assigned a NULL value (e.g. “null” in C# or “Nothing” in Visual Basic) at function invocation on client side. How this exactly works depends on the programming language of the client's implementation. In this case the parameters are not included in the SOAP message. The receiver of the SOAP message MUST handle missing parameters in a SOAP message as NULL assigned parameters and MUST NOT generate an error.

Parameters falling under this category MUST be specified in the command definition.

2.1.1.2 Data Types

All data types are defined according to [XSD] and to [SOAP11]. Therefore the allowed data types for parameters are a subset of the types defined in [XSD].

Every parameter MUST have one of the following types

Table 1 Allowed data types

Type name	Description
Long	64 bit signed integer value
Int	32 bit signed integer value
Short	16 bit signed integer value
Byte	8 bit signed integer value
unsignedLong	64 bit unsigned integer value
unsignedInt	32 bit unsigned integer value
unsignedShort	16 bit unsigned integer value
unsignedByte	8 bit unsigned integer value
Float	IEEE single-precision 32 bit floating point value
Double	IEEE double-precision 64 bit floating point value
Boolean	Binary-value logic. Possible values: {true,false,1,0}
String	Character string. Only legal XML-characters are allowed
Enumeration	A set of distinct names.
dateTime	The SOAPs simple type dateTime
Duration	Duration is a string that represents a time span or time interval. It has to be

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 11 of 65

	formatted according to ISO 8601. (see description below)
array	Array of any simple types above. Also array of array.
xmlDocument	Complex types are generally represented as xmlDocuments. Their schema is given in the corresponding sections. Within the SOAP message it is represented as a String.
Complex type	For some mandatory commands (e.g. GetStatus, StatusEvent) special complex types are defined (see corresponding definitions)

Binary data has to be coded by Base64.

The **Duration** type is of type string. Its value is coded according to the ISO8601 standard and therefore has the format P<date>T<time> where:

P stands for period or time-interval

<date> = #Y#M#W#D # any positive number
 Y = year
 M = months
 W = weeks
 D = days

<time> = #H#M#S H = hours
 M = minutes
 S = seconds

The following rules apply:

- If the number is zero, then the letter is omitted
- T is omitted when H, M, S are zero (e.g. P17D)
- Fractions of seconds are allowed
- Numbers greater than the usual range are allowed (for example PT36H is equivalent to P1DT12H)
- Case IS sensitive

Examples:

P5DT4H12M17S = 5 days 4 hours 12 minutes and 17 seconds

PT3.75S = 3.75 seconds

P41D = 41 days

PT1800S = 1800 seconds = 30 minutes

For SiLA it is recommended not to use the year, the month, or the week tags, because they are not precise. However both Service Provider and Service Consumer have to be able to interpret all representations according to ISO 8601.

2.1.1.3 SiLA Return Value

ID: SP 57; Rev: 34

All responses to a command **MUST** have the form and content of a *SiLA Return Value*. This value is a complex type named SiLAReturnValue, which is composed of the following members.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 12 of 65


Table 2: Structure of SiLAReturnValue

Name	Type (see 2.1.1.2)	Possible values	Description
returnCode	Int	1, ... max	The return code itself.
message	String	No length restrictions.	The return message. In case an error occurs the message describes the error by a human readable text. While device class groups will be defining some common return codes, the detailed description of the error will be different from device to device. Therefore the message text should be helpful to the user of the device and contain enough information to enable a profound analysis of the error.
duration	Duration	no restrictions	<p>If a synchronous command was called, the “Duration” parameters reports the consumed time for command processing.</p> <p>For an asynchronous command the “Duration” parameter returns the estimated duration for processing the asynchronous command in the synchronous (immediate) answer. The PMS MUST use this value to set a timeout, which has not to be exceeded. For unknown duration, a 0 value shall be returned. In this case the PMS cannot compute a timeout value.</p> <p>In the response event of an asynchronous command the “Duration” parameter represents the consumed time for processing the command.</p> <p>For use in the simulation mode, in all three cases the estimated time for processing the command shall be returned.</p>
deviceClass	Int	0, ... max device class number	The ID of the Device Class (see document SiLA GIS Appendix [GISAPP])

Return codes defined in section 8.3 of this document and in the specifications for the Device Class MUST be used whenever possible. Predefined return codes of the Mandatory Commands are in the range 1 to 99. Predefined return codes for the Common Command Set use the range 100-999. All other return codes are 1000 or greater and MAY be defined by the device manufacturer.

Different Device Classes MAY use the same error code for different meanings.

Commands MUST NOT use SOAP fault elements. Any error MUST be returned by use of the *SiLA Return Value* in the synchronous response, the ResponseEvent or the ErrorEvent. SOAP Fault elements will only be used for returning errors by the hosting entity while accessing the web method (e.g. if the method was not found).

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 13 of 65

2.1.1.4 Execution of Commands

ID: SP 59; Rev: 26

All commands **MUST** provide an immediate answer. For the *GetStatus* and the *GetDeviceIdentification* command or if an error occurred within the command invocation, no event will be issued. Such errors include invalid *lockId*, device is not in the appropriate state for the next command and errors in the parameters including 'value out of range'. In all other cases the command execution is started and upon completion or the occurrence of an error, an event is raised by calling the appropriate web service method on the *SiLA EventReceiver*. Commands **MAY** be queued in the device but still the synchronous answer is "immediate". The returned estimated command execution duration value includes the processing time of the commands already in the queue. A queue length of 1 is allowed.

In the immediate answer of a command invocation the *SiLA Service Provider* **MUST** specify a timeout value. The *SiLA Service Consumer* **MUST** handle a missing asynchronous command response within this period as a timeout error.

The device's documentation shall contain information on the possibilities of concurrent use of commands, such as the information, which commands can be processed in parallel. If the device allows queuing then the maximal queue length must also be documented. This documentation is not to be included in the SiLA Web Service Documentation Extension (see 2.1.7).

Further instructions on how to process commands are available in section 3.5.

2.1.1.5 Common Parameters

ID: SP 60; Rev: 14+

All asynchronous commands of the *SiLA Service Provider Interface* **MUST** have the following parameters:

Table 3: Common parameters

Name	Type
<i>requestId</i>	Int
<i>lockId</i>	String

Where:


- **Type** is the XML-Schema type see 2.1.1.2

There are some exceptions in the mandatory commands, where the *lockId* is omitted.

2.1.1.6 The Parameter "requestId"

ID: SP 61; Rev: 9+

The value of the parameter "requestId" shall be generated by the *SiLA Service Consumer* for every command invocation. The *SiLA Service Provider* shall use it in the asynchronous answer and in the event description to identify the command call.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 14 of 65

The Service Consumer **MUST** ensure the uniqueness of this parameter for the runtime of a *protocol*.

It is possible to reuse the IDs as long as there is no other running command on that device with that ID.

The requestID has to be a positive number greater than 0.

2.1.1.7 The Parameter “lockId”

ID: SI 8; Rev: 3

The parameter “lockId” shall be used to realize the device reservation (see 3.2).

2.1.1.8 Optional Parameters

ID: SP 62; Rev: 4

No Optional Parameters are allowed. The *Common Command Set* definition should comprise the maximum number of parameters, thus enabling the omission of parameters by nulling them. This also means that “overloading” of commands is not possible.

2.1.2 Events

ID: SP 63; Rev: 7

The *SiLA Service Provider* shall deliver asynchronous information using SiLA Events. SiLA Events are Web Service methods, which are hosted by the *SiLA Event Receiver*.

A *SiLA Service Consumer* **MUST** register its *SiLA Event Receiver* at the *SiLA Service Provider*. This shall be done by passing the URI as value of a parameter of the Reset command or LockDevice command. This parameter **MUST** not be empty or omitted.


One *SiLA Event Receiver* is registered by the reset command. Additional registrations of other event receivers may be possible with Specific Commands defined in the SiLA Command Dictionary [CCD].

2.1.3 Error Handling

ID: SP 64; Rev: 8

If there is an error while processing a command the *SiLA Service Provider* **MAY** allow various error recovery routines. All possibilities shall be communicated within the ErrorEvent message (see 2.2.1.4) The *SiLA Service Consumer* **MAY** automatically decide, which option to select or it **MAY** enable the user to select the appropriate continuation task.

One of the continuation tasks **MUST** be marked as default task. The PMS **MAY** use this task automatically if a timeout occurred while waiting for the user action.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 15 of 65

2.1.4 Simulation

ID: SP 65; Rev: 6

Every *SiLA Service Provider* MUST implement a simulation mode. If the *SiLA Service Provider* works in this mode, every command shall be answered immediately with the return value containing the estimated duration. The *SiLA Service Provider* computes this duration by using the current parameter values. The response event shall also be provided for all commands, except for the synchronous commands *GetStatus* and *GetDeviceIdentification*.

The simulation mode shall be set by the command “Reset” and is valid until the next “Reset”.

2.1.5 Service Configuration

ID:SI 9; Rev: 7

The following specifications in this section shall ensure the interoperability between any *SiLA Service Provider* and any PMS. This is done with the goal of best interoperability rather than optimized performance.

ID: SP 66; Rev: 12

Every implementation of a *SiLA Service Provider* MUST fulfill the “Basic Profile Version 1.1” of the Webservice Interoperability Organization (hereafter “[Basic Profile]”).


This means especially the following regulations:

- HTTP or HTTPS MUST be chosen as transfer protocol for the Web Service.
- If using HTTPS, TLS 1.0 or SSL 3.0 MUST be used.
- Every HTTP request MUST use the HTTP POST method.
- Every *SiLA Service Provider* MUST deliver a service description by a WSDL Version 1.1 file.

Additionally to the requirements defined in Basic Profile the following SiLA specific requirements for the configuration are defined:

- Any Message MUST be declared as an IN-OUT Message.
- Any communication between the *SiLA Service Consumer* and the *SiLA Service Provider* MUST be a point-to-point communication. No intermediaries are allowed (for instance no webservice proxying is allowed).
- SOAP-Headers MUST NOT be used.
- SOAP-Attachments MUST NOT be used. The commands, which will provide access to a file, MUST transmit the content of the file by an array of base64 coded bytes.
- SOAP-Import MUST NOT be used (no `import` directive in the WSDL).
- The style/use combination MUST be Document/Literal MICROSOFT PROPOSAL
- Web Services are defined in the Namespace <http://sila-standard.org> (`xmlns:sila="http://sila-standard.org"`)
- SOAP-Action format MUST be <http://sila-standard.org/CommandName> (Reset example: `SOAPAction: "http://sila-standard.org/Reset"`)

To identify the *SiLA Service Provider* the vendor MUST deliver the MAC-Address and the Web Service- and WSDL URI, both without node name, with the *SiLA* device.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 16 of 65

2.1.6 Network Interface

ID: SP 67; Rev: 8

The communication of the Web Services has to take place using a special protocol stack and well defined wiring. Compliant devices of the SiLA integration level 2 and the SiLA integration level 3 (see Chapter 5) SHALL offer connectivity by one of the following standards:

- IEEE 802.3 10BASE-T
- IEEE 802.3u 100BASE-TX
- IEEE 802.3ab 1000BASE-T

These options specify Ethernet variants with the transmission ratio of 10Mbit to 1Gbit and define the wiring for connectivity to standard routers and switches. In the protocol stack, the network layer MUST support the Internet Protocol version 4 (IPv4). Additionally version 6 (IPv6) MAY be supported. The only protocol SiLA allows in the transport layer is the Transmission Control Protocol. The specification of an optional security layer for enhanced security is defined in chapter 4. The Web Service calls shall be transmitted over HTTP as SOAP messages.

ID: SP 68; Rev: 11

For a Service Provider the following regulations apply:

- There are no restrictions about the kind of hosting of the Web Service.
- The URIs of WSDL and service MUST be unique, such as a configurable URI.
- A device MAY support discovery. If it does so it MUST be done as described in section 2.1.6.1.

2.1.6.1 Web Service Discovery

A SiLA Service Provider MAY implement dynamic address resolution. If it implements dynamic address resolution, RFC 3927 (AutoIPv4) MUST be used and for devices that additionally support IPv6 RFC4862 (AutoIPv6) MUST also be used.

A SiLA Service Provider MAY implement Web Service Discovery. If it implements this feature it MUST use the WS-Discovery (OASIS-Standard: <http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html>)

Basically if a device supports discovery it has to check for a DHCP server for at least 10 seconds. If no DHCP server responds, the device has to assign an IP-Address itself automatically according to [RFC 3927] and [RFC 4862] respectively in order to be able to communicate on the local network. Additionally if web service discovery would be offered, it has to do it via SOAP messages as specified in the above mentioned [WS-Discovery]. In particular, important information that shall be included in WS-Discovery messages are the following SiLA device parameters:

- SiLA URI of the WSDL (e.g.: <http://192.168.1.2:8000/someDevice?wsdl>)
- SiLA Interface Version (e.g.: 1.3)
- SiLA Device Class (e.g. 1)
- SiLA Device Class Version, basically the Common Command Version (e.g. 0.94)

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 17 of 65

- SiLA SubDevice Class can optionally be included, see Section 3.1 for the conditions.
- SiLA Device Manufacturer string (e.g. manufacturerName)
- SiLA Device Name string (e.g. someName)
- SiLA Device Serial number (e.g. 12345678)
- SiLA Device Firmware Version (e.g. 0.7)

See Section 3.1 for more information.

2.1.7 SiLA Web Service Documentation Extensions

ID:SI 11; Rev: 21+

The SiLA device interface shall use the documentation tag of the Web Service method to include additional information about the method or its parameters. This documentation tag is part of the WSDL. The tag itself is structured as an XML document such as in Listing 1 and Listing 2.

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="PT30M17S"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-
standard.org/schemata/SoapAnnotation_1.2.xsd">
  <Summary>This command performs processing steps after a delay.</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    parameter is the identification of the PMS, which has locked the device.
  </Parameter>
  <Parameter name="delay" xsi:type="Duration">
    The time before the processing steps are performed.
  </Parameter>
  <Parameter name="processingSteps" xsi:type="complexParameter" typeName="processSteps">
    The processing steps. The type "processingSteps" have to be defined according to the
    SiLA Data Capture Specification.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="5">
    <Description>
      The first five process steps; each ParameterSet includes name and duration.
    </Description>
    <Value name="stepName" type="String"/>
    <Value name="stepDuration" type="Duration"/>
  </Response>
</SiLACommandDescription>
```

Listing 1: WSDL documentation tag example with details about command's response

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 18 of 65

```


<SiLACommandDescription isCommonCommand="true" estimatedDuration="PT30.7S"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://silastandard.org/schemata/SoapAnnotation_1.2.xsd">
  <Summary>This command performs a measurement at a specified speed.</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    parameter is the identification of the PMS, which has locked the device.
  </Parameter>
  <Parameter name="speed" unit="m/s">
    The speed of the measurement.
  </Parameter>
  <Response xsi:type="complexResponse">
    <Description>
      The measurement result. The type of the result has to be formatted according
      to the type named "complexMeasurement" (see SiLA Data Capture Specification).
    </Description>
    <TypeName>complexMeasurement</TypeName>
  </Response>
</SiLACommandDescription>

```

Listing 2: WSDL documentation tag example – describing the command’s response

It is used to:

- distinguish between the *Common Command Set* and *Specific Commands*,
- specify allowed ranges (min, max) of a parameter,
- specify a default value of a parameter, and
- specify a suggested timeout value in seconds. This timeout value MAY be used by the PMS to create an initial schedule.
- Specify the type of a parameter of type “Duration”. This is necessary because the used SOAP-Type of such a parameter is “String”.
- Specify the type of a parameter of type “xmlDocument”. This is necessary because the used SOAP-Type of such a parameter is “String”.
- Specify the type of data the *SiLA Service Provider* will send in the ResponseEvent and DataEvent. This is an xmlDocument containing either a sequence of one or more elements with simple types (“standardResponse”, see schema below) or a list of such sequences. In the latter case each sequence MUST contain the same elements. The attributes of the tag <Value> reference to the attributes of the tag <Parameter> of the schema in section 2.2.1.2. The use of ParameterSetCount allows to give additional information about the response parameters. The ParameterSetCount can be
 - omitted: This means there is no restriction; it is allowed to send (a) array lengths that vary with each method call, (b) a single value, (c) no value, (d) always the same array length
 - 0: This means a response event containing no values can be expected
 - 1: This means a single value or a single set of values can be expected
 - >1: This means that always the given count of sets will be returned.
- Alternatively the data of the *SiLA Service Provider* can be a more complex xmlDocument (“complexResponse”) with a name. This name can be mapped to the data type according to the SiLA Data Capture Specification [DCS] and the device specifications.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 19 of 65


The listed information is not necessary to call the command but it delivers additional information, which the *SiLA Service Consumer* MAY use to implement additional features, such as range check for parameters.

The information has to be provided in an XML-structure. The entire XML-structure is the web method documentation tag of the WSDL. It shall be included in the WSDL as String.

Every *SiLA Service Provider* MUST deliver the documentation tag according to the schema of Listing 3, which will be published online:

http://sila-standard.org/schemata/SoapAnnotation_1.2.xsd.

The documentation tag MUST NOT be omitted.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 20 of 65

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="SiLACommandDescription"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="SiLACommandDescription">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Summary" minOccurs="1"/>
        <xs:element name="Parameter" minOccurs="2" maxOccurs="unbounded"
          type="Parameter" />
        <xs:element name="Response" minOccurs="0" type="Response" />
      </xs:sequence>
      <xs:attribute name="isCommonCommand" type="xs:boolean" use="required"/>
      <xs:attribute name="estimatedDuration" type="xs:duration" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Parameter">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="minValue" type="xs:string"/>
        <xs:attribute name="maxValue" type="xs:string"/>
        <xs:attribute name="unit" type="xs:string"/>
        <xs:attribute name="defaultValue" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="Duration">
    <xs:simpleContent>
      <xs:extension base="Parameter" />
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="xmlDocument">
    <xs:simpleContent">
      <xs:extension base="Parameter" />
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="Value">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="type" type="AllowedType" use="required"/>
    <xs:attribute name="unit" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="Response" abstract="true">
    <xs:sequence>
      <xs:element name="Description" minOccurs="1" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="standardResponse">
    <xs:complexContent mixed="false">
      <xs:extension base="Response">
        <xs:sequence>
          <xs:element name="Value" minOccurs="0" maxOccurs="unbounded"
            type="Value" />
        </xs:sequence>
        <xs:attribute name="parameterSetCount" type="xs:int"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="complexResponse">
    <xs:complexContent mixed="false">
      <xs:extension base="Response">
        <xs:sequence>
          <xs:element name="TypeName" type="xs:string" minOccurs="1"/>
          <xs:element name="Version" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

```

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 21 of 65

```

<xs:simpleType name="AllowedType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Long"/>
    <xs:enumeration value="Int"/>
    <xs:enumeration value="Short"/>
    <xs:enumeration value="Byte"/>
    <xs:enumeration value="unsignedLong"/>
    <xs:enumeration value="unsignedInt"/>
    <xs:enumeration value="unsignedShort"/>
    <xs:enumeration value="unsignedByte"/>
    <xs:enumeration value="Float"/>
    <xs:enumeration value="Double"/>
    <xs:enumeration value="Boolean"/>
    <xs:enumeration value="String"/>
    <xs:enumeration value="dateTime"/>
    <xs:enumeration value="Duration"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Listing 3: XSD of the WSDL documentation tag

ID: SP 69; Rev: 10

Additionally to the schema of Listing 3, the following regulations apply:

- All parameters **MUST** have a parameter element in the structure with the range and unit attributes when applicable.
- Default values can also be included.

2.2 SiLA Service Consumer

ID:SI 12; Rev: 4

The *SiLA Service Consumer* shall use the commands described in the *Common Command Set* of a specific SiLA Device Class. Every *SiLA Service Consumer* **MUST** use the commands accordingly to section 2.1.1.

The *SiLA Service Consumer* **MAY** use the SiLA Web Service Documentation Extensions of the WSDL of the *SiLA Service Provider* to provide better service to the user.

2.2.1 SiLA Event Receiver

ID: SP 70; Rev: 9

A *SiLA Service Consumer* **MUST** implement a *SiLA Event Receiver*.

The *SiLA Event Receiver* contains one Web Service method for each kind of event. The following sections describe the available events.

The *SiLA Event Receiver* **MUST** be able to understand different forms of SOAP, currently the versions 1.1 and 1.2 as well as different representations of arrays [Basic Profile].

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 22 of 65

The allowed data formats for event parameters are the same as for commands of the *SiLA Service Provider Interface* (see section 2.1.1.1).

2.2.1.1 Status Event

ID:SI 13; Rev: 17

The Status Event shall notify the PMS asynchronously about status changes in the device. The *SiLA Service Provider* shall send this event only if the state change is not requested by a command. In the case of a requested status change, such as by the Initialize-Command, the new state depends on the return value.


An error, which is not related to a command, shall also be sent by a Status Event.

The eventDescription (see section 8.2.4) shall be structured according to the schema of Listing 4, which will be published online:

http://sila-standard.org/schemata/StatusEvent_1.3.xsd.

The fields have the following meanings:

- `DeviceId` is the ID that was transmitted with the Reset command.
- `DeviceURI` means the service URI.
- `DeviceState` is the main state of the state machine
- `StateChangeTime` is the time, since when the state machine is in the state that is reported by the `DeviceState` field.
- `StatusMessage` can be used for human readable information about the occurrence of this status event.
- `FaultCorrectionHints` is a human readable advice for the operator about how to fix the situation.
- `CurrentValue` can contain a current value of a variable.
- `Extension` can be used to provide some context information.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 23 of 65

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="StatusEvent" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="StatusEvent">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DeviceId" type="xs:string"/>
        <xs:element name="DeviceURI" type="xs:string"/>
        <xs:element name="DeviceState" type="Status"/>
        <xs:element name="StateChangeTime" type="xs:dateTime"/>
        <xs:element name="StatusMessage" type="xs:string"/>
        <xs:element name="FaultCorrectionHints" type="xs:string" minOccurs="0"/>
        <xs:element name="CurrentValue" type="xs:string" minOccurs="0"/>
        <xs:element name="Extension" type="xs:string" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="startup"/>
      <xs:enumeration value="resetting"/>
      <xs:enumeration value="standby"/>
      <xs:enumeration value="initializing"/>
      <xs:enumeration value="idle"/>
      <xs:enumeration value="paused"/>
      <xs:enumeration value="inError"/>
      <xs:enumeration value="busy"/>
      <xs:enumeration value="errorHandling"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>

```

Listing 4: XSD of the Event Description

2.2.1.2 Data Event

ID:SI 14; Rev: 20


A command delivers requested data by its Response Event. In some cases, such as kinetic measurements, it is necessary to transmit intermediate data while the command is processing.

In this case the Data Event shall notify the *SiLA Service Consumer* asynchronously about new available data. The data MAY be included in the event message or alternatively a link to the data store MAY be provided.

All data shall be transmitted by Data Events with an XML document as parameter. The requestId parameter is used to identify the command that requested the data. The XML will be structured according to the schema of Listing 5 or to the SiLA Data Capture Specification [DCS].

The schema will be published online:

http://sila-standard.org/schemata/ResponseType_1.2.xsd.


	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 24 of 65

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ResponseData" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Value" nillable="true" type="Parameter"/>
  <xs:complexType name="Parameter">
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="1">
        <xs:element name="Boolean" type="xs:boolean"/>
        <xs:element name="DateTime" type="xs:dateTime"/>
        <xs:element name="Float32" type="xs:float"/>
        <xs:element name="Float64" type="xs:double"/>
        <xs:element name="Byte" type="xs:byte"/>
        <xs:element name="Short" type="xs:short"/>
        <xs:element name="Duration" type="xs:duration"/>
        <xs:element name="unsignedByte" type="xs:unsignedByte"/>
        <xs:element name="unsignedShort" type="xs:unsignedShort"/>
        <xs:element name="unsignedInt" type="xs:unsignedInt"/>
        <xs:element name="unsignedLong" type="xs:unsignedLong"/>
        <xs:element name="Int32" type="xs:int"/>
        <xs:element name="Int64" type="xs:long"/>
        <xs:element name="String" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="parameterType" type="AllowedType"/>
  </xs:complexType>
  <xs:element name="ParameterSet" nillable="true" type="ParameterSet"/>
  <xs:complexType name="ParameterSet">
    <xs:sequence>
      <xs:element name="Parameter" type="Parameter"
        minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ResponseData" nillable="true" type="ResponseData"/>
  <xs:complexType name="ResponseData">
    <xs:sequence>
      <xs:choice>
        <xs:element name="AnyData" type="xs:anyType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="ParameterSet" type="ParameterSet"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Parameter" type="Parameter" minOccurs="0"
          maxOccurs="1"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="AllowedType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Boolean"/>
      <xs:enumeration value="DateTime"/>
      <xs:enumeration value="Float32"/>
      <xs:enumeration value="Float64"/>
      <xs:enumeration value="Byte"/>
      <xs:enumeration value="Short"/>
      <xs:enumeration value="Int32"/>
      <xs:enumeration value="Int64"/>
      <xs:enumeration value="unsignedByte"/>
      <xs:enumeration value="unsignedShort"/>
      <xs:enumeration value="unsignedInt"/>
      <xs:enumeration value="unsignedLong"/>
      <xs:enumeration value="String"/>
      <xs:enumeration value="Duration"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>

```

Listing 5: XSD of Data Event

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 25 of 65

There are three possible ways to include data according to the above schema:

- Single values are transmitted with the AnIML compatible fragment <Parameter>.
- Sets of parameters use the AnIML compatible fragment <ParameterSet> with its sub tags <Parameter>.
- For special purposes SiLA device class specifications MAY define an XSD-Schema to represent the specific data by use of the tag <AnyData>.

The command definitions shall specify which parameters **MUST** be transmitted in the XML structure and also define the attribute names. Additionally, the WSDL documentation tag listed in the specification document of the device, **MUST** report which parameters and data minimally must be transmitted by the data event (response event).

2.2.1.3 Response Event

ID:SI 15; Rev: 17

SiLA shall support commands with an asynchronous response (see 2.1.1.4). The event to transmit the asynchronous response from the *SiLA Service Provider* to the *SiLA Event Receiver* **MUST** always contain the command identifier (`requestId`) belonging to the original command and the response itself.

The Response Event shall provide an XML based output Parameter to report the data generated during the command execution. Again the format of the reported XML structure shall be structured according to listing 5. The SiLA Data Capture Specification [DCS] gives more information about the referring schema. The command definitions shall specify which parameters **MUST** be transmitted in the XML structure and also define the tag names.

2.2.1.4 Error Event


ID:SI 16; Rev: 24

With the Error Event the *SiLA Service Provider* shall notify the PMS about an error situation while executing a command. It shall use the *SiLA Return Value* to report the error code and the error message.

Within the Error Event, the *SiLA Service Provider* shall deliver a list of allowed continuation tasks according to the schema of Listing 6, which will be published online: <http://silastandard.org/schemata/ContinuationTask.xsd>. For every continuation task this list shall contain

- a task identifier (TaskType) that **MUST** be reproducible (no continuous numbering),
- a human readable message describing the possible continuation modes (TaskText),
- a tag specifying the default continuation task (DefaultTask),
- a user interaction timeout (UserInteractionTimeout),
- and the information whether a user interaction is required or the default continuation task shall be executed after the timeout period. (AutomaticAfterTimeout)

By additional tags the *SiLA Service Provider* MAY deliver command depending data for use with any of the tasks. It MAY also provide default values for this data. The PMS MAY modify these values automatically or MAY allow the user to modify these values in the error handling procedure.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 26 of 65

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ContinuationTask" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ContinuationTask">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ErrorMessage" type="xs:string"/>
        <xs:element name="DefaultTask">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TaskType" type="xs:string"/>
              <xs:element name="TaskText" type="xs:string"/>
              <xs:element name="UserInteractionTimeout" type="xs:duration"/>
              <xs:element name="Extension" type="xs:string"
                minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="AutomaticAfterTimeout" type="xs:boolean" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="Task" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TaskType" type="xs:string"/>
              <xs:element name="TaskText" type="xs:string"/>
              <xs:element name="Extension" type="xs:anyType"
                minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="SelectedTask" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TaskType" type="xs:string"/>
              <xs:element name="Extension" type="xs:string"
                minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Extension" type="xs:string"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Listing 6: XSD of the Continuation Task

The *SiLA Event Receiver* shall report the selected continuation task back to the *SiLA Service Provider* within the SOAP response of the *ErrorEvent*.

The Error Event is specified in detail in section 8.2.2.

2.2.2 Configuration

ID: SP 71; Rev: 3

Every *SiLA Event Receiver* MUST be a Web Service implementation with the same regulations as for the configuration of the *SiLA Service Provider* (see 2.1.5).

The Web Service of the *SiLA Service Consumer* MUST NOT work in application scope like the Web Service of the *SiLA Service Provider*. It MUST be able to receive Events at any time. It MUST NOT be blocked for a longer time.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 27 of 65

3 Interaction between Service Provider and Consumer

3.1 Device Identification

ID: SP 72; Rev: 15

Every *SiLA Service Provider* MUST implement the *Mandatory Command* GetDeviceIdentification. This command shall deliver information by a SiLA defined complex data type, named SiLA_DeviceIdentification, which shall be structured according to table 4

Table 4: Structure of the SiLA_DeviceIdentification

Name	Type (2.1.1.2)	Possible values	Description
WSDL	String	No length restrictions	The WSDL URI of the service provider
SiLAInterfaceVersion	String	No length restrictions	The version of the SiLA General Interface Specification.
SiLADeviceClass	Int	No length restrictions	The SiLA device class
SiLADeviceClassVersion	String	No length restrictions	The version of the Common Command Set.
SiLASubDeviceClass	Array of Int	No length restrictions	If the SiLADeviceClass is 1000 or more, this property will list the Device Classes out of which this device is composed. If the SiLADeviceClass is lower than 1000, this property MUST be omitted.
DeviceManufacturer	String	No length restrictions	Company name of the device manufactur. It is recommended to use the URI of the manufacturer's homepage for this entry.
DeviceName	String	No length restrictions	The model name of the device (vendor specific)
DeviceSerialNumber	String	No length restrictions	The serial number of the device (vendor specific)
DeviceFirmwareVersion	String	No length restrictions	The version of the firmware (vendor specific)

Every *SiLA Service Provider* MUST allow calling this command in all states. It MUST therefore be implemented synchronous and the DeviceIdentification will be included in the SOAP response.

3.2 Device Reservation

ID: SP 73; Rev: 22

For locking a device for exclusive use the Service Consumer shall invoke a Lock-command with a lockId as parameter. The locked Service Provider shall accept commands exclusively from the Service Consumer, which uses the same lockId in the command.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 28 of 65

The device locking mechanism does not prevent that a malicious code / system may lock a device for its exclusive use and the PMS will not be able to control the device. But it ensures that such a situation will be detected and no unrecognized work can be done.

Locking a device shall be done by the *Mandatory Command* LockDevice (see 8.1.6). Every *SiLA Service Provider* MUST implement this command.

The calling *SiLA Service Consumer* shall be identified by a unique String (`lockId`) generated by the caller itself. There shall be no restrictions about the String format.

The Service Provider shall store this `lockId` and answer only calls, which provide this `lockId`.

Unlocking a device shall be done by the *Common Command* UnlockDevice. Every *SiLA Service Provider* MUST implement this command.

Only the *SiLA Service Consumer*, which has locked a device is able to unlock the device. Therefore it uses the same `lockId` as used during locking the device.

Every *SiLA Service Provider* MUST unlock a device implicitly following a local reset of the device. The local reset can also be done by a device specific procedure, such as power off and power on.

3.3 Mandatory Commands

ID: SP 75; Rev: 10

Every device MUST implement a set of *Mandatory Commands*. These commands are used to initialize and reset the devices, to control the access to the device, and to obtain information about the device.

These commands are listed below with a short description for understanding the next sections. A detailed description can be found in section 8.1.

Every *SiLA Service Provider* MUST be implemented as state machine. Additionally corresponding sub-state machines might be necessary. Details are described in section 3.5.

ID: SP 76; Rev: 20

Tables 5 and 6 list the mandatory commands of the *SiLA Service Provider* and the *SiLA Service Consumer (SiLA Event Receiver)* respectively. The commands' details such as parameters are described in section 8.1 and 8.2.

Table 5: SiLA Service Provider's Mandatory Commands

Command name	Description
Abort	Aborts all currently running asynchronous commands.
DoContinue	Brings the device back into the normal state after a Pause.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 29 of 65

Command name	Description
GetDeviceIdentification	Returns information about the <i>SiLA Service Provider</i> .
GetStatus	By calling this command any Web Service client can receive information about the current state of the <i>SiLA Service Provider</i> at any time.
Initialize	Command used to initialize the device. After initialization the device can accept all its common and specific commands.
LockDevice	Locks the device for the sending <i>SiLA Service Consumer</i> .
Pause	Brings the device into a safe state. It is typically used for error handling.
Reset	Resets the device and transmits some basic parameters of the PMS to the device.
UnlockDevice	Unlocks the device.

ID: SP 77; Rev: 14

Every *SiLA Event Receiver* MUST implement the following *Mandatory Commands*.

Table 6: SiLA Service Consumer's Mandatory Commands

Command name	Description
DataEvent	By using this command the <i>SiLA Service Provider</i> can transmit data asynchronously to the <i>SiLA Event Receiver</i> .
ErrorEvent	By using this command the <i>SiLA Service Provider</i> signals an error and delivers information for the error handling.
ResponseEvent	By using this command the <i>SiLA Service Provider</i> signals the completion of the asynchronous command to the <i>SiLA Event Receiver</i> .
StatusEvent	Reports changes, which are not controlled by the state machine.

3.4 Communication Pattern

ID:SI 17; Rev: 3

In the following sections some typical communication sequences are shown. The diagrams contain just a subset focusing on the important command parameters.

3.4.1 Synchronous Command Invocation

ID:SI 18; Rev: 12

Figure 2 shows the synchronous command invocation for the GetStatus command.

SiLA Rapid Integration® Standardisation in Lab Automation	SiLA Device Control & Data Interface Specification	Specification Version: 1.3.08 Page 30 of 65
---	---	---

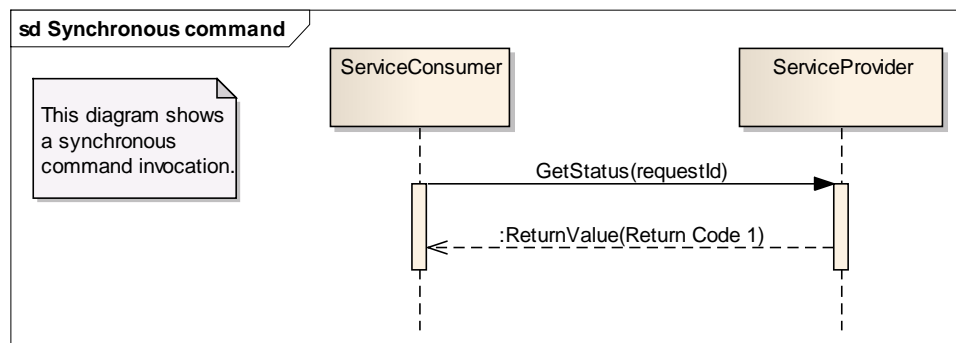


Figure 2: Communication sequence for a synchronous command

3.4.2 Asynchronous Command Invocation

ID:SI 19; Rev: 15

Figure 3 shows an asynchronous command invocation as a part of a larger communication sequence which is not included in the diagram. In this case there is no intermediate data, which would have to be transmitted before the command finishes.

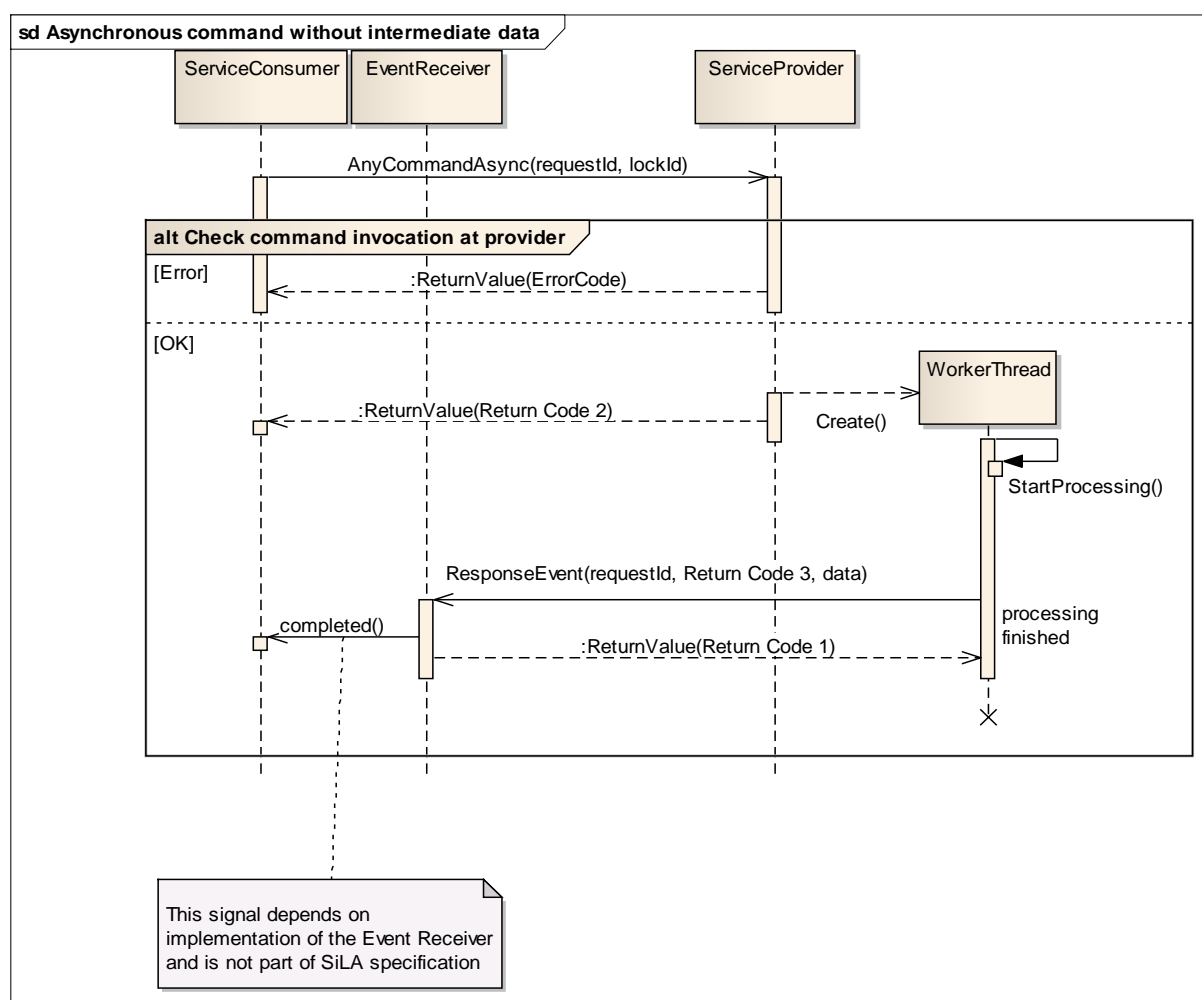


Figure 3: Asynchronous command

SiLA Rapid Integration® Standardisation in Lab Automation	SiLA Device Control & Data Interface Specification	Specification Version: 1.3.08 Page 31 of 65
---	---	---

ID:SI 20; Rev: 13

Figure 4 shows an asynchronous command invocation with one or more data events during a device's task. It shows also the case of an error in the command invocation.

If there is only one data item to be transmitted it MAY be done as described in Figure 3.

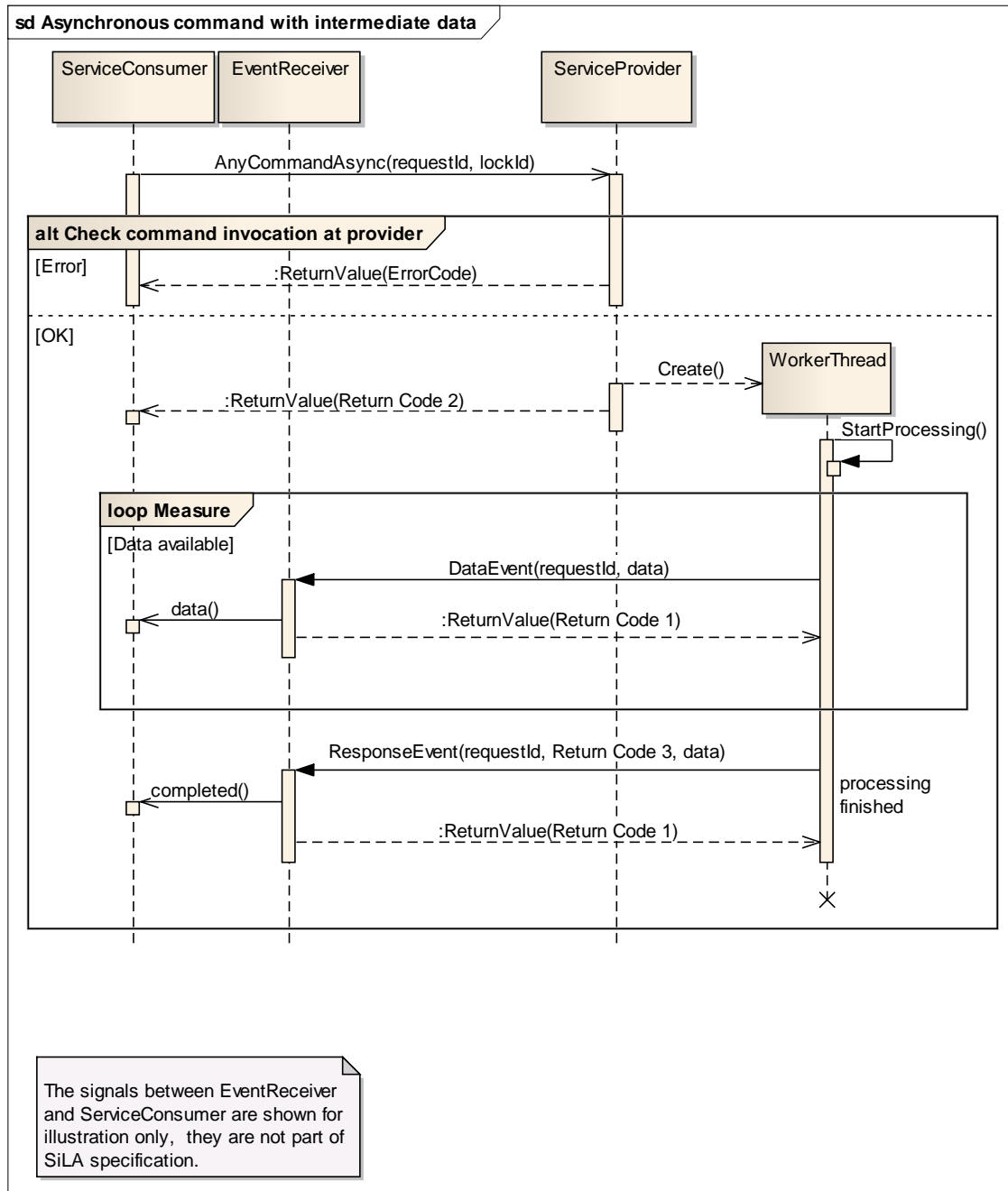


Figure 4: Asynchronous command with intermediate data

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 32 of 65

Figure 5 shows a typical communication sequence between a *SiLA Service Consumer* and one *SiLA Service Provider*. All command invocations are asynchronous.

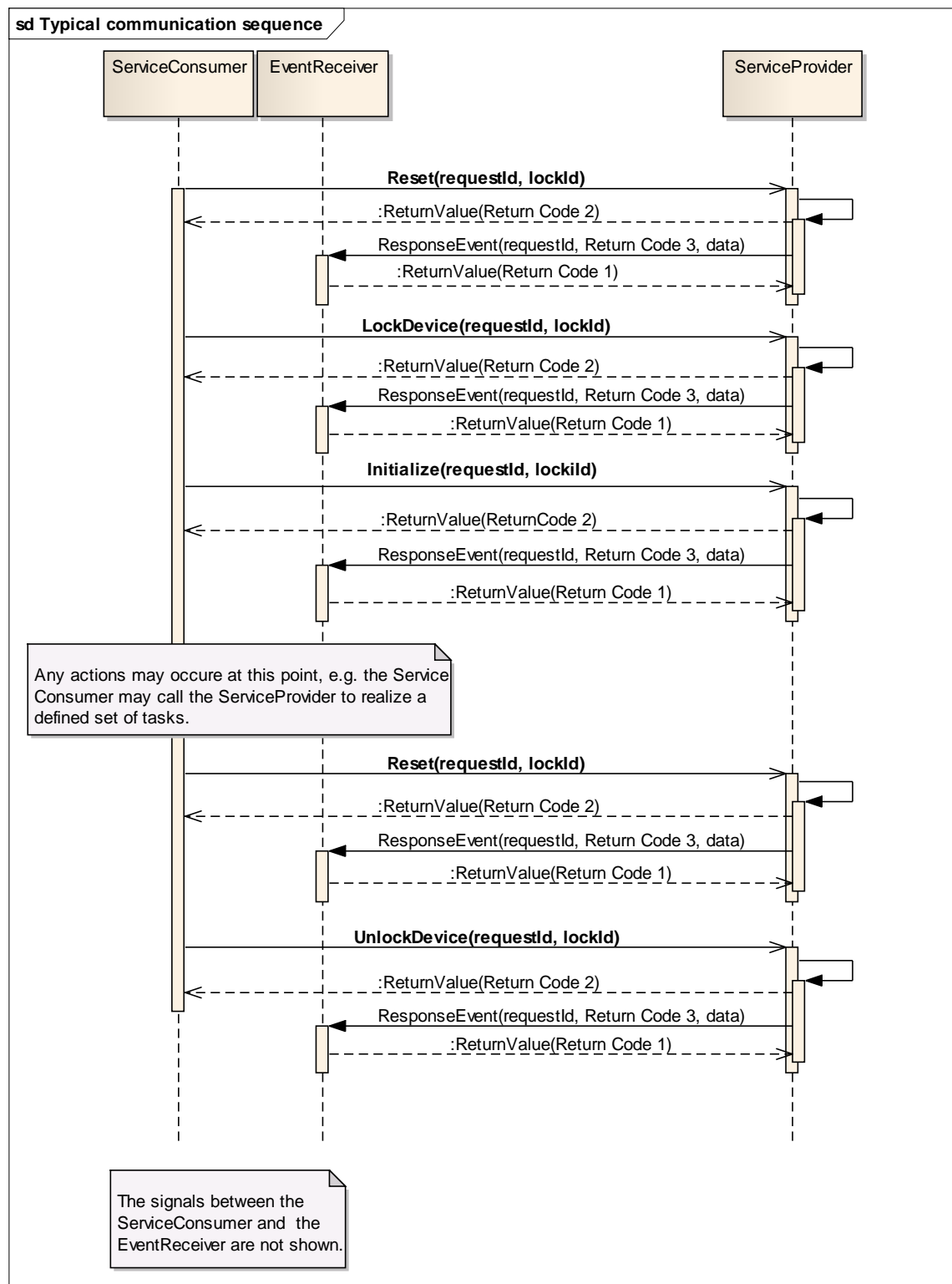



Figure 5: Typical communication sequence

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 33 of 65

3.4.3 Error Handling

ID:SI 21; Rev: 16

Figure 6 shows the interaction for the case that there is an error while processing an asynchronous command by the *SiLA Service Provider*.

For better readability of the diagram the WorkerThread is not shown. However it is the same logic as described in Figure 3 and Figure 4.

First the invocation of an asynchronous command occurs (AnyCommandAsync). While processing, the *SiLA Service Provider* detects an error and signals it by sending an ErrorEvent.

The PMS checks the continuation task. If the chosen continuation task requires a user interaction, the PMS sends a Pause command to all other *SiLA Service Providers* to set them into a paused state. As soon as the PMS receives all responses, it sends the continuation task with the response to the *SiLA Service Provider*.

The *SiLA Service Provider* sends a ResponseEvent as command execution result. Then the PMS will bring all paused *SiLA Service Providers* into normal operation state by sending the DoContinue commands.

SiLA Rapid Integration® Standardisation in Lab Automation	SiLA Device Control & Data Interface Specification	Specification Version: 1.3.08 Page 34 of 65
---	---	---

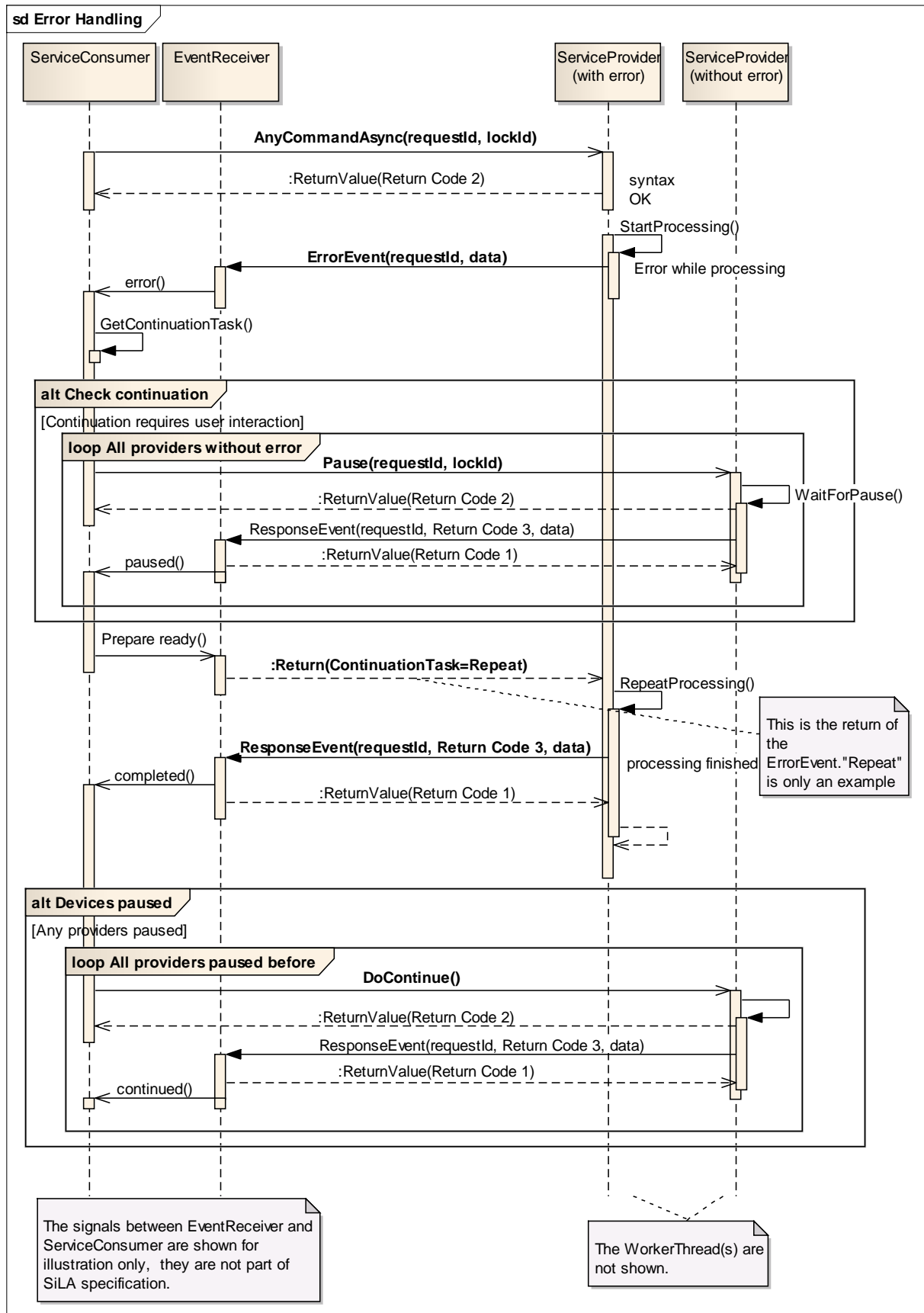



Figure 6: Error handling

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 35 of 65

If a new error occurs while repeating the command, the procedure is identical except execution of the Pause commands.

ID: SP 78; Rev: 6

If more than one *SiLA Service Provider* detects an error, such / these errors **MUST** be processed sequentially by the PMS.

3.5 State Machine

ID: SP 79; Rev: 32

The state machine describes the possible states and the triggers that enable the transitions between the states. Devices may show different levels of complexity. A basic device may only allow processing of one command after the other (except for some of the mandatory commands mentioned above) and therefore also not allow queuing. More advanced devices may then allow queuing of commands but still execute them sequentially. Even more complex devices, for example a pipetting robot with gripper, may allow parallel processing and can act on multiple commands concurrently. Again such a device may in addition also provide queuing. In the latter devices the *SiLA Service Consumer* is interested to know in which state every issued command currently is and therefore the device has to manage a separate substate machine for every command. So the *GetStatus* command **MUST** be able to report a substate for each executing substate machine (see also command definition).

In addition to the state machine diagrams, the following rules **MUST** be followed:

- All data describing the device state, **MUST** be kept from one command call to the next for the lifetime of the Web Service. If the Web Service is restarted the data **MAY** be lost.
- The Mandatory Commands *Reset*, *GetStatus* and *GetDeviceIdentification*, which are described in chapter 8, **MUST** be processed immediately according to the state machine described in this section.
- *Reset* may only be invoked once concurrently. A second *Reset* will be rejected with return code 9.
- All other mandatory commands are executed sequentially and only when they are allowed according to the state diagram. They are not queued. Therefore they are refused while another of these mandatory commands is executing. For example a started *Pause* command cannot be followed by an *Abort* command.
- *Abort* halts all executing asynchronous commands. Aborted commands cannot be continued.
- Paused commands **MUST** be enabled to execute to completion after the command *DoContinue*.
- Use the *Reset* command as a “software” emergency stop.
- A queue with a length of 1 (in case of no parallelism) implies sequential execution of commands. This means the next command from the Common Command Set or a Specific Command will only be accepted if the previous command is finished and the device is in the idle state.
- If a command has to be queued, the estimated duration is calculated based on the total estimated duration of all queued commands. This includes the remaining execution duration of the currently processing command.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 36 of 65

- Syntax (e.g. Duration) and Ranges need to be checked prior to start a command. In case of an error, the command has to be rejected (only depicted in busy state machine) with return code 11.

If there are some asynchronous answers pending and the *SiLA Service Provider* is not able to process a command additionally received, it **MUST** return the defined return code see 8.3).

Figure 7 depicts the basic state machine for the *SiLA Service Provider*. The busy state will be detailed in the following figures. Substates are used to describe the busy state in more detail. All other states do not use sub-states and the value of the substate parameter is null. In the next sections the states of the state machine and the triggers are described.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 37 of 65

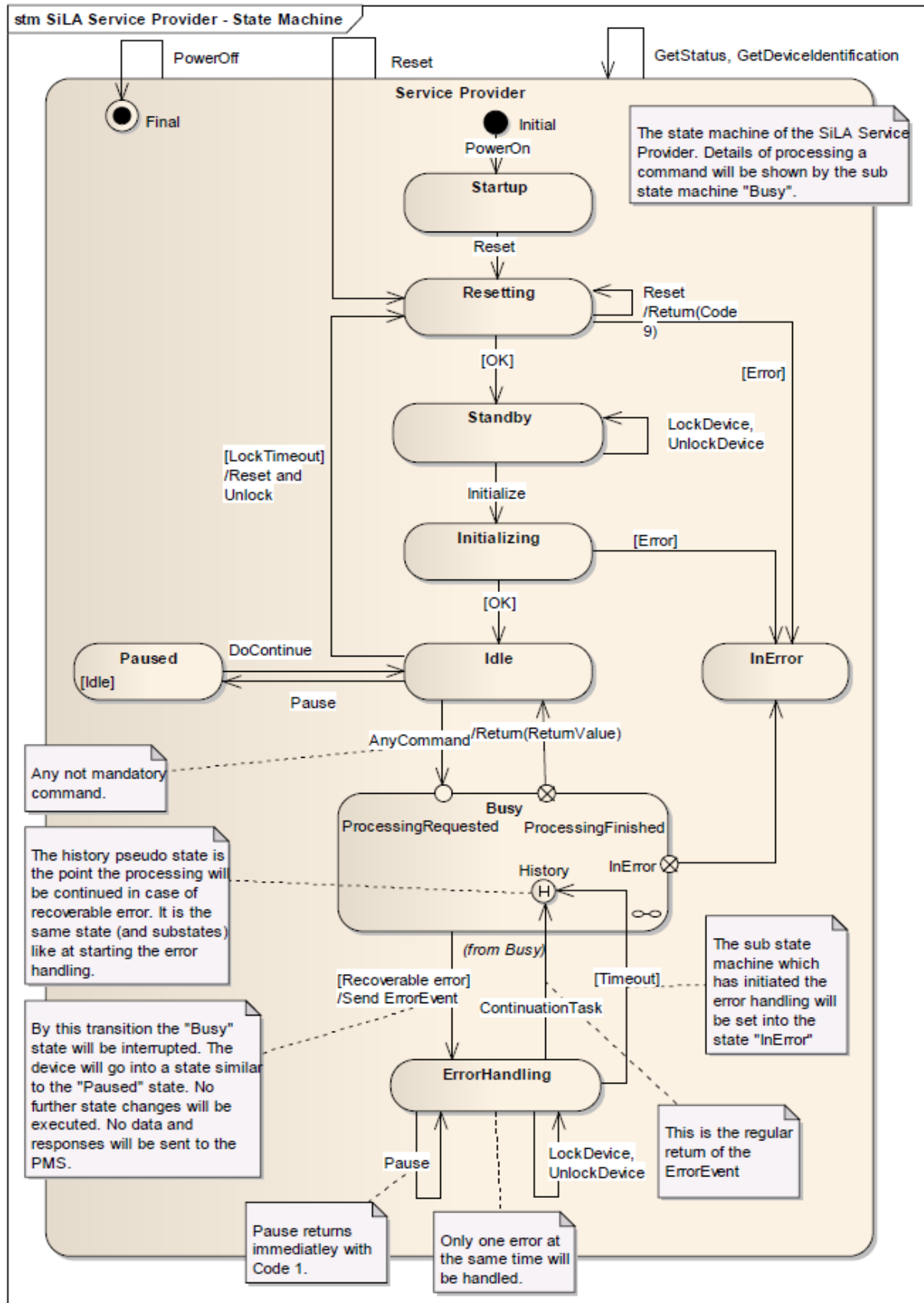


Figure 7: State Machine of SiLA Service Provider

Figure 8 details the busy state of the state machine.

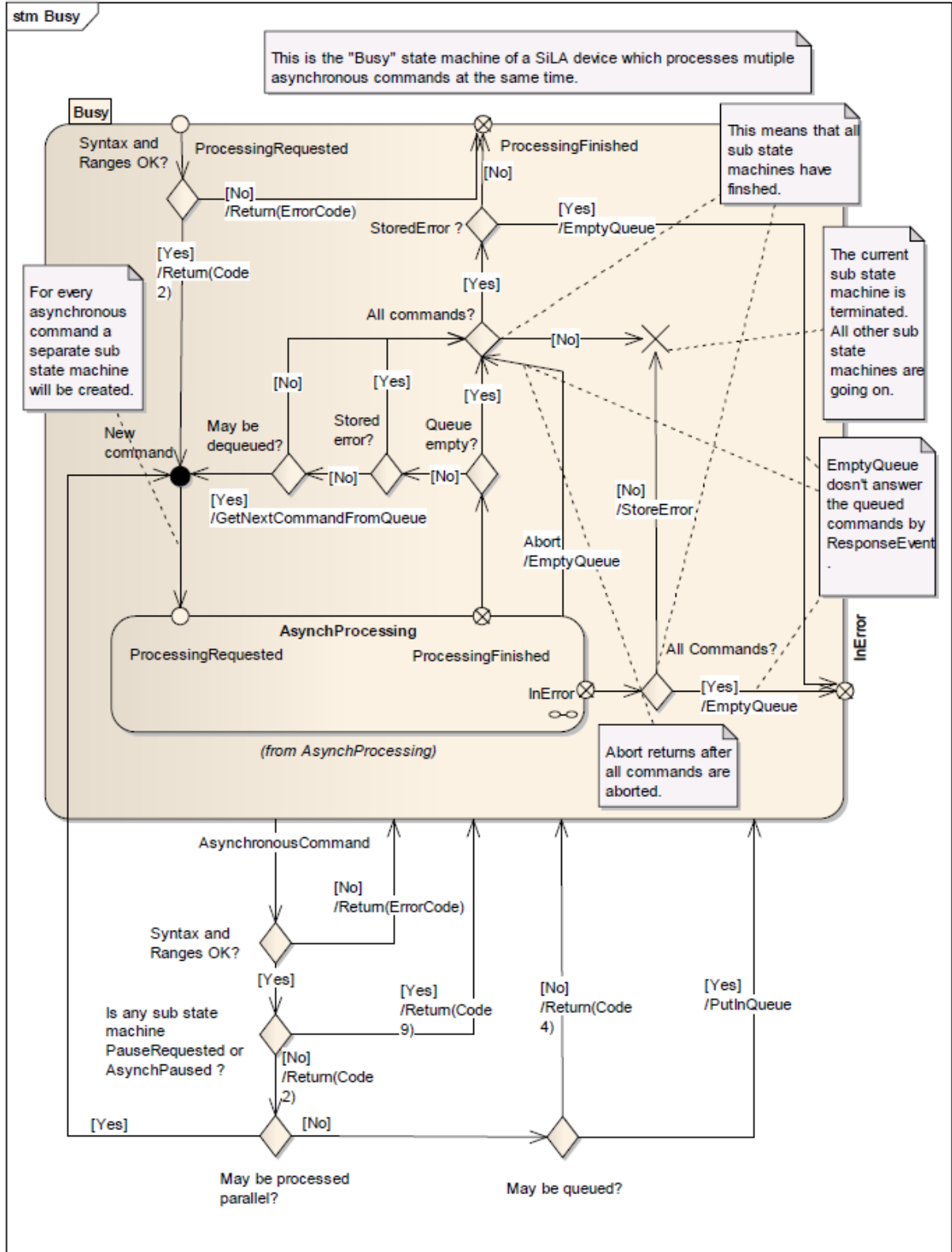


Figure 8: Sub State Machine for multiple asynchronous commands processing

SiLA Rapid Integration® Standardisation in Lab Automation	SiLA Device Control & Data Interface Specification	Specification Version: 1.3.08 Page 39 of 65
---	---	---

The AsynchProcessing state is detailed in Figure 9. Multiple instances of this sub-state machine are possible if the device allows parallel processing of multiple asynchronous commands.

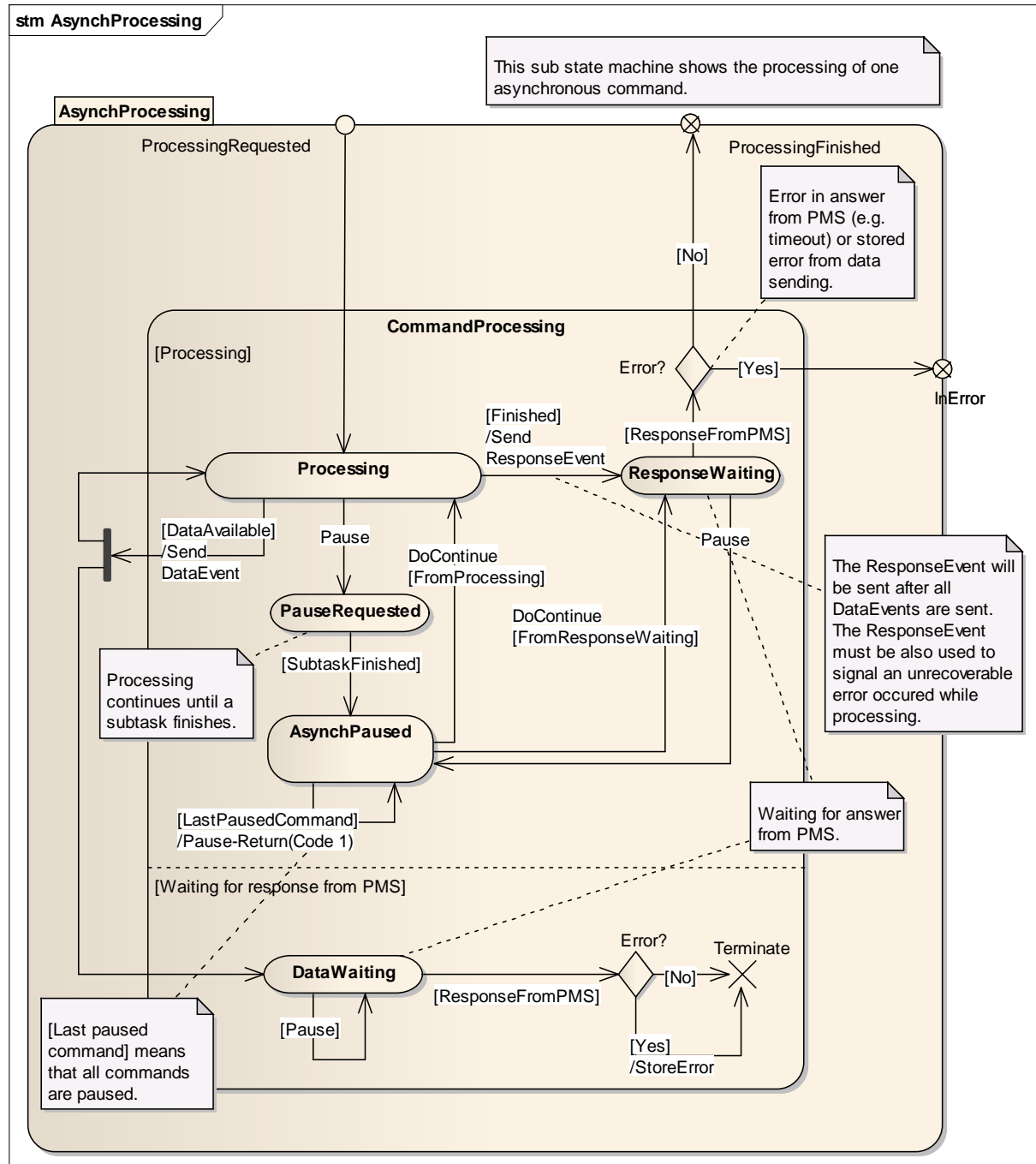



Figure 9: Detail of the AsynchProcessing state

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 40 of 65

3.5.1 States

ID:SI 22; Rev: 7

This section describes the states of the state machine of the *SiLA Service Provider*. The transition from one state to another one are not described, they are only contained in the diagrams. The states are described in alphabetical order.

3.5.1.1 resetting

ID:SI 23; Rev: 5

Handling the process of resetting as a separate state shall avoid the risk that several reset-commands are accepted concurrently or overwrite each other.

3.5.1.2 busy

ID:SI 24; Rev: 5

In this state the *SiLA Service Provider* is working. It has some sub states which are described below.

3.5.1.3 inError

ID:SI 30; Rev: 5

In this state the *SiLA Service Provider* has detected an error, which is not recoverable. This can also occur during communication with the *SiLA Event Receiver*.

3.5.1.4 initializing

ID:SI 59; Rev: 3

The process of initializing can take quite some time. Therefore the state initializing is introduced so that the GetStatus command can report this transitory state.

3.5.1.5 errorHandling

ID:SI 31; Rev: 5


In this state the *SiLA Service Provider* has detected an error, which is recoverable: It has notified the *SiLA Event Receiver* by sending an Error Event and is now waiting for the continuation instruction. This instruction shall be delivered within the return value of the Error Event.

Also all actions shall be taken to ensure that an operator can interfere with the device or its environment without any risks.

3.5.1.6 idle

ID:SI 32; Rev: 2

In this state the *SiLA Service Provider* is ready to execute a command. The connection to the device is established if necessary. The hardware is initialized, e.g. motors are homed.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 41 of 65

3.5.1.7 paused

ID:SI 33; Rev: 5

In this state the *SiLA Service Provider* has taken all necessary actions to ensure that an operator can interfere with the device or its environment without any risks. The sub-state commands Paused is used to indicate that there are asynchronous commands that can be continued.

3.5.1.8 startup

ID:SI 34; Rev: 3

This state shall be reached after power on.

3.5.1.9 standby

ID:SI 35; Rev: 3

In this state the *SiLA Service Provider* has reset all software data. No actions are carried out with the device.

3.5.2 Sub-States

ID: SP 147; Rev: 9

For sequential processing the sub-states are relatively simple and are based on the state diagrams in figures 9 and 10. For parallel processing it can be complicated to define a sub-state of the whole device. Therefore the return value of the GetStatus command reports on all currently executing and queued commands in a list. The sub-states used in this list are shortly explained in the remainder of this section.

asynchPaused

The asynchronous command is paused. If not all subtasks are finished it can be continued with the DoContinue command.

pauseRequested

Pause was requested but the command (subtask) has not completed yet.

responseWaiting

Instrument is waiting for the answer from the EventReceiver upon sending of the ResponseEvent.

processing

The asynchronous command is currently processing.

dataWaiting

Instrument is waiting for the answer from the EventReceiver upon sending of a DataEvent.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 42 of 65

3.5.3 State Machine Events

ID:SI 36; Rev: 9

Events of the state machine are conditions within the state machine which cause the transition from one state to another one. There are two kinds of state machine events:

- Trigger events, caused by external conditions.
- Guard events, caused by execution results of activities within the source state. They are shown in the diagram by UML Guards, which are shown as text enclosed in square brackets.

ID: SP 80; Rev: 25

This section provides the description of the *SiLA Service Provider's* state machine events.

Table 7: State machine events of *SiLA Service Provider*

Trigger and action name	Description
Abort	The <i>SiLA Service Provider</i> has received the appropriate <i>Mandatory Command</i> .
GetDeviceIdentification	
GetStatus	
DoContinue	
Initialize	
LockDevice	
Pause	
Reset	
UnlockDevice	
AnyCommand	Any not mandatory command.
PowerOn	Power was switched on.
PowerOff	Power was switched off.
ContinuationTask	The <i>SiLA Event Receiver</i> has sent the continuation_task, such as "Repeat" within the SOAP Response of an Error Event.
[TimeOut]	A timeout was encountered while in state <i>ErrorHandling</i>
EmptyQueue	Delete the commands in the queue because device is in the <i>inError</i> state.
PutInQueue	Device is processing a command and cannot start a new one therefore put the new command into the queue.
GetNextCommandfrom Queue	Start processing the next command in the queue
[OK]	The activity in the source state finished successfully.
[Data available]	There is some data available, which has to be sent to the <i>SiLA Event Receiver</i> without finishing the asynchronous command.
[LockTimeout]	Device has remained in the idle state for longer than the <i>LockTimeout</i> period. This triggers an internal reset and <i>unlockDevice</i> so that the device rests in the <i>Standby</i> state.
[LastPausedCommand]	All parallel executing commands are either paused or finished. Triggers the return of the <i>Pause</i> command.
[SubtaskFinished]	Subtask successfully finished
[Finished]	Processing finished

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 43 of 65

If any trigger occurred and there is no transition defined in the state machine for the current state, the *SiLA Service Provider* MUST reject the received command with the error code 9 (see section 8.3). In such a case the current state MUST be unchanged.

4 Security

ID:SI 37; Rev: 5

The SiLA standard does not define any security related aspects.

The standard definition ensures the possibility to use the PMS and devices in certain environments:

- Internet
- Intranet (corporate network)
- Isolated private network. A physically separated network containing all *SiLA Service Providers* which will be controlled by the PMS.

Therefore the applier can build up a system fulfilling different security requirements.

By using the protocol HTTPS it is possible to set up an advanced security environment.

4.1 SiLA and HTTPS

ID: SP 81; Rev: 4

The following section describes the model for using HTTPS in a SiLA System.

If HTTPS is used for communication between the *SiLA Service Consumer* and the *SiLA Service Provider* HTTPS MUST also be used for communication between the same *SiLA Service Provider* and the *SiLA Event Receiver*.

It is possible to use HTTPS only for a subset of SiLA Devices of a SiLA System. Therefore the PMS MUST be able to configure this subset.

4.1.1 Creation of Certificates


ID:SI 38; Rev: 4

SiLA uses server certificates for HTTPS. They MAY be self-signed certificates. SiLA does not define the way to create such certificates. Client certificates will not be used.

ID: SP 82; Rev: 10

For every *SiLA Service Provider* in the SiLA System communicating by HTTPS the applier must create a certificate with the following properties:

- The IP-Address of the *SiLA Service Provider* or its hostname. The same entry has to be used in the WSDL for the Web Service location.
- The serial number of the certificate.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 44 of 65

The certificates must be installed in the *SiLA Service Provider* HTTPS hosting entity, such as Microsoft IIS. Every SiLA Device, which supports HTTPS, **MUST** deliver an instruction how to do this in the device environment.

Both the IP-Address/hostname and the serial number of the SiLA Device's certificate **MUST** be available for the PMS through a trusted environment.

To use at least one SiLA Device by HTTPS the applier **MUST** create a certificate for the *SiLA Event Receiver* containing the same information as the certificates for the *SiLA Service Providers*. Such certificate must be installed in the *SiLA Event Receiver* HTTPS hosting entity, such as Microsoft IIS. Every PMS, which supports HTTPS **MUST** deliver an SOP on how to implement this in the PMS environment.

The certificate of the *SiLA Event Receiver* **MUST** be declared as trusted at all SiLA Devices, which support HTTPS. Alternatively the *SiLA Service Provider* **MUST** check the certificate (e.g. serial number against CRL). Every SiLA Device, which supports HTTPS and does not check the certificate itself, **MUST** deliver an SOP on how to declare a certificate as trusted in the device environment.

If the *SiLA Service Provider* gets its IP-Address by DHCP the configuration of the DHCP-Server **SHOULD** ensure that the IP-Address or hostname will be assigned to the *SiLA Service Provider* for a sufficient period of time. This can be achieved by increasing the lease time, by assigning IP-Addresses on the basis of their MAC-addresses, or by allowing the devices to update the DNS server (see RFC 2136).

4.1.2 Usage of Certificates

ID: SP 83; Rev: 22

To secure the connection between the *SiLA Service Consumer* and one or more *SiLA Service Providers*, the *SiLA Service Consumer* **MUST** send the Reset command to these *SiLA Service Providers* with an URI starting with "https:".

Then the hosting entity of the *SiLA Service Provider* will send the certificate. The *SiLA Service Consumer* **MUST** implement the check algorithm of the received certificate (e.g. serial number against CRL).

If any of the checked data is wrong the *SiLA Service Consumer* **MUST** generate an error and stop further processing.

If the *SiLA Service Provider* was called with a Reset by HTTPS it **MUST** send a Status Event within the Reset method by HTTPS before it returns. Then the *SiLA Event Receiver* sends its certificate to the calling *SiLA Service Provider*. If an error occurs on checking the certificate the *SiLA Service Provider* **MUST** detect this and **MUST** return an error and go into the InError state.

Figure 10 illustrates the handshake mentioned above. There SendCertificate() will be initiated by the hosting entity of the Web Service. The receiving hosting entity of the certificate may check it itself without interaction with the Web Service (e.g. for Device) or may delegate the check to the Web Service (e.g. for PMS, see above) and generate the

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 45 of 65

answer (CertificateOK or any error message). Then the Web Service, which has received the StatusEvent() (resp. the Reset()), returns.

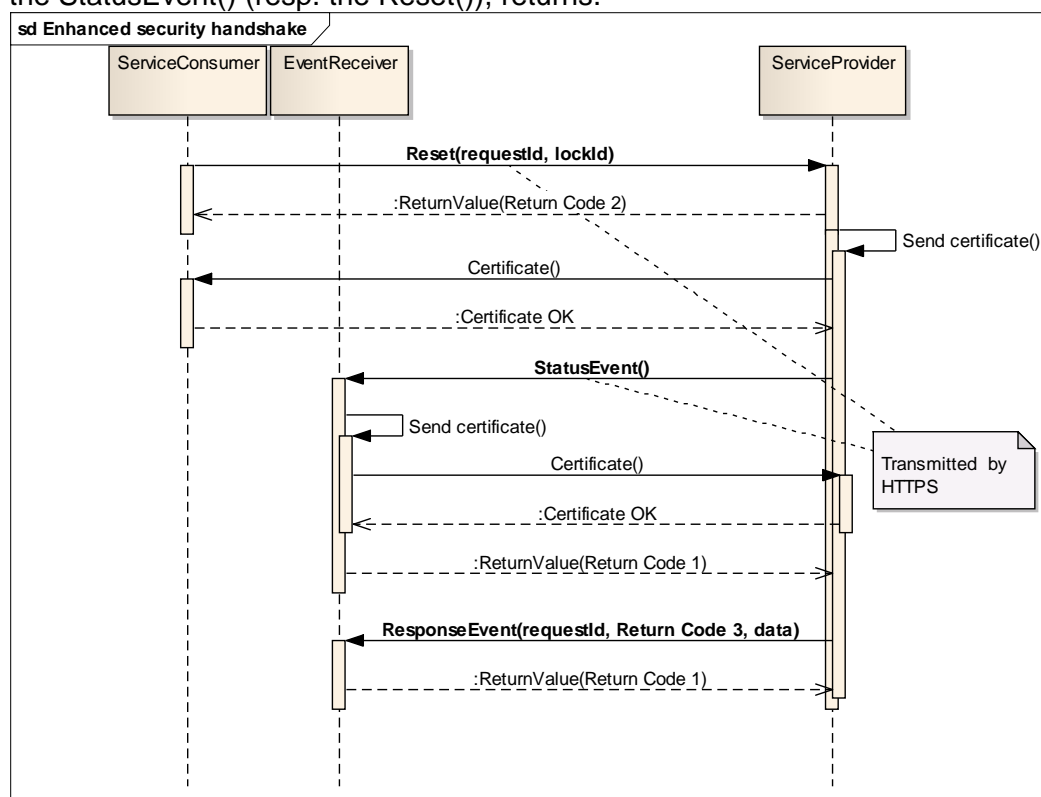


Figure 10: Enhanced security handshake

5 SiLA Integration

ID:SI 39; Rev: 19

The implementation of the SiLA standard requires a redesign of the device control software. In order to enable a fast transition to SiLA compliant devices different integration levels are introduced.

To illustrate the integration levels the next figures contain UML deployment diagrams with some possible configurations. Components with names starting with “Node” represent a computer and all components with stereotype <<device>> a laboratory device.

SiLA Integration Level 3 (SiLA Certified Equipment)

Devices fulfilling this level incorporate the Service Provider in the device. The device is connected to the PMS by wired Ethernet. A necessary exchange of Device1 by another device of the same Device Class and same configuration level will not be affected by any hardware resources. The integration of such a device does not require installation of any software on the computer hosting the PMS.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 46 of 65

Figure 11 shows the preferred configuration. The SiLA Device itself contains the SiLA Service Provider accessible by wired Ethernet.

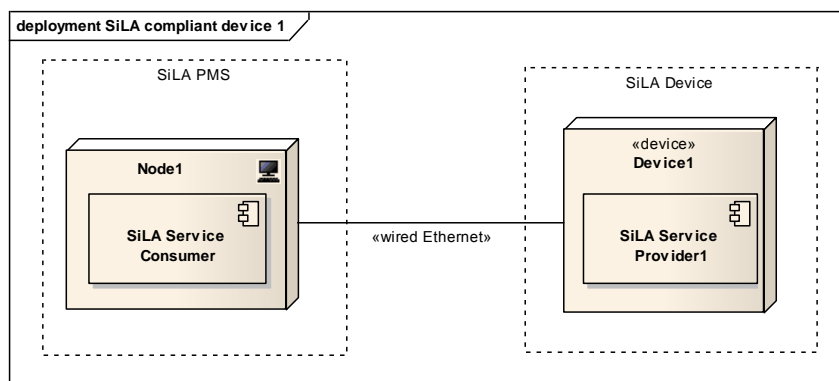


Figure 11: SiLA integration Level 3

ID:SI 40; Rev: 18

SiLA integration Level 2 (SiLA Supported Equipment)

Devices fulfilling level 2 consist of the device and a “translator” device. The translator device can be a PC, a microcontroller, an interface converter/adaptor, or a similar unit. The translator device connects to the computer hosting the PMS by wired Ethernet and to the device by the appropriate interface (most commonly RS232 Serial). The translator device acts as the SiLA Service Provider and translates the service requests to instructions for the device. Such a configuration enables the correct integration of legacy devices using RS232, USB, or similar ports. Device exchanges with a device complying with level 2 or 3 are still straight forward but require exchange of the device and the translator unit. The integration of such a device again does not require installation of any software on the computer hosting the PMS.

(Translator devices could also be interface adapters, which are available from different vendors for a hardware cost in the \$100 range)

Figure 12 shows the case where the SiLA Service Provider is hosted by an external computer / controller Node2.

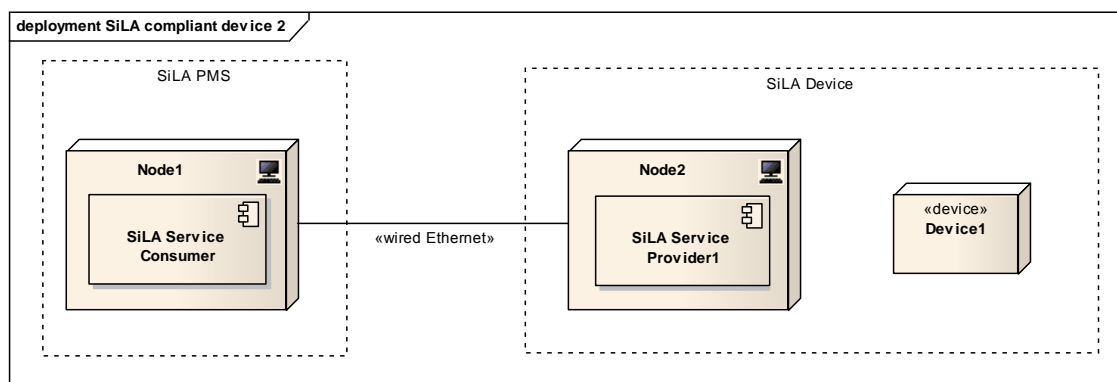


Figure 12: SiLA integration Level 2

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 47 of 65

ID:SI 42; Rev: 19

SiLA integration Level 1 (SiLA Compliant Driver)

Devices complying to level 1 implement the SiLA standard by software only. The Web Service controlling the device is hosted on the computer also hosting the PMS. The PMS accesses the Web Service via the localhost. The device can be connected to the computer by any interface including RS232, USB, and Ethernet. Device exchanges are rather complex because new software has to be installed on the computer and the vendor has to provide installation routines. (Also all disadvantages of using serial ports remain).

Figure 13 shows a level 1 configuration with the service provider installed on the computer hosting the PMS and the device is attached by any interface.

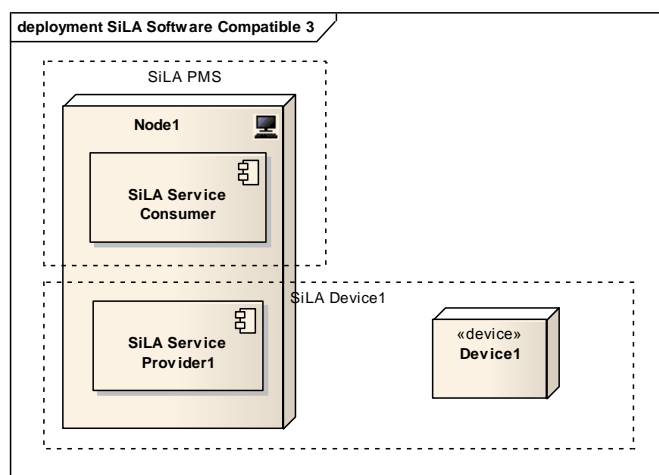


Figure 13: SiLA integration Level 1

For integrators exchanges of devices of integration levels 2 and 3 are relatively simple. Replacement of a device by a device of level 1 are more involved and should be avoided.

6 Comments

ID:SI 43; Rev: 2

This section illustrates certain specific points for discussion purposes. These comments are not part of the SiLA Standard.


6.1 Web Service Discovery

ID:SI 44; Rev: 3

Because the IP-Address of a device may be assigned manually or by DHCP, there must be a way to configure the *SiLA Service Provider* to use e.g. a new device.

The discovery may be done in two steps

- Get the IP-Address of the device
- Construct the URI of the Web Service of the device

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 48 of 65

6.1.1 Get the IP Address

ID:SI 45; Rev: 4+

If the IP-address is assigned manually this address is known.

If it is assigned by DHCP at least the MAC-Address of the device must be known to integrate the device into an existing configuration. Then it is possible to get the assigned IP-Address by the MAC-Address. Therefore freely available DHCP tools can be used.

Another possibility to get the IP-Address is the implementation of discovery in the PMS by using a DHCP-Client application or the AutoIP like implementation and WS-Discovery mechanisms as mentioned in chapter 2.1.6.1.

6.1.2 Build the URI

ID:SI 46; Rev: 6

Because the Web Service name and the TCP-Port number are known (from device documentation or using WS-Discovery), it is possible to build the Web Service URI and therefore to configure the *SiLA Service Consumer* to use the device.

6.1.3 Discover Functionality

ID:SI 47; Rev: 7

If the Web Service URI is known then it will be possible to discover the full functionality by


- Analyzing the WSDL which can be read from the Web Service itself (for static information).
- Calling *GetDeviceIdentification* for current information (see section 3.1)

Of course if the Web Service URI is not known, there is optionally possibility to query it automatically using WS-Discovery mechanisms as mentioned in chapter 2.1.6.1.

6.2 Performance

ID:SI 48; Rev: 2

To consider the performance of the provided solution, one can distinguish between several different aspects such as transmission time, CPU load, and memory usage. The aspects CPU load and memory usage can be considered for both, *SiLA Service Provider* and *SiLA Service Consumer*. However the Service Consumer typically is installed on a personal computer with enough memory and CPU power so that it is sufficient to consider those aspects for the *SiLA Service Provider*.

	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 49 of 65

6.2.1 Transmission Time

ID:SI 49; Rev: 9

The time for transmission of data packages depends on their size. A message that encodes a bar code shall have 40 byte. If the data is transmitted over a serial line (RS232) and after 8 data bits one additional bit is used as stop bit or parity check, 360 bits will be necessary to be transmitted. So it would need 18.75 ms to transmit that message with 19200 bps.

By the use of Web Services, the size of data packages is increased. Sending such a bar code as a SOAP object can easily result in a data package of 700 byte. However, the Web Services are accessible using Ethernet or a faster technology. Using Ethernet, the data package is transmitted with 10Mbit/s. Due to TCP's acknowledges and the protocol stack an effective data rate of 70%, which means 7Mbit/s is assumed. In such case the data package of 700 byte will be transmitted within 0.8 ms.

Therefore the drawback of data packages increment is irrelevant because of the use of Ethernet.

6.2.2 Memory usage

ID:SI 50; Rev: 4

To measure memory usage and CPU load, a 1.6 GHz personal computer with 256 MB of memory was used to host simple echo servers. Two different kinds of echo servers were tested, both were realized in the Java programming language. The first one offers its service based on a socket connection, the other one offers its service based on web services.

An idling socket connection initially used 3.9% of the memory. The same application used 4.9% of the memory for answering requests. After answering requests the amount of memory remained 4.9%, even if no more messages needed to be answered.

The web service application initially used 5.1% of the machines memory. After it had answered the first request it used 5.6% of the memory. If no more messages arrived the amount of memory would have remained at 5.6%.

The tests were made with messages every 250 ms and every 100 ms.

6.2.3 CPU Load

ID:SI 51; Rev: 3

Again the same applications were used for comparison. The realization based on socket connection produced a load between 0 and 7% of the CPU. The web service realization only produced a load between 0 and 3% of the CPU.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 50 of 65

7 References

- [XSD] Paul Biron, Ashok Malhorta: *XML Schema Part 2: Datatypes Second Edition*, URI: <http://www.w3.org/TR/xmlschema-2>, W3C, 2004.
- [SOAP11] Don Box et al.: *Simple Object Access Protocol (SOAP) 1.1*, URI: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, W3C, 2000.
- [SOAP12P0] Nilo Mitra, Yves Lafon: *SOAP Version 1.2 Part 0: Primer (Second Edition)*, URI: <http://www.w3.org/TR/soap12-part0>, W3C, 2007.
- [SOAP12P1] Martin Gudgin et al.: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, URI: <http://www.w3.org/TR/soap12-part1>, W3C, 2007.
- [SOAP12P2] Martin Gudgin et al.: *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*, URI: <http://www.w3.org/TR/soap12-part2>, W3C, 2007.
- [WSDL] Erik Christensen et al.: *Web Services Description Language (WSDL) 1.1*, URI: <http://www.w3.org/TR/wsdl>, W3C, 2001.
- [HTTP] Roy Fielding et al.: *Hypertext Transfer Protocol – HTTP/1.1*, URI: <http://tools.ietf.org/html/rfc2616>, IETF, 1999.
- [Basic Profile] Keith Ballinger et al.: *Basic Profile Version 1.1*, URI: <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>, Web Services Interoperability Organization, 2004.
- [RFC2119] Scott Bradner: *Key words for use in RFCs to Indicate Requirement Levels*, URI: <http://www.ietf.org/rfc/rfc2119.txt>, IETF, 1997.
- [RFC2136] Paul Vixie: *Dynamic Updates in the Domain Name System (DNS UPDATE)*, URI: <http://www.ietf.org/rfc/rfc2136.txt>, IETF, 1997.
- [RFC3927] Stuart Cheshire et al.: *Dynamic Configuration of IPv4 Link-Local Addresses*, URI: <http://www.ietf.org/rfc/rfc3927.txt>, IETF, 2005.
- [RFC4862] Susan Thomson et al.: *IPv6 Stateless Address Autoconfiguration*, URI: <http://www.ietf.org/rfc/rfc4862.txt>, IETF, 2007.
- [LLMNR] Bernard Aboba et al.: *Link-local Multicast Name Resolution (LLMNR)*, URI: <http://tools.ietf.org/html/draft-ietf-dnsext-mdns-47>, IETF, 2006.
- [Guide] Remo Hochstrasser, Dieter Speidel, Christoph Karthaus, *SiLA Device Class Interface Design Guidelines*, SiLA, 2010. URI: <http://www.sila-standard.org/downloads/>
- [DCS] Remo Hochstrasser et al.: *SiLA Data Capture Specification*, SiLA, 2013, gives you information about Data transfer methodologies and how to structure XMLdocument based Schemas based on all possible data types. You can find it on the SiLA Website under: URI: <http://www.sila-standard.org/downloads/>
- [GISAPP] Christoph Karthaus et al.: *SiLA General Interface Specification Appendix*, URI: <http://www.sila-standard.org/downloads/>, SiLA, 2010.
- [CCD] Christoph Karthaus et al.: *SiLA Common Command Dictionary*, SiLA, 2010, URI: <http://www.sila-standard.org/downloads/>
- [CCD Overview] Common Command Dictionary overview table SiLA, 2010, URI: <http://www.sila-standard.org/downloads/>
- [RFC3927] Cheshire et al.: *Dynamic Configuration of IPv4 Link-Local Addresses*, URI: <http://www.ietf.org/rfc/rfc3927.txt>, IETF, 2005.
- [RFC4862] Thomson et al.: *IPv6 Stateless Address Autoconfiguration*, URI: <http://www.ietf.org/rfc/rfc4862.txt>, IETF, 2007.
- [WS-Discovery] Nixon et al.: *Web Services Dynamic Discovery (WS-Discovery) Version 1.1*, URI: <http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html>, OASIS-Standard, 2009.
- [WSDL Sample] An example WSDL file for a simple Device can be downloaded as a reference from here: <http://www.sila-standard.org/downloads/>
- [EventReceiver] An example Event Receiver WSDL file for a simple PMS can be downloaded as a reference from here: <http://www.sila-standard.org/downloads/>
- [Labware Spec] Labware Specification gives you information about all possible Labware types that are accepted and whose available data and types are foreseen for Labware. You can find it on the SiLA Website under: <http://www.sila-standard.org/downloads/>

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 51 of 65

8 Appendix

ID:SI 54; Rev: 3

The following commands **MUST** be implemented by every *SiLA Service Provider* and *SiLA Service Consumer*. That is why they are called **Mandatory Commands**.

8.1 Mandatory Command description

ID:SI 53; Rev: 9+

This section describes the *Mandatory Commands* and events.

The parameters of the commands are described in a table of the following format

Name	Type

The column's titles have the following meanings:

- **Name** is the name of the parameter
- **Type** is the XML-Schema type (see section 2.1.1.2)

All of those parameters are not nullable (must be present) for **Mandatory Commands**, you can specify empty string ("") or 0 values.

In this table there is information, which can be extracted from the WSDL-file of the *SiLA Service Provider*.

The additional information is described as SiLA Web Service Documentation Extensions, which may be used as copy template by the developer of a SiLA component. The shown XML structure contains exemplary values on certain places, which must be replaced by the device developer. This is marked by using "xxxxxx" in the XML. All other definitions are fixed by the SiLA standard.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 52 of 65

8.1.1 Abort

ID: SP 84; Rev: 20+

The abort command shall stop all running and queued asynchronous commands (busy state). If no error occurs, the device shall be in the idle state. After completion of the abort command no response events for the aborted asynchronous commands shall be fired any more.

8.1.1.1 Parameter

Name	Type
requestId	Int
lockId	String

8.1.1.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-
standard.org/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command aborts all running and pending asynchronous commands of the
    device.
  </Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId" >
    This parameter is the identification of the PMS which has locked
    the device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.1.3 Return values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId does not match the current one. Therefore the command is rejected. The state of the device is not changed.
6	The requestId has an invalid value.
9	Command not allowed in this state

8.1.1.4 Response Data

No response will be expected, therefore only the XML frame has to be sent, such as

```
<?xml version="1.0" encoding="utf-8"?>
<ResponseData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ResponseType_1.2.xsd"/>
```

SILA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 53 of 65

8.1.2 DoContinue

ID: SP 85; Rev: 19+

The DoContinue command shall enable the continuation of the process (workflow) for a paused system. This means that paused asynchronous commands shall be continued and/or that the system shall start working on the next (queued) command in the process (workflow).

8.1.2.1 Parameter

Name	Type
requestId	Int
lockId	String

8.1.2.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command enables the continuation of the process for a paused
    system.
  </Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId" >
    This parameter is the identification of the PMS, which has locked
    the device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.2.3 Return Values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId doesn't match the current one. Therefore the command is rejected. The state of the device isn't changed.
6	The requestId has an invalid value.
9	Command not allowed in this state.

8.1.2.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

8.1.3 GetDeviceIdentification

ID: SP 88; Rev: 22+

The GetDeviceIdentification command shall report a set of device specific data to the PMS. A detailed description of the transferred data can be found in the section 3.1

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 54 of 65

8.1.3.1 Parameter

As the GetDeviceIdentification command is a synchronous command, it uses out-parameters to return data. The deviceDescription is an out-parameter and SOAP complex type (see also 3.1).

Name	Type
requestId	Int
lockId	String
deviceDescription	SiLA_DeviceIdentification

8.1.3.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command reports on details of the device.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    This parameter is the identification of the PMS which has locked
    the device.
  </Parameter>
  <Parameter name="deviceDescription" >
    The Device Identification. It is a SOAP complex type.
  </Parameter>
</SiLACommandDescription>
```

8.1.3.3 Return values

Code	Description
1	Success.
6	The requestId has an invalid value.

8.1.3.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

8.1.4 GetStatus

ID: SP 89; Rev: 23+

The GetStatus command reports the state of the device (in terms of the state machine).

8.1.4.1 Parameter

As the GetStatus command is a synchronous command, it uses out-parameters to return data. All parameters except for the requestId are out-parameters.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 55 of 65

Name	Type
requestId	Int
deviceId	String
state	Enumeration [State]
subStates	Array of Command Description
locked	Boolean
PMSId	String
currentTime	dateTime

As value for the subStates null will be returned, if the device is in the state startup, standby, or idle. Otherwise each CommandDescription is composed of the members:

CommandDescription

Name	Type (see 2.1.1.1)	Possible values	Description
requestId	Int	1, ..., 2147483647	The request ID of the referring command.
commandName	String	No length restrictions.	The name of the referring command.
queuePosition	Int	1, ..., 2147483647	
startedAt	dateTime	No restrictions.	If the Command is waiting in the queue to start, null will be returned.
currentState	Enumeration [Substate]	See table below.	The actual substate. If the Command is waiting in the queue to start, null will be returned.
dataWaiting	Int	0, ..., 2147483647	A count of data waiting sub states. If the Command is waiting in the queue to start, null will be returned.

The possible values of the Enumerations State and Substate are listed below:

State	Substate
startup resetting standby initializing idle busy paused errorHandling inError	asynchPaused pauseRequested processing responseWaiting dataWaiting

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 56 of 65

8.1.4.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command reports the status of the device.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="deviceId">
    The identification the device returns to identify itself at the PMS.
  </Parameter>
  <Parameter name="state">
    Status of the device
  </Parameter>
  <Parameter name="subStates">
    Substate of the above state.
  </Parameter>
  <Parameter name="locked">
    Lock state of the device (locked=true, unlocked = false)
  </Parameter>
  <Parameter name="PMSId">
    Identification of the PMS that locked the device.
  </Parameter>
  <Parameter name="currentTime">
    Time of reporting status information.
  </Parameter>
</SiLACommandDescription>
```

8.1.4.3 Return values

Code	Description
1	Success.
6	The requestId has an invalid value.

8.1.4.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

8.1.5 Initialize

ID: SP 90; Rev: 20+

This command shall initialize the SiLA Service Provider. It shall only be possible to be invoked when the device is in the “standby” state.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 57 of 65

8.1.5.1 Parameter

Name	Type
requestId	Int
lockId	String

8.1.5.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command initializes the device.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter ist the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    This parameter is the identification of the PMS which has locked
    the device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.5.3 Return values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId does not match the current one. Therefore the command is rejected. The state of the device is not changed.
6	The requestId has an invalid value.
9	Command not allowed in this state.

8.1.5.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

8.1.6 LockDevice

ID: SP 157; Rev: 9+

If the Service Consumer wants to lock a device for exclusive use, it shall invoke a LockDevice command with a generated String as value of the parameter "lockId". Subsequently the Service Provider shall only accept commands from a Service Consumer, which uses the same lockId in the command calls.

The SiLA Service Consumer **MUST** ensure the uniqueness of the lockId to avoid conflicts with other SiLA Service Consumers.

The lockTimeout parameter is used to unlock the device automatically after the indicated time period has passed, without receiving commands, by executing a Reset and the UnlockDevice commands internally on the device. The device will then be in the state Standby. If the lockTimeout parameter is 0 or Null, then the locking is permanent until the next UnlockDevice or a power shutdown.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 58 of 65

8.1.6.1 Parameter

Name	Type
requestId	Int
lockId	String
lockTimeout	Duration
eventReceiverURI	String
PMSId	String

8.1.6.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command locks the device for exclusive use.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId" >
    This parameter hands over the lock identification of the PMS to the device. The
    device will only accept further commands, if they use the same lockId.
  </Parameter>
  <Parameter name="lockTimeout" xsi:type="Duration">
    If this parameter is omitted, no timeout will be set. Otherwise
    the device will unlock itself if it does not receive any commands within the
    timeout period.
  </Parameter>
  <Parameter name="eventReceiverURI" >
    Service URI of the Service Consumer's event Receiver (used in case of timeout
    that will issue a reset with it).
  </Parameter>
  <Parameter name="PMSId" >
    Id of the PMS in order to identify the PMS that locked a device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.6.3 Return values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId does not match the current one. Therefore the command is rejected. The state of the device isn't changed.
6	The requestId has an invalid value.
9	Command not allowed in this state.

8.1.6.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

SILA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 59 of 65

8.1.7 Pause

ID: SP 91; Rev: 20+

The Pause command shall initiate a routine that turns the device into a “safe” state. Commands of the Common Command Set and Specific Commands shall be stopped after the completion of the current subtask and the state change to *busy* with a sub-state *asynchPaused* for each command shall be triggered. In such a state the PMS may now allow a user to interfere manually with the system. After completion of the manual tasks the process can be resumed by calling the DoContinue command.

8.1.7.1 Parameter

Name	Type
requestId	Int
lockId	String

8.1.7.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command pauses the process/workflow in order to enable user
  intervention.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    This parameter is the identification of the PMS, which has locked
    the device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.7.3 Return values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId does not match the current one. Therefore the command is rejected. The state of the device it not changed.
6	The requestId has an invalid value.
9	Command not allowed in this state.

8.1.7.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 60 of 65

8.1.8 Reset

ID: SP 92; Rev: 22+

This command shall reset the SiLA Service Provider. All queues shall be emptied without sending response events of already started or queued commands. It shall be callable at any time and from any state of the devices state machine. It shall also be used to report the connection information of the service consumer's event receiver available to the SiLA Service Provider (or to overwrite it in subsequent calls). Additionally sends the deviceId to the Service Provider so that the service Provider can identify itself at the Service Consumer in case of a status event.

8.1.8.1 Parameter

Name	Type
requestId	Int
lockId	String
deviceId	String
eventReceiverURI	String
PMSId	String
errorHandlingTimeout	Duration
simulationMode	Boolean

8.1.8.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command is used to reset the Service Provider at any time from any
    state.
  </Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId" >
    This parameter is the identification of the PMS, which has locked
    the device.
  </Parameter>
  <Parameter name="deviceId" >
    The identification the device returns to identify itself at the PMS.
  </Parameter>
  <Parameter name="eventReceiverURI" >
    Connection information of the Service Consumers event Receiver.
  </Parameter>
  <Parameter name="PMSId" >
    Id of the PMS in order to identify the PMS that locked a device.
  </Parameter>
  <Parameter name="errorHandlingTimeout" xsi:type="Duration">
    Timeout until an errorhandling state is changed into an error state.
  </Parameter>
  <Parameter name="simulationMode" >
    Selects simulation mode.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.8.3 Return Values

Code	Description
------	-------------

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 61 of 65

2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId does not match the current one. Therefore the command is rejected. The state of the device isn't changed.
6	The requestId has an invalid value.
7	The deviceId is invalid.
9	Command not allowed in this state

8.1.8.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

8.1.9 UnlockDevice

ID: SP 94; Rev: 25+

The UnlockDevice command shall unlock the device and allow access to the device by any Web Service client.

8.1.9.1 Parameter

Name	Type
requestId	Int
lockId	String

8.1.9.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command is used to unlock the device.
</Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="lockId">
    This parameter is the identification of the PMS which has locked
    the device.
  </Parameter>
  <Response xsi:type="standardResponse" parameterSetCount="0">
    <Description>Empty response</Description>
  </Response>
</SiLACommandDescription>
```

8.1.9.3 Return values

Code	Description
2	Asynchronous command accepted
3	Asynchronous command has finished
5	The device is locked and the given lockId doesn't match the current one. Therefore the command is rejected. The state of the device isn't changed.
6	The requestId has an invalid value.
9	Command not allowed in this state.

8.1.9.4 Response Data

No response will be expected, therefore only the XML frame has to be sent (see section 8.1.1.4).

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 62 of 65

8.2 Event Command Description

The following sections list the methods of the SiLA Event Receiver. All these methods are synchronous and the SiLA Return Value is used as response.

8.2.1 DataEvent

ID: SP 86; Rev: 19+

The data event shall enable the transmission of data generated during an asynchronous command execution to the PMS.

The schema of the parameter dataValue is given by listing 5. The SiLA Data Capture Specification [DCS] gives more information about the referring schema.

8.2.1.1 Parameter

All Parameters are out-parameters

Name	Type
requestId	Int
dataValue	xmlDocument

8.2.1.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command is used to enable the transmission of data generated during
    an asynchronous command execution.
  </Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of this command call.
  </Parameter>
  <Parameter name="dataValue">
    The description and values of the data transferred.
  </Parameter>
</SiLACommandDescription>
```

8.2.1.3 Return values

Code	Description
1	Success. Event received.
6	The request id has an invalid value.
10	Data Error

8.2.2 ErrorEvent

ID: SP 87; Rev: 25+

The error event shall be fired when the device enters an error state that may be recovered by executing a special task. The event transmits a list of possible continuation tasks. The PMS may decide whether it executes the task marked as default task or to let the user decide which action to take.

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 63 of 65

If the user selects a continuation task, the selection is reported back to the SiLA Service Provider by setting the SelectedTask inside of the continuationTask.
The structure of the returnValue is given in section 2.1.1.3. The structure of the continuationTask MUST follow the schema of section 2.2.1.4.

8.2.2.1 Parameter

All Parameters except of the continuationTask are out-parameters. The continuation Task is an in/out-parameter.

Name	Type	
requestId	Int	Input parameter
returnValue	SiLA Return Value	Input parameter
continuationTask	xmlDocument	Input/Output parameter

8.2.2.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx "
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This event reports the occurrence of a recoverable error and sends the
    required information for intelligent error handling (list of possible actions
    for error recovery).
  </Summary>
  <Parameter name="requestId" minValue="1" maxValue="2147483647">
    This parameter is the unique identification of the command that generated the
    error.
  </Parameter>
  <Parameter name="returnValue">
    The complex type describing all details of the return value.
  </Parameter>
  <Parameter name="continuationTask">
    The complex type describing the possible recovery actions.
  </Parameter>
</SiLACommandDescription>
```

8.2.2.3 Return values

Code	Description
1	Success. Event received.
6	The requestId has an invalid value.
10	Data error on continuation task parameter.

8.2.3 ResponseEvent

ID: SP 93; Rev: 21+

The response event shall inform the PMS about the successful completion of an asynchronous command.

The structure of the returnValue is given in section 2.1.1.3. The schema of the parameter dataValue is given by listing 5. The SiLA Data Capture Specification [DCS] gives more information about the referring schema.

SILA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 64 of 65

8.2.3.1 Parameter

All Parameters are out-parameters

Name	Type
requestId	Int
returnValue	Sila Return Value
responseData	xmlDocument

8.2.3.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command is used to return the successful completion of an
asynchronous command.
</Summary>
<Parameter name="requestId" minValue="1" maxValue="2147483647">
  This parameter is the unique identification of the asynchronous command, which
is now finished.
</Parameter>
<Parameter name="returnValue">
  The complex type describing all details of the return value as would be for a
synchronous call.
</Parameter>
<Parameter name="responseData">
  The XML document describing additional data.
</Parameter>
</SiLACommandDescription>
```

8.2.3.3 Return values

Code	Description
1	Success. Event received.
6	The requestId has an invalid value.
10	Data Error in Response Data.

8.2.4 StatusEvent

ID: SP 100; Rev: 15+

The Status event shall be triggered in an unsolicited fashion. It shall be used to report changes in the device, which are not controlled by the state machine, such as over temperature, errors in keep-alive mechanisms, or errors in motor regulation.

The structure of the returnValue is given in section 2.1.1.3. The schema for the eventDescription is given in section 2.2.1.1.

8.2.4.1 Parameter

All Parameters are out-parameters

SiLA Rapid Integration®	SiLA Device Control & Data Interface Specification	Specification
Standardisation in Lab Automation		Version: 1.3.08 Page 65 of 65

Name	Type
deviceId	String
returnValue	SiLA Return Value
eventDescription	xmlDocument

8.2.4.2 Documentation

```
<SiLACommandDescription isCommonCommand="true" estimatedDuration="xxxxxx"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sila-standard.org
/schemata/SoapAnnotation_1.2.xsd">
  <Summary> This command is used to report device events.
</Summary>
  <Parameter name="deviceId">
    The identification the device returns to identify itself at the PMS.
  </Parameter>
  <Parameter name="returnValue">
    The complex type describing all details of the return value.
  </Parameter>
  <Parameter name="eventDescription">
    The complex type describing the event.
  </Parameter>
</SiLACommandDescription>
```

8.2.4.3 Return values

Code	Description
1	Success. Event received.
6	The requestId has an invalid value.
10	Data Error in eventDescription.

8.3 Common Return Codes

ID: SP 95; Rev: 17

The following table contains the common usable return codes, which are the value of the member “Code” in the *SiLA Return Value* structure.

Code	Description
1	Success
2	Asynchronous command accepted
3	Asynchronous command has finished
4	Device is busy due to other command execution
5	Error on lockId
6	Error on requestId
7	Error on deviceId
8	Error on certificate check
9	Command not allowed in this state
10	Error in Data sent to Event Receiver
11	Invalid command parameter
12	Finished with warning
13	Command unexecuted de-queued due to error in previous command execution