Interface Description

# SAP Auto-ID Infrastructure Device Controller Interface (SAP AII-DC 5.1)

For SAP Auto-ID Infrastructure 5.1

Document Version 5.10 – June, 2007

SAP AG
Dietmar-Hopp-Allee 16
69160 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

# Contents

# 1 About this Document

This document describes the interface between *SAP Auto-ID Infrastructure (SAP AII)* and an RFID device controller (DC). It includes sample messages that work with predefined SAP AII message processing rules.

SAP AII supports EPCglobal's Application Level Events (ALE) interface as well as the SAP AII *classic* message interface that consists of an *XML command message* sent from SAP AII to the device controller and an *XML notification message* sent from the device controller to SAP AII.

Here we define Version 5.1 of the *SAP Auto-ID Infrastructure Device Controller* Interface (*SAP AII-DC 5.1*) which is supported by SAP AII 5.1. This interface enables SAP partners to deliver RFID device controllers that are interoperable with SAP AII 5.0.

Device controllers that support the prior version of this interface definition (*SAP AII-DC 4.0 and 1.0*) can also interoperate with SAP AII 5.1. See Appendix 6.3 for a summary of additions to the interface since *SAP AII-DC 4.0* was published in 2006.

# 2 Overview

## 2.1 SAP Auto-ID Infrastructure

Radio frequency identification (RFID) is an important method for automating the identification (Auto-ID) of goods and for improving supply chain visibility. It enables goods that contain a chip with an antenna (RFID tag) to be tracked at several points along the supply chain. In contrast to barcode technologies that typically identify only the material of the shipment (for example, the EAN), most RFID applications focus on identifying each object in the shipment (pallet, case, or individual item), and give much more detailed information than barcodes alone. SAP AII enables backend systems such as mySAP ERP and supply chain visibility tools such as SAP Event Manager to leverage RFID technology. It connects the physical world, as observed by RFID device controllers, with the business-oriented document view of an ERP system. For example, the RFID detection of pallets and cases at a gate in the warehouse (physical world) can confirm that an expected inbound delivery (business document) is complete.

## 2.2 Architecture of an RFID Enabled Landscape

This is an example of an RFID enabled landscape:

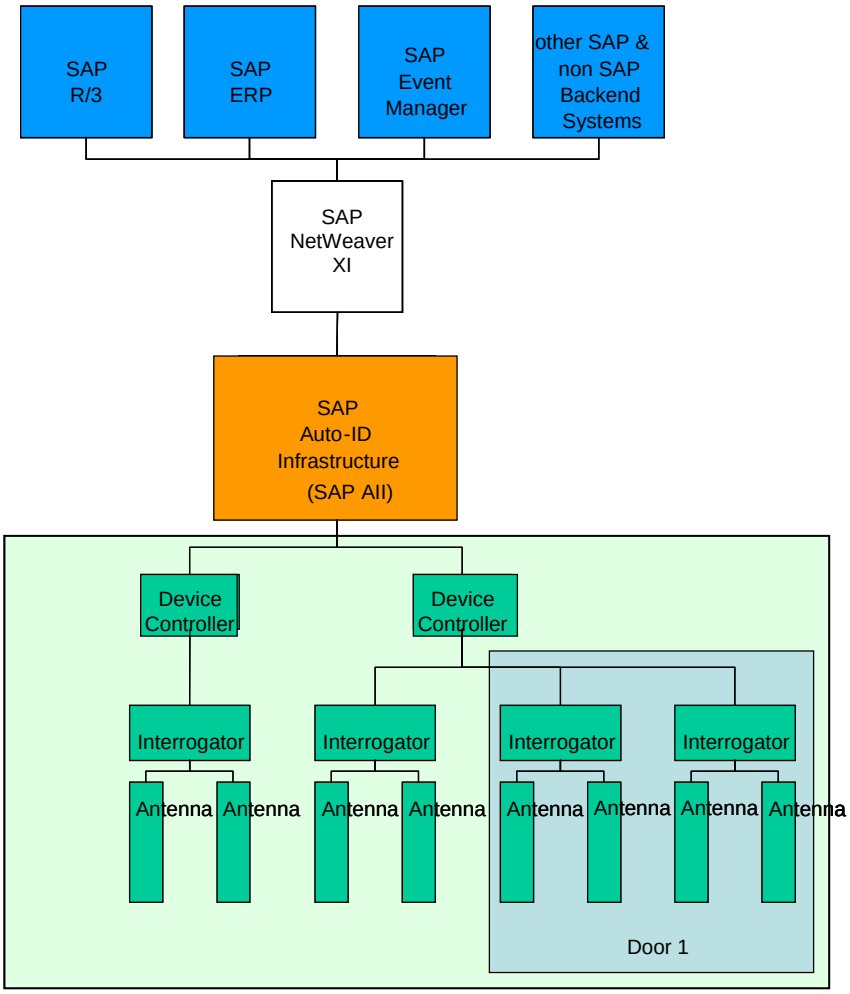



Figure 1

Device controllers are software components that link the SAP Auto-ID Infrastructure with interrogator hardware components. A device controller may run on the interrogator's hardware or on different hardware. One device controller may manage many interrogators.

An interrogator is a hardware device that reads RFID tags. It may have one or more antennae plus additional equipment, such as light sensors, that can determine the direction in which the goods are moving.

The components below the SAP Auto-ID Infrastructure in the shaded area of Figure1 are provided by SAP partners. In SAP AII, these components are modeled by RFID device controllers that manage groups of RFID devices. SAP AII RFID devices can represent interrogators, specific antennae, RFID printers, or non-RFID sensors such as light sensors and barcode readers.

The area labeled *Door 1* in Figure 1 shows the RFID devices in a single warehouse location. This location may be a door or gate where goods are moved from the warehouse and loaded onto a truck. For better reading accuracy and directional sensing, the device controller may use two interrogators to track the goods issued. In this case, a single SAP AII RFID device may be used to model both interrogators.

## 2.3 Message Processing in SAP Auto-ID Infrastructure

SAP AII uses *rules* to process the notification messages that it receives and it selects the appropriate rule based on information derived from the incoming message.

A rule is a sequence of *activities* that contain detailed application logic such as how to send a command message to a device controller. SAP AII includes many predefined rules and activities to support a variety of scenarios. Customers can use these as templates to develop additional customized rules and activities.

# 3 EPCglobal's ALE 1.0

SAP AII can act as a client to a device controller that supports EPCglobal's ALE 1.0.and it can be configured to call Web Service methods on the device controller to establish a subscription to RFID tag observations. Once the subscription is established, the device controller sends notification messages to SAP AII when observations occur.

Based on the options selected, the SAP AII Device Subscription transaction (/AIN/CS) can call the ALE services needed to subscribe and unsubscribe to notifications. In addition, customized SAP AII activities can call the ALE Web Services when more dynamic subscriptions are required.

You enable standard SAP AII processing of ALE notifications by selecting specific values for several attributes of the ALE ECSpec. If the ECSpec is included with the ECReport in the notification, SAP AII can know the device group and derive its location and if you request Raw Hex data, the IDs will be represented by the HEX values that SAP AII expects. SAP AII expects XML ALE notification messages that are posted via HTTP.

**Table: ALE Protocol Specific Attribute Settings in /AIN/CS**

| Attribute | Value |
|---|---|
| Include Spec In report | 1 |
| Include Raw Hex | 1 |

Since ALE does not have a concept similar to the SAP AII *Command* in the notification message, fixed reader message routing conditions are limited to reader location. For ALE notifications, the *Command* field in the SAP AII condition table should be set to blank.

For specific definitions of ALE services and the ALE notification, refer to the EPCglobal standard, *The Application Level Events (ALE) Specification, Version 1.0* at

http://www.epcglobalinc.org/standards/ale/ALE_1_0-StandardRatified-20050915.pdf

Note:

We suggest that you use ALE where possible. Use the SAP AII classic interface for tag writing, existing implementations, non-EPC applications, and other processes that require ALE extensions.

Additional ALE 1.0 (and SAP AII) extensions are required to support:

- Notification rule selection based on more than one command value
- Additional observation level values such as *container ID* and *document number*
- Non-EPC observations (tag write failure detection and case observations prior to tagging)
- Data on tag or other tag-related information

# 4 SAP AII Classic Message Interface

This chapter describes the SAP AII classic message interface that has two types of messages:

- Command messages that SAP AII *sends* to the device controller to control tag writing, subscriptions, and light towers
- Notification messages that SAP AII *receives* from the device controller to report tag observations and other events

Later in the chapter we provide a technical description of the XML structure of each message type followed by sample messages that are used with predefined SAP AII activities.

The message elements (fields) and their parent–child relationships are described by illustrations (figures) and additional notes are provided where necessary for clarification.

**Notes on the figures in the following sections:**

- Each rectangle ( 

) represents an element and shows its name. A plus sign at the right edge of the element's symbol indicates that it includes further elements which are shown in a separate figure. A minus symbol indicates that the child elements are included in the same figure.
- The choice symbol (  ) indicates that the element on its left can only have one of the elements on the right as a child element.
- The sequence symbol (  ) indicates that the element on its left has all the elements on the right as children and the order of the elements on the right must be maintained.
- If one element can be a member of another element more than once, that is noted near the element. If the element can be omitted, the border of the rectangle is dotted ( 

).

## 4.1 Command Message

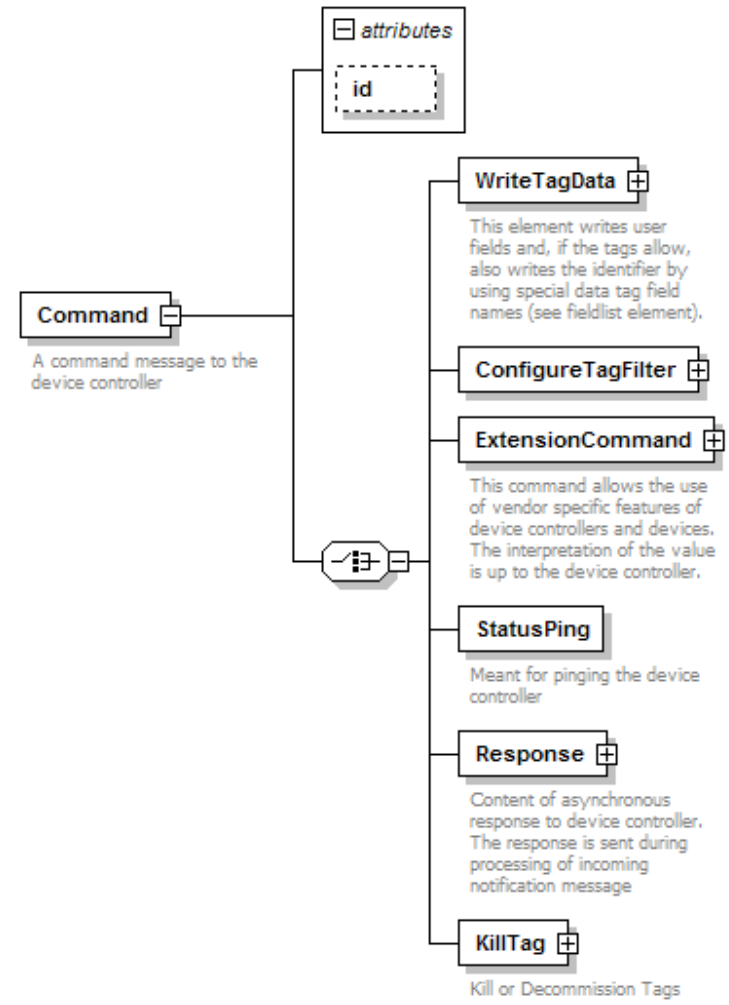
### 4.1.1 Elements of the Command Message

The XML schema for the command message is provided in Appendix 6.1. Details of the Command element is given below

#### 4.1.1.1 Details of the Command Element

Predefined SAP AII activities send five types of command messages: *WriteTagData, ConfigureTagFilter*, *ExtensionCommand, Response, and KillTag*. In addition, there is a report that sends the *StatusPing* command. The optional ID attribute of the command element is not used.

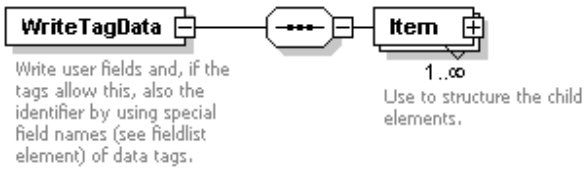
element **Command**

<table>
<tr><td>diagram</td><td> 
</td></tr>
<tr><td>children</td><td>WriteTagData ConfigureTagFilter ExtensionCommand StatusPing Response KillTag</td></tr>
<tr><td>attributes</td><td>Name | Type | Use | Annotation<br>id | xs:string | optional</td></tr>
<tr><td>annotation</td><td>documentation | A command message to the device controller</td></tr>
</table>

### 4.1.1.2 Details of the WriteTagData Element

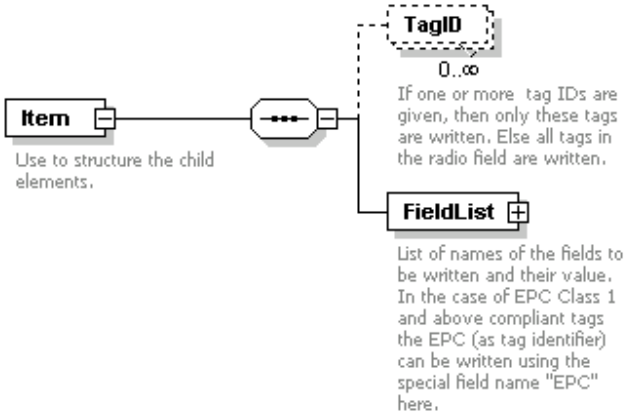
The *readerID* attribute of the *WriteTagData* element contains the RFID device ID as it is configured in SAP AII.

element **Command/WriteTagData**

| | |
|---|---|
| diagram |  |
| children | Item |
| attributes | Name: readerID; Type: xs:string; Use: optional; Annotation: documentation: Used if only a specific reader should be addressed. |
| annotation | documentation: Write user fields and, if the tags allow, write the identifier by using special field names (see fieldlist element) of data tags. |

### 4.1.1.3 Details of the WriteTagData/Item Element

element **Command/WriteTagData/Item**

| | |
|---|---|
| diagram |  |
| children | TagID FieldList |
| annotation | documentation: Use to structure the child elements. |

### 4.1.1.4 Details of the WriteTagData/Item/TagID Element

The *TagID* element is not used for tag commissioning. The ID to be written to the tag is contained in the *FieldList/Field name=EPC* (see below). The *TagID* element is only relevant for rewriting a specific tag. The *schemeID* attribute gives the SAP AII ID Version (for example, EPC_1.30).

element **Command/WriteTagData/Item/TagID**

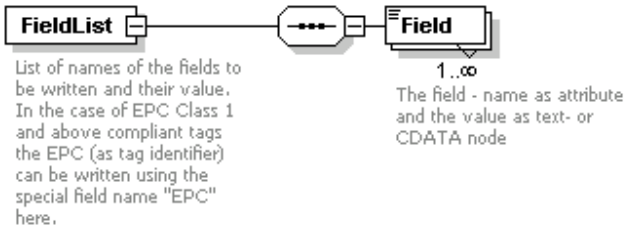
| | |
|---|---|
| diagram |  |

| | | | | |
|---|---|---|---|---|
| attributes | Name | Type | Use | Annotation |
| | schemeID | **xs:string** | optional | |
| annotation | documentation | If one or more tag IDs are given, then only these tags are written. Otherwise, all tags in the radio field are written. | | |

### 4.1.1.5 Details of the Item/FieldList Element

For the *FieldList* element, SAP AII populates the *jobName* and *quantity* attributes only if the *format* attribute is populated. The quantity value is always *1*. (If multiple tags are expected, separate item elements are provided). The *jobName* is constructed from the *readerID* and time.

element **Command/WriteTagData/Item/FieldList**

| | | | | |
|---|---|---|---|---|
| diagram |  | | | |
| children | **Field** | | | |
| attributes | Name | Type | Use | Annotation |
| | format | **xs:string** | optional | |
| | jobName | **xs:string** | optional | |
| | quantity | **xs:string** | optional | |
| | printerName | **xs:string** | optional | |
| annotation | documentation | List of names of the fields to be written and their values.<br>In the case of EPC Class 1 (and above) compliant tags, the EPC (as tag identifier) can be written using the special field name "EPC" here. | | |

### 4.1.1.6 Details of the Item/FieldList/Field Element

The device controller must support the *field name="EPC"* in order to commission an RFID tag.

Example of the Field element

```
<Field name="EPC">01002345678900FFABCD0001</Field>
```

In addition to a HEX representation of the binary EPC, configuration allows the following fields to be included in the *WriteTagData* command *FieldList*:

| Values of Field Name Attribute | Description | Max Size |
|---|---|---|
| GRAI | Global Returnable Asset Identifier (GRAI) integer with or without serial number component | 30 |
| GTIN | Global Trade Identification Number (GTIN) | 14 |
| INDV_ASSET_REFERENCE | Individual Asset Reference | 24 |
| ISSUE_AGENCY | Issuing Agency | 2 |
| ITEM_REFERENCE | Item Reference | 40 |
| PARTITION | Partition | 1 |
| PRODUCT | Product | 40 |
| PRODUCT_DESCRIPTION | Product Description | 40 |
| PRODUCT_QUANTITY | Product Quantity | 31 |
| SERIAL_NO | Serial Number | 60 |
| SERIAL_REFERENCE | Serial Reference of SSCC | 11 |
| SHIP_TO_ADDRESS | Ship-To Address | 60 |
| SHIP_TO_PARTY | Ship-To Party | 10 |
| SHIP_TO_PARTY_NAME | Ship-To Party Name | 50 |
| SSCC | Serial Shipping Container Code (SSCC) integer | 20 |
| UOM | Unit Of Measure | 3 |
| GIAI | Global Individual Asset Identifier (GIAI) EAN/UCC | 30 |
| ALT_HU_ID | Handling Unit Alternative ID (maintained only as user data) | 128 |
| ASSET_TYPE | Asset Type | 6 |
| BASE_UOM | Base Unit Of Measure | 3 |
| BUSINESS_PARTNER | Business Partner | 10 |
| BUSINESS_PARTNER_ADDRESS | Business Partner Address | 60 |
| BUSINESS_PARTNER_NAME | Business Partner Name | 50 |
| CHECK_DIGIT | Check Digit | 1 |
| COMPANY_PREFIX | Company Prefix | 20 |
| DAMAGED | Damaged (when value is “X”) | 1 |
| DOCUMENT_NO | Document Number | 35 |
| DOCUMENT_TYPE | Document Type | 2 |
| EMPTY | Empty (when value is “X”) | 1 |
| EPC | Hexadecimal EPC | 96 |
| EPC_TYPE | Encoding Type | 10 |
| EPC_URN | EPC in URN format | 128 |
| EPC_URN_NO_HEADER | EPC in URN format but without header | 128 |
| FILTER_VALUE | Filter Value | 1 |
| BC_UII_CONSTRUCT_1 | UII construct #1 in a barcode string, format 06, ISO/15434 | 4096 |
| BC_UII_CONSTRUCT_2_PART | UII construct #2 in a barcode string, format 06, ISO/15434 | 4096 |
| <user defined element> | <any> | 128 |

element **Command/WriteTagData/Item/FieldList/Field**

| | | | | | |
|---|---|---|---|---|---|
| diagram | Field<br>The field - name as attribute and the value as text- or CDATA node | | | | |
| attributes | Name | Type | Use | Annotation | |
| | name | **xs:string** | required | documentation | Name of the data field on the tag |
| annotation | documentation | The field - name as attribute and the value as text- or CDATA node | | | |

### 4.1.1.7 Details of the ConfigureTagFilter Element

The *ConfigureTagFilter* command allows SAP AII to update a logical subscription model on the device controller. The three objects in the subscription model are logical device, filtering command, and rule. The device controller is obliged to notify SAP AII of any logical device observations that satisfy the rules in the filtering command. All observations are sent if the list of rules is empty.

Logical Device Attributes:

- Name (SAP AII device ID and *ConfigureTagFilter* attribute *readerID*)
- URL (SAP AII notification url and *ConfigureTagFilter* attribute *notificationURL)*
- List of filtering command names (0 or more)

Filtering Command Attributes:

- Name (notification message command value and *FilteringCommand* attribute name)
- List of rule names (0 or more)

Rule Attributes:

- Name (*FilteringRules*/Rule attribute name)
- Type (*FilteringRules*/Rule attribute ruleType =*"inclusive"*)
- One of the following three:
  - Filter (*TagMask* attributes mask and value)
  - List of tag IDs (0 or more Rule/Tags/Tag attribute TagID)
  - Range of tag IDs (Rule/Range attributes *StartTagID* and *EndTagID*)

The **notificationURL** attribute of the *ConfigureTagFilter* element:

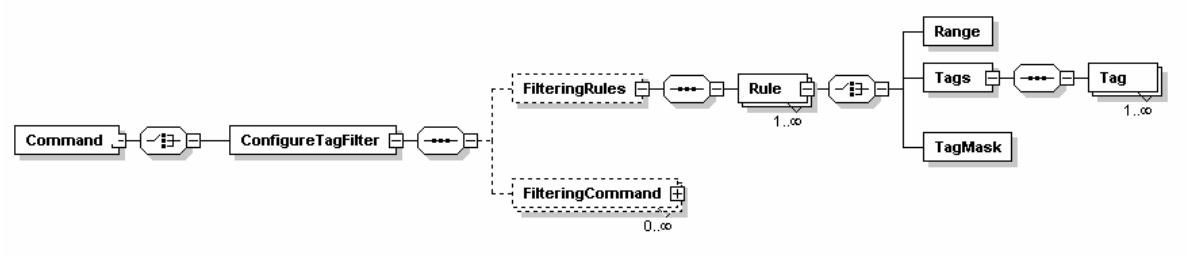
Contains the url of the SAP AII notification message processor (SAP service/sap/scm/ain). This url has the form:

http://<SAP AII system IP address> :< port>/sap/scm/ain/sap-client=<SAP AII client number>

The **readerID** attribute of the *ConfigureTagFilter* element:

Contains the RFID device ID, as configured in SAP AII

element **Command/ConfigureTagFilter**




### 4.1.1.8 Details of the FilteringRules Element

The optional *FilteringRules* element is usually a list of rule definitions; however, use of the optional *action* attribute with a value *"delete"* is a request to delete all previously defined rules for this notificationURL. No rule elements are required for this "delete" option. An action value of *"add"* may be provided, but it has no meaning.

### 4.1.1.9 Details of the FilteringRules/Rule Element

The *action* attribute of the *Rule* element contains *"add"* or *"delete."* This applies to the named rule and its subordinate Range, Tags or *TagMask* element. The value *"add"* makes this rule available to references by the current and future *ConfigureTagFilter* commands. A value of *"delete"* is a request to remove the named rule from the device controller (unless it is referenced by a Filtering Command).

When the *ruleType* attribute of the *Rule* element is *"inclusive*," tags that match this rule should be included.

The *name* attribute of the *Rule* element contains a unique identifier that can be referenced in the *refid* attribute of the *FilteringCommand*/Rule.

### 4.1.1.10 Details of the FilteringRules/Rule/TagMask Element

The *mask* attribute of the *TagMask* element indicates which bits in the tag ID must match the content of the *value* attribute. Both attributes are HEX representations of binary patterns and are the length of the tag ID. A binary one in the mask indicates that the corresponding bit in the tag ID must match the corresponding *value* bit. A zero in the *mask* means that the corresponding bit in tag ID can be either zero or one.

### 4.1.1.11 Details of the FilteringCommand Element

The optional *action* attribute of the *FilteringCommand* element contains *"add," "edit"* or *"delete."* A value of *"add"* creates a named *FilteringCommand* that can be referenced by future *ConfigureTagFilter* commands. A value of *"delete"* is a request to remove the named *FilteringCommand* and, thereby, cancel the subscription it described. A value of *"edit"* is used when the list of rules in an existing *FilteringCommand* is changed.

The *name* attribute of the *FilteringCommand* element contains the value of the *Command* element that should be used in all notification messages that are sent in response to the subscription.

### 4.1.1.12 Details of the FilteringCommand/Rule Element

The *action* attribute of the *FilteringCommand/Rule* element contains *"add"* or *"delete."* The *"add"* value limits the subscription according to the named *FilteringRule* rule (refid). The value *"delete*" is a request to remove the reference to this filter rule from the definition of the *FilteringCommand.* (It does not delete the filtering rule itself).

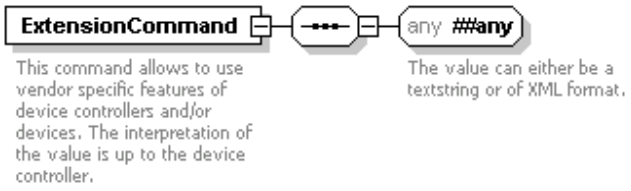
The *refid* attribute of the *FilteringCommand*/Rule element contains the name of a rule defined in the *FilteringRules* element.

### 4.1.1.13 Details of the Command/ExtensionCommand Element

The *ExtensionCommand* element allows SAP AII to use special features of the device controller such as control of a light tower. The *readerID* attribute of the *ExtensionCommand* element should be used by the device controller to delegate the command to the specified device.

The value of the *ExtensionCommand* element must be of type string or XML.

element **Command/ExtensionCommand**

| | | | | |
|---|---|---|---|---|
| diagram |  | | | |
| attributes | Name | Type | Use | Annotation |
| | readerID | **xs:string** | optional | |
| annotation | documentation | This command allows the use of device-specific features. The interpretation of the value is up to the device controller. | | |

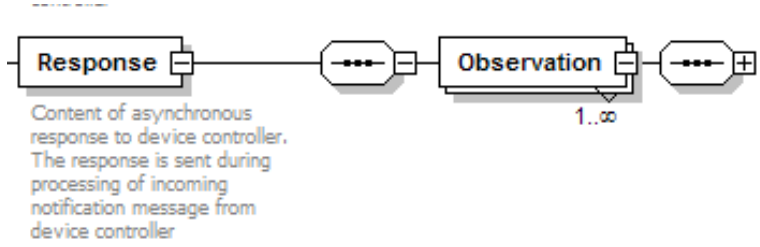
### 4.1.1.14 Details of the Command/StatusPing Element

The *StatusPing* element allows SAP AII to ping a device controller. There is a report that can be used to send this command. If SAP AII does not receive an http OK (response code 200) from the device controller an Alert is logged.

### 4.1.1.15 Details of the Command/Response Element

The Response element allows SAP AII to send an asynchronous response to the device controller during processing of an incoming notification message from the device controller. This is the reason why the structure of the Observation child element of the Response element almost parallels the Observation child element of the Sensor element in the notification message.

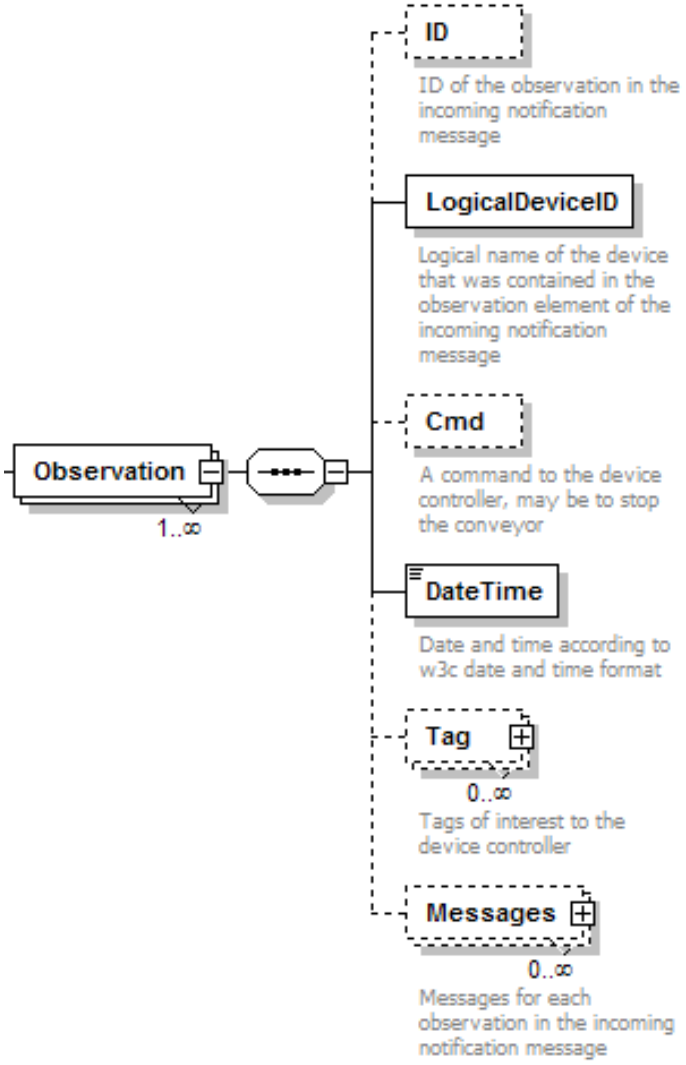
SAP AII could be configured to send error, warning or information messages generated during the processing of a notification message back to the device controller. Customer could write their own activity to send messages specific to each tag ID, for example, if a particular tag ID is invalid.

| | |
|---|---|
| diagram |  |
| children | **Observation** |
| attributes | Name Type Use Annotation |
| annotation | documentation Content of asynchronous response to device controller. The response is sent during processing of incoming notification message from device controller |

### 4.1.1.16 Details of the Response/Observation Element

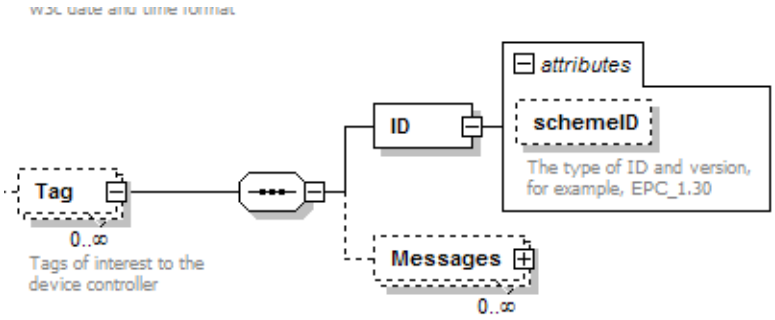
The Observation element has the following children

- ID – If the Observation element of the incoming notification message has an ID, then this field will be populated with the same value. This helps the device controller to identify for which notification message the response is sent
- LogicalDeviceID – This contains the SAP AII Device Group ID. If the observation in the incoming notification message contains the device group ID then the same ID is sent back. If the observation contains device ID then the ID of the device group to which the device belong, is sent back.
- Cmd – This could be a command to the device controller to take certain action. Customer need to write their own activity in order to populate it.
- DateTime – Date and time when the response is sent. This element has time in w3c date time format; se definition at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime.
- Tag – List of tag IDs that could be of interest to the device controller. Customer has to write their own activity in order to populate it
- Messages – List of messages that were generated during notification message processing. Through configuration one could control the severity of the messages that would be sent back. The Messages element is of type MessagesType which is described below

| | |
|---|---|
| diagram |  |
| children | **ID LogicalDeviceID Cmd DateTime Tag Messages** |
| attributes | Name Type Use Annotation |
| annotation | documentation |

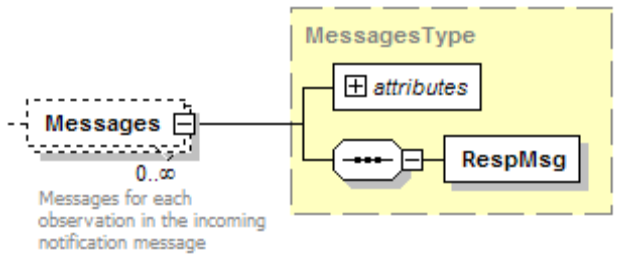
### 4.1.1.17 Details of the Observation/Tag Element

The Tag element contains the ID of the Tag and list of messages specific to the tag. The ID of the tag has an attribute schemeID which contains the ID type and version of the ID, for example, EPC_1.30. The Messages element is of type MessagesType which is described below.

| | |
|---|---|
| diagram |  |
| children | **ID Messages** |
| attributes | Name Type Use Annotation |
| annotation | documentation Tags of interest to the device controller |

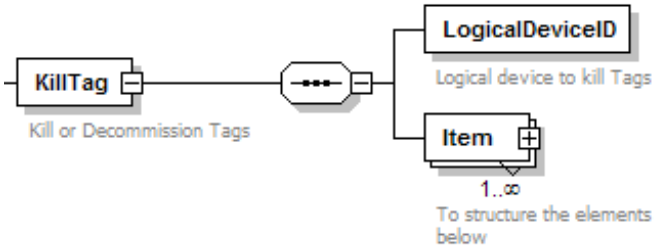
### 4.1.1.18 Details of complex type MessagesType

The *Messages* element is of complex type MessagesType and contains an attribute RespCode which gives the severity of the message. The values are ‘E’, ‘W” and ‘I’.

| | |
|---|---|
| diagram |  |
| children | **RespMsg** |
| attributes | Name Type Use Annotation<br>RespCode **Xs:string** Optional |
| annotation | documentation |

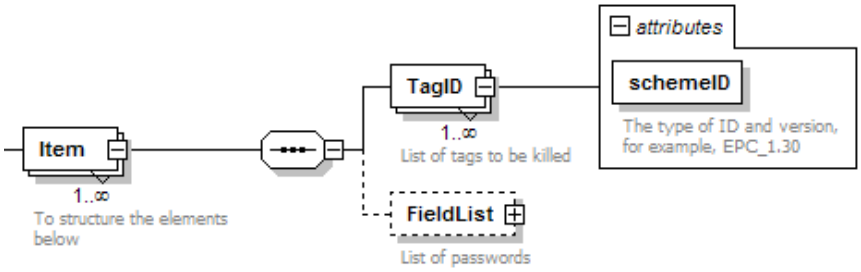
### 4.1.1.19 Details of the Command/KillTag Element

The KillTag element contains the Logical Device to be used to kill the tag and a list of tags and their password. The KillTag element has two children: LogicalDeviceID and Item. LogicalDeviceID element contains a SAP AII Device Group ID. The Item element contains tag and password.

| | |
|---|---|
| diagram |  |
| children | **LogicalDeviceID Item** |
| attributes | Name Type Use Annotation |
| annotation | documentation Kill or Decommission Tags |

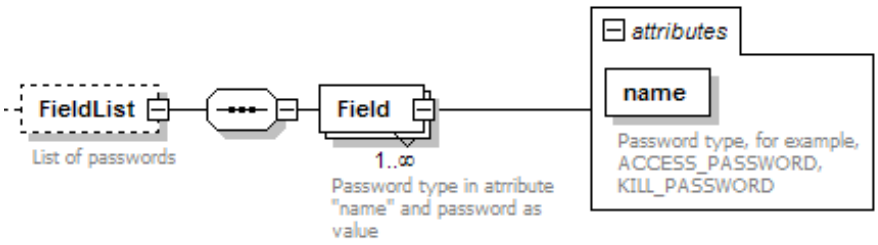
### 4.1.1.20 Details of the KillTag/Item Element

The Item element contains a list of tags and passwords for those tags. All the tags in an Item have the same passwords. Each TagID child element of the Item element contains the ID of a tag, and an attribute schemeID that specifies the ID type and version. The Fieldlist child element contains the list of passwords.

| | |
|---|---|
| diagram |  |
| children | **TagID FieldList** |
| attributes | Name Type Use Annotation |
| annotation | documentation To structure the elements below |

### 4.1.1.21 Details of the Item/FieldList element

The FieldList element has a set of Field child element. Each Field element contains a password and has an attribute that specifies the password type. SAP AII supports two password types: ACCESS_PASSWORD or KILL_PASSWORD. Customer can add their own password types through configuration and writing an activity.

| | |
|---|---|
| diagram |  |
| children | **Field** |
| attributes | Name Type Use Annotation |
| annotation | documentation List of passwords |

## 4.1.2 Samples of the WriteTagData Command

The *WriteTagData* command message is sent by SAP AII to instruct the device controller to write the ID on an RFID tag. In SAP AII, this is called *tag commissioning*. Optionally, *WriteTagData* supports RFID printers that can print information on a label and commission the embedded RFID tag. The predefined SAP AII Activity, DEVICE_PRINT_TAG, sends this *WriteTagData* command.

### 4.1.2.1 RFID Tag Commissioning

After sensing the presence of a product case, SAP AII sends a *WriteTagData* command to the device controller requesting that the Writer_Device write a specific ID to the RFID tag embedded in the corrugate of the case.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Command.xsd">
    <WriteTagData readerID="Writer_Device">
        <Item>
            <FieldList>
                <Field name="EPC">3074024220403B8000000008</Field>
            </FieldList>
        </Item>
    </WriteTagData>
</Command>
```

### 4.1.2.2 RFID Label Printing

After the cases on a pallet are stretch wrapped, SAP AII sends a *WriteTagData* command to the device controller requesting that a pallet label be printed and a specific ID be written to the RFID tag in the label. The command message includes additional information fields and field list attributes used by the printer (Writer_Device).

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Command.xsd">
    <WriteTagData readerID="Writer_Device">
        <Item>
            <FieldList format="C:LABEL.PL"
                    jobName="Writer_Device20040929165746"
                    quantity="1">
                <Field name="EPC">3074024220403B8000000008</Field>
                <Field name="EPC_TYPE">SGTIN-96</Field>
                <Field name="EPC_URN">
                    urn:autoid:tag:sgtin-96:3.5.0037000.065774.8</Field>
                <Field name="PRODUCT">SGPROD</Field>
                <Field name="PRODUCT_DESCRIPTION">Test product</Field>
            </FieldList>
        </Item>
    </WriteTagData>
</Command>
```

The sequence of events leading to sending the *WriteTagData* command (step 5) is given below:

1. SAP AII provides a master list of variable names, including “EPC”, which can be used during label design. (The values of these variables can be provided in the *WriteTagData* command sent by SAP AII).
2. The label designer creates a named format with tools provided by the printer partner. The content of the format directs the printer to print a label that includes the value of several variables from the master list. The format also directs the printer to write the value of the “EPC” variable to the RFID tag.
3. An RFID printer is connected to the TCP/IP network and given a specific IP address. The printer listens for print commands on an HTTP server. The printer must have access to the format created in step 2.
4. SAP AII is configured with the address and port of the RFID printer, the name of the printing format, and the associated list of variable names.

5. SAP AII selects the printer and printer format and evaluates the associated data variables, including a unique ID value. A *WriteTagData* command message is sent to the printer.
6. The RFID printer prints a label and writes the ID to the RFID tag embedded in the label.

#### 4.1.2.3 RFID Tag Updating

During preparation for delivery, the delivery document number is added to the previously tagged case. The command message includes *TagID* to be updated and field list attributes to be written by the RFID device (Writer_Device).

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Command.xsd">
    <WriteTagData readerID="Writer_Device">
        <Item>
            <TagID schemeID="EPC_1.30">30740242205C35C00000000B</TagID>
            <FieldList>
                <Field name="DOCUMENT_NO">123456</Field>
            </FieldList>
        </Item>
    </WriteTagData>
</Command>
```

### 4.1.3 Sample of the ConfigureTagFilter Command

The *ConfigureTagFilter* command message is sent by SAP AII to the device controller to describe the content and address of notification messages.

SAP AII can be configured to send this message to establish static subscriptions. Custom activities can also use this message to dynamically change subscriptions.

#### 4.1.3.1 Subscribing to SAP AII Tag Observations

A device at the loading door is expected to send notification messages to SAP AII for all RFID tag observations. The following command tells the device that notification messages should be posted to the notificationURL. The value of the *Command* element in each notification message should be the *FilteringCommand* name *"IN"*.

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

```
    <ConfigureTagFilter notificationURL="http://aii-
        system:50019/sap/scm/ain?sap-client=001" readerID="DOOR_1_RID">
        <FilteringRules action="add" />
        <FilteringCommand action="add" name="IN">
      </FilteringCommand>
    </ConfigureTagFilter>
</Command>
```

### 4.1.3.2 Subscribing to Selected Tag Observations

At a loading door, only SSCC and GRAI pallet tags need be reported. SAP AII configures the fixed device at the loading door using the following message.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Command.xsd">
    <ConfigureTagFilter notificationURL="http://aii-
        system:50019/sap/scm/ain?sap-client=001" readerID="DOOR_1_RID">
      <FilteringRules action="add">
          <Rule action="add" name="SSCC-96" ruleType="inclusive">
            <TagMask mask="FF00000000000000007FFFFF"
                value="3100000000000000000000000" />
          </Rule>
          <Rule action="add" name="GRAI-96" ruleType="inclusive">
            <TagMask mask="FF80000000000000000000000"
              value="3300000000000000000000000" />
          </Rule>
      </FilteringRules>
      <FilteringCommand action="add" name="IN">
          <Rule action="add" refid="SSCC-96" />
          <Rule action="add" refid="GRAI-96" />
      </FilteringCommand>
    </ConfigureTagFilter>
</Command>
```

### 4.1.3.3 Canceling Tag Observation Subscriptions

A device at the loading door is expected to stop sending previously requested notification messages with command value *"IN"* to SAP AII. The following command tells the device that the subscription to

*FilteringCommand "IN"* should be cancelled. Any unused rules should be deleted, along with the *"IN" FilteringCommand*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Command.xsd">
    <ConfigureTagFilter notificationURL="http://aii-
        system:50019/sap/scm/ain?sap-client=001" readerID="DOOR_1_RID">
        <FilteringRules action="delete" />
        <FilteringCommand action="delete" name="IN" />
    </ConfigureTagFilter>
</Command>
```

## 4.1.4 Sample of the ExtensionCommand

The predefined SAP AII activity, DEVICE_SEND_LIGHT, uses the *ExtensionCommand* message to tell the device controller to change the color of a light tower (device MY_READER) to RED or GREEN.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="Command.xsd">
        <ExtensionCommand readerID="MY_READER">GREEN</ExtensionCommand>
</Command>
```

## 4.1.5 Sample of the StatusPing command

The report /AIN/STATUS_PING send the ping command.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="Command.xsd">
        <StatusPing />
</Command>
```

## 4.1.6 Sample of the Response command

There are two predefined SAP AII activities: RESPONSE_MSG_SEND_DEV_PRE, RESPONSE_MSG_SEND_DEV. If these two activities are included at the end of a SAP AII rule in the order mentioned above then all the messages returned by all previous activities in the rule will be

returned to the device controller. You can configure the rule to control that only messages above a certain severity level be send to the device controller. There is also a predefined exception activity, RESPONSE_MSG_SEND_DEV_EXC which could be specified as an exception activity to a standard activity in a rule. When the standard activity throws an exception, then the exception activity returns the message from the standard activity to the device controller. Below is a sample of such a message. This message was generated because someone tried to commission a tag that was already there in the SAP AII database

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="Command.xsd">
    <Response>
        <Observation>
            <LogicalDeviceID>GMC-DG-SCAN</LogicalDeviceID>
            <DateTime>2007-06-14T20:14:56Z</DateTime>
            <Messages RespCode="E">
                <RespMsg>
                    Physical object with ID '31140242240000041C000000' (EPC_1.30) already
                exists
                </RespMsg>
            </Messages>
        </Observation>
    </Response>
</Command>
```

## 4.1.7 Sample of the KillTag command

There are two predefined SAP AII activities: TAG_SENDKILL_DEV_PRE, TAG_SENDKILL_DEV which have to be included in a rule in that order for SAP AII to be able to send a kill command. TAG_SENDKILL_DEV activity composes and sends the message. The other activity prepares the data and gets the passwords if they have been stored for the object.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="Command.xsd">
        <KillTag>
            <LogicalDeviceID>GMC-DG-SCAN</LogicalDeviceID>
            <Item>
                <TagID schemeID="EPC_1.30">
```

```
            303402422027314000000014
        </TagID>
        <TagID schemeID="EPC_1.30">
            303402422027314000000015
        </TagID>
        <FieldList>
            <Field name="ACCESS_PASSWORD">GTIN00371</Field>
            <Field name="KILL_PASSWORD">GTIN0037KILL</Field>
        </FieldList>
    </Item>
    <Item>
        <TagID schemeID="EPC_1.30">
            313402422000001107000000
        </TagID>
        <FieldList>
            <Field name="ACCESS_PASSWORD">SSCC00371</Field>
            <Field name="KILL_PASSWORD">SSCC0037KILL</Field>
        </FieldList>
    </Item>
  </KillTag>
</Command>
```

## 4.2 Notification Message

The notification messages sent by the device controller are processed in SAP AII as fixed reader messages. SAP AII configuration specifies the conditions used to route fixed reader messages to processing rules.

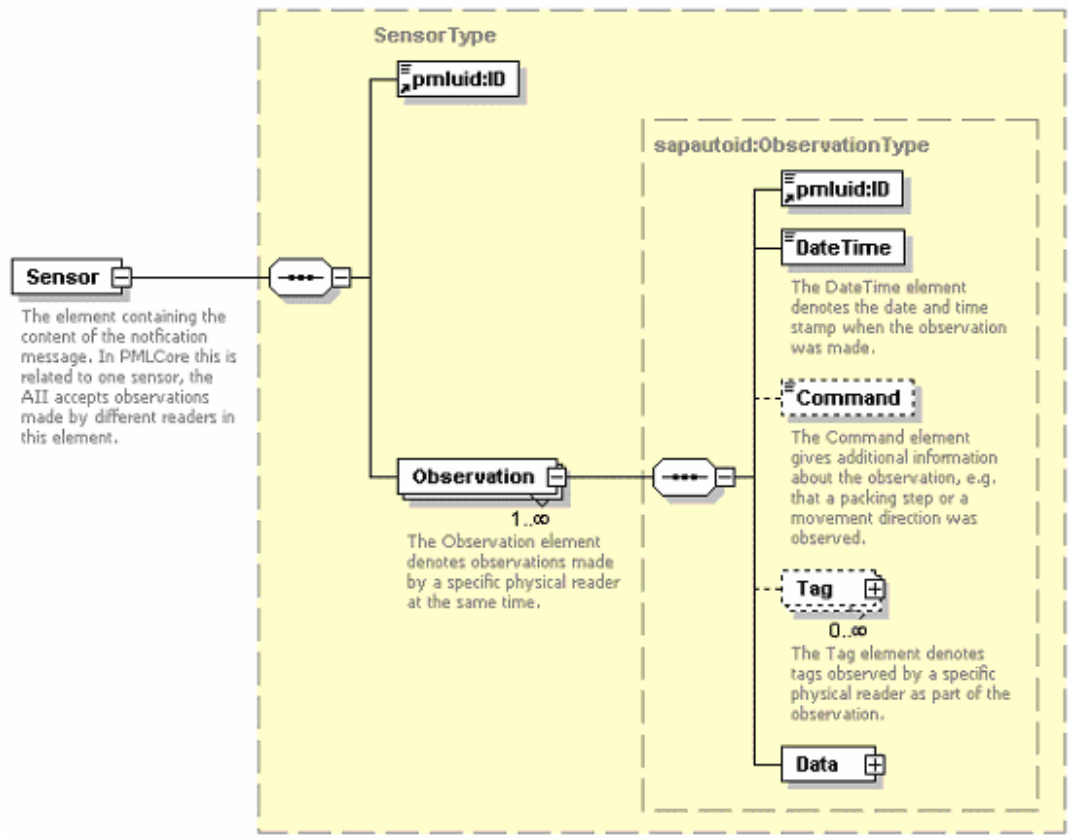
### 4.2.1 Elements of the Notification Message

The notification message in SAP AII-DC 4.0 is an enhancement of the sensor message described in EPCglobal's *PML Core Specification 1.0*. The PML specification, including an XML schema, can be found at the following link:

http://www.epcglobalinc.org/standards_technology/Secure/v1.0/PML_Core_Specification_v1.0.pdf

Exceptions and extensions used by SAP AII are noted in the sections below.

The following figure shows an overview of the complete notification message.

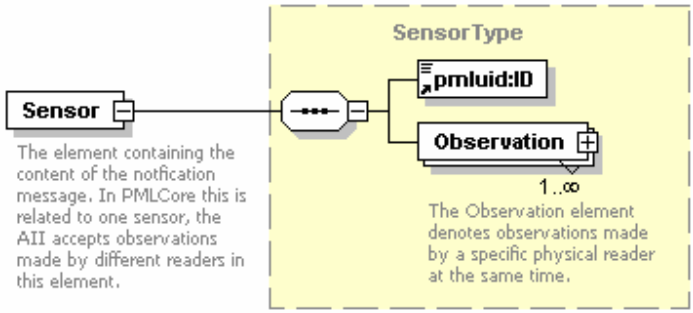



### 4.2.1.1 Details of the Sensor Element

All elements of the notification are included in the *Sensor* element. The *pmluid:ID* element, which is the first child element of the *Sensor* element, identifies the device controller that sent the notification message.

A notification message may contain one or more observation elements, each from a different RFID device. Each observation is processed by the appropriate SAP AII rule logic; however, processing of the entire notification message is one logical transaction. SAP AII will commit all observations or none to the database.

element **Sensor**

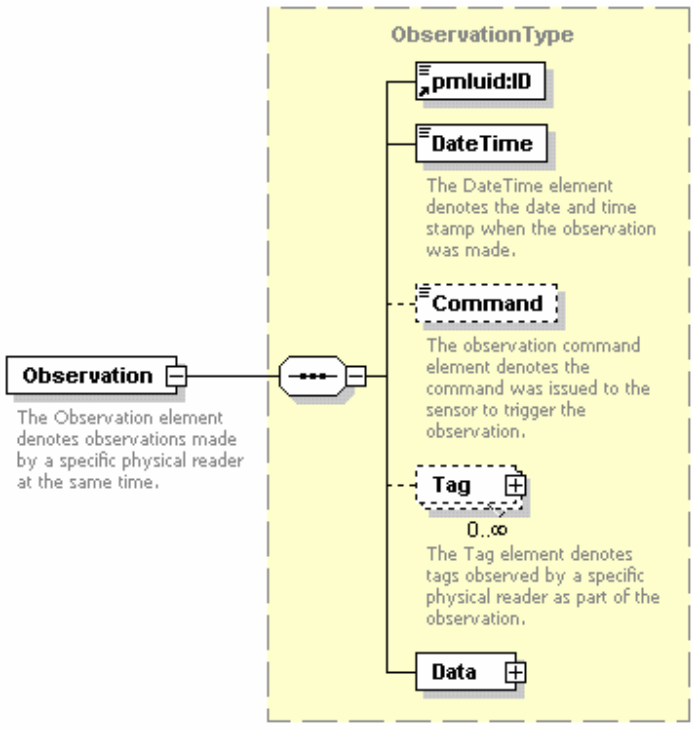
| | |
|---|---|
| diagram |  |
| children | pmluid:ID Observation |
| annotation | documentation: The element containing the content of the notification message. In PMLCore this is related to one sensor. SAP AII accepts observations made by different readers in this element. |

### 4.2.1.2 Details of the Observation Element

The *pmluid:ID* element that identifies the observation is optional in SAP AII.

The *Command* element is required by SAP AII because it is used to select the message processing logic. There is a further description in the Observation/Command section below.

element **Sensor/Observation**

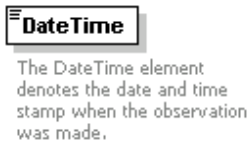
| | |
|---|---|
| diagram |  |

| | |
|---|---|
| children | **pmluid:ID DateTime Command Tag Data** |
| annotation | documentation: The *Observation* element denotes observations made at the same time by a specific reader. |

### 4.2.1.3 Details of the Observation/DateTime Element

The type *"dateTime"* is a standard XML data type. For more information, see the definition, at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime.

element **Observation/DateTime**

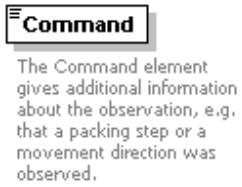
| | |
|---|---|
| diagram |  |
| type | **dateTime** |
| properties | isRef 0<br>content simple |
| annotation | documentation: The *DateTime* element denotes the date and time stamp that registers when the observation was made. |

### 4.2.1.4 Details of the Observation/Command Element

The values of the command can be specified by the partner; there are no predefined values in SAP AII. Typically, command values indicate the action associated with the observation. Examples include:

- Direction of movement (IN or OUT)
- Business action (PACK, UNPACK)

element **Observation/Command**

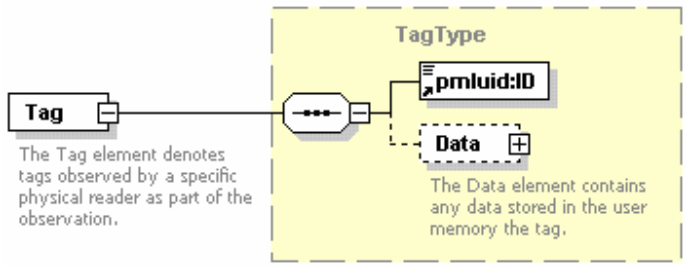
| | |
|---|---|
| diagram |  |
| type | **string** |
| annotation | documentation: The *Command* element gives additional information about the observation, for example, that a packing step or a movement direction was observed. |

### 4.2.1.5 Details of the Observation/Tag Element

The *pmluid:ID* element is the unique identifier of the RFID tag. This element is omitted from observations of objects without RFID tags.

An optional attribute of the pmluid:ID is *schemeID*. It is used to specify the SAP AII ID version (for example, EPC_1.27) that should be used to decode the ID.

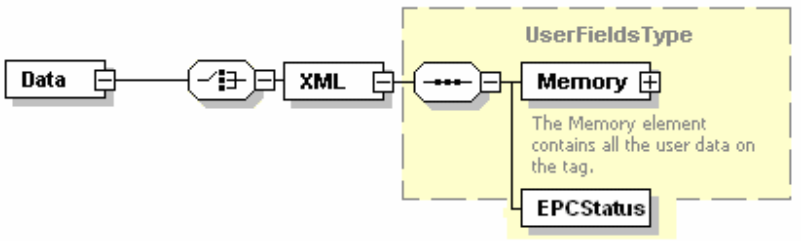
element **Observation/Tag**

| | |
|---|---|
| diagram |  |
| children | **pmluid:ID Data** |
| annotation | documentation The *Tag* element denotes tags observed by a specific physical reader as part of the observation. |

## 4.2.1.6 Details of the Observation/Tag/Data/XML Element

*EPCStatus* is a child element of the *Tag/Data/XML* element used in a predefined SAP AII activity. The value of EPCStatus can indicate that a tag was commissioned successfully. The *Memory* element can contain a list of Data Field elements that hold tag data values.

element **Observation/Tag /Data/XML**

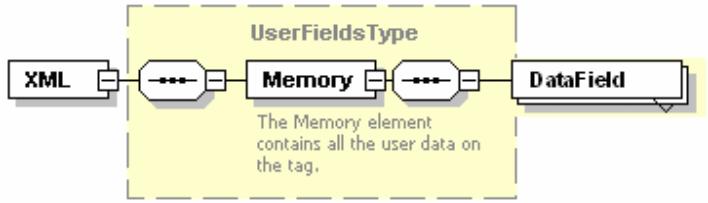
| | |
|---|---|
| Diagram |  |

| | |
|---|---|
| children | **EPCStatus Memory** |

## 4.2.1.7 Details of the Observation/Tag/Data/XML/Memory/DataField Element

The *fieldName* attribute of the *DataField* element identifies the element's tag data value. The names must be included in the element set SAP_ALL_FIELDS.

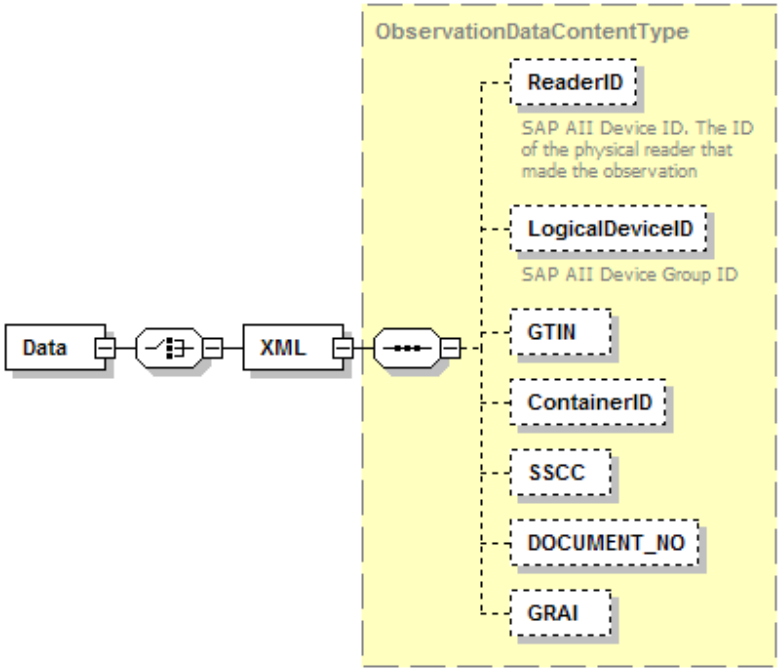
element **Observation/Tag /Data/XML/Memory/DataField**

| | |
|---|---|
| diagram |  |
| attributes | Name: fieldName; Type: **xs:string**; Use: required; Annotation: |

## 4.2.1.8 Details of the Observation/Data/XML Element

The *Observation/Data/XML* element has optional child elements that are used by SAP AII. The *ReaderID* element contains the SAP AII RFID device ID that made the observation. The *LogicalDeviceID* contain the SAP AII device group ID. In AII one or more devices could belong to a device group. Either *ReaderID* or *LogicalDeviceID* must be specified. If both are specified then the device must belong to the device group in AII. The *GTIN*, *GRAI*, and *SSCC* elements are used by SAP AII to capture data from non-RFID observations. *ContainerID* can be used to explicitly identify the pallet associated with a list of case observations. *DOCUMENT_NO* can be used to associate an observation with a particular SAP AII document.

element **Observation/Data /XML**

| | |
|---|---|
| diagram |  |
| children | **ReaderID LogicalDeviceID GTIN SSCC GRAI ContainerID DOCUMENT_NO** |

## 4.2.2 Samples of RFID Tag Notification Messages

### 4.2.2.1 Single Tag

One scenario for the notification message is reading a single RFID tag and sending the ID to SAP AII. A sample XML shows how the device controller sends the message to SAP AII. This message might be configured to trigger a verify, pack, move, or load action.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <pmlcore:Sensor
     xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
     xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
     <pmlcore:Observation>
      <pmlcore:DateTime>2006-02-04T01:30:00.762-08:00</pmlcore:DateTime>
      <pmlcore:Command>IN</pmlcore:Command>
        <pmlcore:Tag>
          <pmluid:ID>30740242205C35C00000000A</pmluid:ID>
      </pmlcore:Tag>
        <pmlcore:Data>
          <pmlcore:XML>
            <ReaderID>PACKING_DEVICE_NAME</ReaderID>
        </pmlcore:XML>
      </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

With the *Tag Verify (TAGVERIFY)* rule, this message can be used to verify that a tag commissioned by SAP AII can be read.

With the *Conveyor Packing – Inner Item Reading (CPACK)* rule, this message can be used to report the observation of a case that is being packed onto a pallet.

In the move scenario, this message can report the location of a pallet and, by implication, the associated cases. The location is determined from the location assigned to the RFID Device in SAP AII.

With the *LOAD* rule, this message can confirm the loading of a pallet and, by implication, the associated cases.

### 4.2.2.2 Multiple Tags in One Observation

Observations of more than one tag can be used in all the single tag scenarios above except conveyor packing. In addition, multi-tag observations can enable enhanced pack and load logic.

With the *PACK* rule, this message can be used to report the observation of a pallet and the cases that are being packed. If the pallet is not explicitly identified (see the Smart Packing example), then the rule logic relies on the tag's *"EPC Filter"* value to determine which object is the pallet. Only one pallet can be included in the list of tags in a valid pack observation. All other tags are assumed to be cases.

In the load scenario, this message can confirm the loading of a pallet, together with the associated cases. Some configurations require that only some of the cases be observed before a loading can be confirmed.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <pmlcore:Sensor
    xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
    xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
   <pmlcore:Observation>
    <pmlcore:DateTime>2006-02-03T02:00:00.762+01:00</pmlcore:DateTime>
    <pmlcore:Command>IN</pmlcore:Command>
     <pmlcore:Tag>
      <pmluid:ID>31540242200000004500000O</pmluid:ID>
     </pmlcore:Tag>
     <pmlcore:Tag>
      <pmluid:ID>30740242204031C0000003E7</pmluid:ID>
     </pmlcore:Tag>
     <pmlcore:Tag>
      <pmluid:ID>30740242204031400097999 9</pmluid:ID>
     </pmlcore:Tag>
     <pmlcore:Data>
       <pmlcore:XML>
        <ReaderID>PACKING_DEVICE_NAME</ReaderID>
       </pmlcore:XML>
     </pmlcore:Data>
   </pmlcore:Observation>
</pmlcore:Sensor>
```

### 4.2.2.3 Smart Packing with Tag Data

This notification message reports three tags. The first two have tag data (batch and expiration date) and the third is identified as the container. This message also provides examples of:

- schemeID set to EPC_1.24 and EPC_1.27
- DOCUMENT_NO

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<pmlcore:Sensor
    xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1
    " xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <pmluid:ID>GC_DC</pmluid:ID>
   <pmlcore:Observation>
      <pmlcore:DateTime>2005-02-09T16:20:54.843+01:00</pmlcore:DateTime>
      <pmlcore:Command>IN</pmlcore:Command>
      <pmlcore:Tag>
         <pmluid:ID
            schemeID="EPC_1.24">30740242204031C0000003E8</pmluid:ID>
            <pmlcore:Data>
               <pmlcore:XML>
                  <Memory>
                     <DataField fieldName="ZExpirationDate">2012-11-
                        06T13:04:34.86-06:00</DataField>
                     <DataField fieldName="ZBatchNo">1234</DataField>
                  </Memory>
               </pmlcore:XML>
            </pmlcore:Data>
       </pmlcore:Tag>
      <pmlcore:Tag>
         <pmluid:ID schemeID="EPC_1.27">98002186B8000018</pmluid:ID>
         <pmlcore:Data>
            <pmlcore:XML>
               <Memory>
                  <DataField fieldName="ZExpirationDate">2010-11-
                     06T13:04:34.86-06:00</DataField>
                  <DataField fieldName="ZBatchNo">1235</DataField>
               </Memory>
            </pmlcore:XML>
         </pmlcore:Data>
      </pmlcore:Tag>
      <pmlcore:Tag>
         <pmluid:ID schemeID="EPC_1.24">0880008000000E74</pmluid:ID>
      </pmlcore:Tag>
      <pmlcore:Data>
         <pmlcore:XML>
            <ReaderID>GC_PACK</ReaderID>
            <ContainerID>0880008000000E74</ContainerID>
            <DOCUMENT_NO>801568</DOCUMENT_NO>
```

```
            </pmlcore:XML>
          </pmlcore:Data>
        </pmlcore:Observation>
    </pmlcore:Sensor>
```

### 4.2.2.4 Verification of External Tag Commissioning

This process reports that a tag was commissioned and indicates whether the tag was successfully read. The SAP AII rule *Tag Verify* (*TAGVERIFY)* handles notification messages that include the *EPCStatus* element. The following sample XML shows how the device controller sends the message to SAP AII.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <pmlcore:Sensor
     xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
     xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
     <pmlcore:Observation>
      <pmlcore:DateTime>2004-11-12T02:00:00.762+01:00</pmlcore:DateTime>
      <pmlcore:Command>IN</pmlcore:Command>
        <pmlcore:Tag>
         <pmluid:ID>A003006078000020</pmluid:ID>
         <pmlcore:Data>
             <pmlcore:XML>
              <EPCStatus>WS</EPCStatus>
            </pmlcore:XML>
          </pmlcore:Data>
      </pmlcore:Tag>
        <pmlcore:Data>
          <pmlcore:XML>
            <ReaderID>VERIFY_DEVICE_NAME</ReaderID>
          </pmlcore:XML>
      </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

The element *EPCStatus* may have following values:

- *'WS'* to indicate that the tag was written and verified successfully
- *'WU'* to indicate that the tag was written, but not verified successfully

## 4.2.3 Samples of Other Observations (Barcodes, Light Sensors, Etc.)

The notification message can be used to report events other than reading an RFID tag. In the following examples, such events are used by SAP AII to trigger tag commissioning. These might be used on a conveyer. The notification message is sent when a case is detected. SAP AII rule *Tag Commissioning (TAGCOM)* responds by sending a tag commission command *WriteTagData* to the RFID device that is downstream on the conveyer.

### 4.2.3.1 Reading a GTIN (from a Barcode) to Trigger Tag Commissioning

A fixed bar code reader reads the GTIN on a case. The device controller sends the following XML to SAP AII. This notification message triggers the TAGCOM rule which uses the GTIN to generate an SGTIN EPC and sends the tag commission command *WriteTagData t*o an RFID device.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <pmlcore:Sensor
     xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
     xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
     <pmlcore:Observation>
      <pmlcore:DateTime>2004-11-01T13:04:33.050+06:00</pmlcore:DateTime>
      <pmlcore:Command>PRNT</pmlcore:Command>
        <pmlcore:Data>
          <pmlcore:XML>
            <ReaderID>READER_DEVICE_NAME</ReaderID>
            <GTIN>00037000567394</GTIN>
         </pmlcore:XML>
       </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

### 4.2.3.2 Reading an SSCC to Trigger Printing

A fixed bar code reader reads the SSCC on a pallet. The device controller sends the following XML to SAP AII. This notification message triggers the *TAGCOM* rule which uses the SSCC to generate an SSCC EPC and send the tag commission command *WriteTagData* to an RFID device. GRAIs can also be commissioned with this approach.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <pmlcore:Sensor
     xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
     xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
```

```
    <pmlcore:Observation>
      <pmlcore:DateTime>2002-11-06T13:04:33.050-06:00</pmlcore:DateTime>
      <pmlcore:Command>PRNT</pmlcore:Command>
          <pmlcore:Data>
          <pmlcore:XML>
            <ReaderID>READER_DEVICE_NAME</ReaderID>
            <SSCC>000370000000000266</SSCC>
        </pmlcore:XML>
      </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

### 4.2.3.3 Sending a Blank Message to Trigger Tag Commissioning

A light sensor detects a case on the conveyer. The device controller sends the following XML message to SAP AII. This notification message triggers the *TAGCOM* rule which generates an ID and sends the tag commission command *WriteTagData* to an RFID device.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <pmlcore:Sensor
     xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
     xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
      <pmlcore:Observation>
        <pmlcore:DateTime>2002-11-01T13:04:33.05-06:00</pmlcore:DateTime>
        <pmlcore:Command>PRNT</pmlcore:Command>
            <pmlcore:Data>
            <pmlcore:XML>
              <ReaderID>READER_DEVICE_NAME</ReaderID>
          </pmlcore:XML>
        </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

# 5 Communication Protocol

SAP AII uses a standard SOAP binding to the ALE Web Services, as specified in EPCglobal's ALE 1.0.

The format of all messages is XML. The header of the XML message identifies the encoding, for example, UTF-8. SAP AII sends messages with UTF-8 encoding and expects incoming messages to have UTF-8 encoding.

The interface uses the HTTP 1.1 protocol (RFC 2.6.1.6, http://www.w3.org/protocol) for the messages from SAP AII to the device controller and from the device controller to SAP AII. Partners can deliver HTTPS as an extension.

The messages to and from SAP AII are sent via HTTP POST and must be answered synchronously by HTTP OK. This indicates successful receipt of the message but does not imply successful processing of the message by the application logic.

Alternately, SAP AII 4.0 can be configured to send the *WriteTagData* Command message via the SAP printer spooler.

SAP AII 2.1 and SAP AII 4.0 can be configured to send the *WriteTagData* Command message to a TCP socket on a printer.

Note: TCP socket printing will not be available in releases after SAP AII 4.0.

# 6 APPENDIX

## 6.1 Command Message XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="Command">
		<xs:annotation>
			<xs:documentation>A command message to the device controller</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:choice>
				<xs:element name="WriteTagData">
					<xs:annotation>
						<xs:documentation>This element writes user fields and, if the tags allow, also writes
the identifier by using special data tag field names (see fieldlist element).</xs:documentation>
					</xs:annotation>
					<xs:complexType>
						<xs:sequence>
							<xs:element name="Item" maxOccurs="unbounded">
								<xs:annotation>
									<xs:documentation>Use to structure the child
elements.</xs:documentation>
								</xs:annotation>
								<xs:complexType>
									<xs:sequence>
										<xs:element name="TagID" minOccurs="0"
maxOccurs="unbounded">
											<xs:annotation>
												<xs:documentation>If one or more  tag IDs are given,
then only these tags are written. Otherwise all tags in the radio field are written.</xs:documentation>
											</xs:annotation>
										</xs:element>
										<xs:element name="FieldList">
											<xs:annotation>
												<xs:documentation>List of names of the fields to be
written and their values.
In the case of EPC Class 1 (and above) compliant tags, the EPC (as a tag identifier) can be written using the special
field name "EPC" here.</xs:documentation>
											</xs:annotation>
											<xs:complexType>
												<xs:sequence>
													<xs:element name="Field"
maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>The field - name as
attribute and the value as text- or  CDATA node</xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
<xs:attribute name="name"
type="xs:string" use="required">
<xs:annotation>
<xs:documentation>Name of
the data field on the tag</xs:documentation>
</xs:annotation>
</xs:attribute>
</xs:complexType>
```

```
                                                    </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="format" type="xs:string"
use="optional"/>
                                                <xs:attribute name="jobName" type="xs:string"
use="optional"/>
                                                <xs:attribute name="quantity" type="xs:string"
use="optional"/>
                                                <xs:attribute name="printerName" type="xs:string"
use="optional"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="readerID" type="xs:string" use="optional">
                            <xs:annotation>
                                <xs:documentation>Used to indicate that a specific reader should be
queried.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                    </xs:complexType>
                </xs:element>
                <xs:element name="ConfigureTagFilter">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="FilteringRules" minOccurs="0">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Rule" maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Rule element gives the range of
tags that the Antennas and Ports should look for.</xs:documentation>
                                            </xs:annotation>
                                            <xs:complexType>
                                                <xs:choice>
                                                    <xs:element name="Range">
                                                        <xs:complexType>
                                                            <xs:attribute name="StartTagID"
type="xs:hexBinary" use="required" />
                                                            <xs:attribute name="EndTagID"
type="xs:hexBinary" use="required" />
                                                        </xs:complexType>
                                                    </xs:element>
                                                    <xs:element name="Tags">
                                                        <xs:complexType>
                                                            <xs:sequence>
                                                                <xs:element name="Tag"
maxOccurs="unbounded">
                                                                    <xs:complexType>
                                                                        <xs:attribute
name="TagID" type="xs:hexBinary" use="required" />
                                                                    </xs:complexType>
                                                                </xs:element>
                                                            </xs:sequence>
                                                        </xs:complexType>
                                                    </xs:element>
                                                    <xs:element name="TagMask">
                                                        <xs:complexType>
                                                            <xs:attribute name="mask"
type="xs:string" use="required">
                                                                <xs:annotation>
                                                                    <xs:documentation>Mask
gives the wildcard/dontcare bits that need to be applied to the "value" attribute, to get the tags that match this filter
rule.</xs:documentation>
```

```
                                                                    </xs:annotation>
                                                                </xs:attribute>
                                                                <xs:attribute name="value"
type="xs:string" use="required">
                                                                    <xs:annotation>
                                                                        <xs:documentation>Value
gives the tag's bit pattern to which the mask is applied</xs:documentation>
                                                                    </xs:annotation>
                                                                </xs:attribute>
                                                            </xs:complexType>
                                                        </xs:element>
                                                    </xs:choice>
                                                    <xs:attribute name="action" type="xs:string"
use="required">
                                                        <xs:annotation>
                                                            <xs:documentation>The value of this attribute
is "add" to add this rule, and "delete" to delete the rule. The rule is not deleted if it is being used by any of the
FilteringCommands configured on the system.</xs:documentation>
                                                        </xs:annotation>
                                                    </xs:attribute>
                                                    <xs:attribute name="name" type="xs:string"
use="required">
                                                        <xs:annotation>
                                                            <xs:documentation>The value of this attribute
gives the name of the rule. The FiltetringCommand element uses this name to include these Rules into one
FilteringCommand.</xs:documentation>
                                                        </xs:annotation>
                                                    </xs:attribute>
                                                    <xs:attribute name="ruleType" type="xs:string"
use="required">
                                                        <xs:annotation>
                                                            <xs:documentation> The ruleType specifies
the action to be taken when the Tag Pattern matches the one that is read by the antenna. There are two possibilities:
1) The tag is accepted by the reader and is reported to SAP AII if an event is associated with it. 2) The tag is rejected
by the system without further processing.</xs:documentation>
                                                        </xs:annotation>
                                                    </xs:attribute>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:attribute name="action" type="xs:string">
                                            <xs:annotation>
                                                <xs:documentation>In case the user wants to delete all the
Filtering Rules without specifying each one individually, the value of this attribute equals
"delete".</xs:documentation>
                                            </xs:annotation>
                                        </xs:attribute>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="FilteringCommand" minOccurs="0"
maxOccurs="unbounded">
                                    <xs:annotation>
                                        <xs:documentation>Each FilteringCommand is mapped to a physical
port or a logical collection of ports. In other words, there is one-to-one mapping between a "FilteringCommand" and
the physical or logical port.</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="Rule" maxOccurs="unbounded">
                                                <xs:complexType>
                                                    <xs:attribute name="refid" use="required" />
                                                    <xs:attribute name="action" type="xs:string"
use="required">
                                                        <xs:annotation>
  <xs:documentation> If the value of this attribute is "add", then theFilteringRule must be added to the
FilteringCommand. If the value of this attribute is "delete", then the FilteringRule is deleted from the
```

```
FilteringCommand. The rule is not deleted from the system.</xs:documentation>
                                                                </xs:annotation>
                                                            </xs:attribute>
                                                        </xs:complexType>
                                                    </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="name" type="xs:string" use="required">
                                                    <xs:annotation>
                                                        <xs:documentation> The name attribute provides the name
of the FilteringCommand that is used when a FilteringRule is added or deleted.</xs:documentation>
                                                    </xs:annotation>
                                                </xs:attribute>
                                                <xs:attribute name="action" type="xs:string" use="required">
                                                    <xs:annotation>
                                                        <xs:documentation>This attribute is used to add, edit, or
delete this FilteringCommand from the reader.</xs:documentation>
                                                    </xs:annotation>
                                                </xs:attribute>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="notificationURL" type="xs:string" use="required">
                                        <xs:annotation>
                                            <xs:documentation>Identifies the URL to be informed of the events
generated by the filters being configured</xs:documentation>
                                        </xs:annotation>
                                    </xs:attribute>
                                     <xs:attribute name="readerID" type="xs:string">
                                        <xs:annotation>
                                            <xs:documentation>Used to specify a particular reader. If the reader is not
specified, the device controller determines the reader to which this request is forwarded..</xs:documentation>
                                        </xs:annotation>
                                    </xs:attribute>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="ExtensionCommand">
                                <xs:annotation>
                                    <xs:documentation>This command allows the use of vendor specific features of device
controllers and devices. The interpretation of the value is up to the device controller.</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:any namespace="##any">
                                            <xs:annotation>
                                                <xs:documentation>The value can be formatted as a text string or as
XML.</xs:documentation>
                                            </xs:annotation>
                                        </xs:any>
                                    </xs:sequence>
                                    <xs:attribute name="readerID" type="xs:string" use="optional"/>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="StatusPing">
                                <xs:annotation>
                                    <xs:documentation>Meant for pinging the device controller</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="Response">
                                <xs:annotation>
                                    <xs:documentation>Content of asynchronous response to device controller. The
response is sent during processing of incoming notification message from device controller</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Observation" maxOccurs="unbounded">
                                            <xs:complexType>
```

```
                                        <xs:sequence>
                                            <xs:element name="ID" minOccurs="0">
                                                <xs:annotation>
                                                    <xs:documentation>ID of the observation in the
incoming notification message</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                            <xs:element name="LogicalDeviceID" minOccurs="0">
                                                <xs:annotation>
                                                    <xs:documentation>Logical name of the device that was
contained in the observation element of the incoming notification message</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                            <xs:element name="Cmd" minOccurs="0">
                                                <xs:annotation>
                                                    <xs:documentation>A command to the device
controller, may be to stop the conveyor</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                            <xs:element name="DateTime" type="xs:dateTime">
                                                <xs:annotation>
                                                    <xs:documentation>Date and time according to w3c
date and time format</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                            <xs:element name="Tag" minOccurs="0"
maxOccurs="unbounded">
                                                <xs:annotation>
                                                    <xs:documentation>Tags of interest to the device
controller</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element name="ID">
                                                            <xs:complexType>
                                                                <xs:attribute name="schemeID"
type="xs:string”>
                                                                    <xs:annotation>
                                                                        <xs:documentation>The type
of ID and version, for example, EPC_1.30</xs:documentation>
                                                                    </xs:annotation>
                                                                </xs:attribute>
                                                            </xs:complexType>
                                                        </xs:element>
                                                        <xs:element name="Messages"
type="MessagesType" minOccurs="0" maxOccurs="unbounded"/>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="Messages" type="MessagesType"
minOccurs="0" maxOccurs="unbounded">
                                                <xs:annotation>
                                                    <xs:documentation>Messages for each observation in
the incoming notification message</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="KillTag">
                        <xs:annotation>
                            <xs:documentation>Kill or Decommission Tags</xs:documentation>
                        </xs:annotation>
```

```
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="LogicalDeviceID">
                                    <xs:annotation>
                                        <xs:documentation>Logical device to kill Tags</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="Item" maxOccurs="unbounded">
                                    <xs:annotation>
                                        <xs:documentation>To structure the elements
below</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="TagID" maxOccurs="unbounded">
                                                <xs:annotation>
                                                    <xs:documentation>List of tags to be
killed</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:attribute name="schemeID" type="xs:string“>
                                                        <xs:annotation>
                                                            <xs:documentation>The type of ID and
version, for example, EPC_1.30</xs:documentation>
                                                        </xs:annotation>
                                                    </xs:attribute>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="FieldList">
                                                <xs:annotation>
                                                    <xs:documentation>List of
passwords</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element name="Field">
                                                            <xs:annotation>
                                                                <xs:documentation>Password type in
atrribute "name" and password as value</xs:documentation>
                                                            </xs:annotation>
                                                            <xs:complexType>
                                                                <xs:attribute name="name">
                                                                    <xs:annotation>
                                                                        <xs:documentation>Password
type, for example, ACCESS_PASSWORD, KILL_PASSWORD</xs:documentation>
                                                                    </xs:annotation>
                                                                </xs:attribute>
                                                            </xs:complexType>
                                                        </xs:element>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
                <xs:attribute name="id" type="xs:string" use="optional"/>
            </xs:complexType>
        </xs:element>
        <xs:complexType name="MessagesType">
            <xs:sequence>
                <xs:element name="RespMsg"/>
            </xs:sequence>
```

```
            <xs:attribute name="RespCode" type="xs:string"/>
        </xs:complexType>
</xs:schema>
```

## 6.2 Summary of Message Elements

In the command message sent by SAP AII, the fields marked *"Mandatory"* are always populated. Fields marked *"Optional"* are populated as described. For more information on configuration, see the SAP AII documentation.

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command WriteTagData** | WriteTagData | readerID attribute is provided | |
| | WriteTagData/Item/TagID | Provided for tag update with schemeID attribute | Not used for tag commissioning |
| | Item/FieldList | Element is provided | Attributes for printing are configurable |
| | Item/FieldList/Field | Field named EPC is provided | Additional named fields are configurable |

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command ConfigureTagFilter** | ConfigureTagFilter | notificationURL and readerID are provided | |
| | ConfigureTagFilter/FilteringRules | Action "add" for subscribe, "delete" for unsubscribe | |
| | ConfigureTagFilter/FilteringRules/Rule | "inclusive" | |
| | Rule/Range | | Custom |
| | Rule/Tags | | Custom |
| | Rule/TagMask | HEX mask and value attributes are derived from configuration | |
| | ConfigureTagFilter/FilteringCommand | For Subscribe (action "add"), command is provided in the name attribute<br>For Unsubscribe action is "delete" | "edit" attribute |

| | | | |
|---|---|---|---|
| | ConfigureTagFilter/FilteringCommand/Rule | For Subscribe, filtering, rules are “add”ed. | |

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command ExtensionCommand** | ExtensionCommand | The light color and readerID attribute are provided | |

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command StatusPing** | StatusPing | X | |

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command Response** | Response/Observation/ID | | Provided, if the incoming notification message has an Observation ID |
| | Response/Observation/LogicalDeviceID | X | |
| | Response/Observation/Cmd | | Custom |
| | Response/Observation/DateTime | X | |
| | Response/Observation/Tag | | Custom |
| | Response/Observation/Messages | If messages are present, the attribute RespCode will be provided | Provided, if the activity returns any message during rule processing |

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Command KillTag** | KillTag/LogicalDeviceID | X | |
| | KillTag/Item | X | |
| | KillTag/Item/TagID | X, The attribute schemeID is always populated | Custom |

| | | | |
|---|---|---|---|
| | KillTag/Item/FieldList | | Provided, if password(s) is present in physical object context or Custom |
| | KillTag/Item/FieldList/Field | X, the attribute name is also mandatory | Custom |

The notification message received by SAP AII is expected to include the elements marked *"Mandatory"*. Optional fields apply only in specific scenarios.

| | Element | Mandatory | Optional |
|---|---|---|---|
| **Sensor (Notification)** | Sensor/pmluid:ID | | device controller ID |
| | Observation/DateTime | x | |
| | Observation/Command | | x |
| | Observation/Tag | | only for RFID observations |
| | Tag/pmluid:ID | HEX ID required if Tag is observed | schemeID attribute can give ID version, i.e. EPC_1.27 |
| | Tag/Data/XML/Memory/DataField | | Report tag data name and value |
| | Tag/Data/XML/EPCStatus | | WS or WU for write validation |
| | Observation/Data/XML/ReaderID | Device ID, either ReaderID or LogicalDeviceID must be provided | |
| | Observation/Data/XML/LogicalDeviceID | Device Group ID, either ReaderID or LogicalDeviceID | |

| | | | |
|---|---|---|---|
| | | must be provided | |
| | Observation/Data/XML/GTIN | | Print/write signal |
| | Observation/Data/XML/SSCC | | Print/write signal |
| | Observation/Data/XML/GRAI | | Print/write signal |
| | Observation/Data/XML/ContainerID | | Parent of observed Tags |
| | Observation/Data/XML/DOCUMENT_NO | | Document associated with observation |
| | <custom element> | | <custom activity logic> |

## 6.3 Interface Additions in SAP AII-DC 5.1

This is a summary of changes in the device controller interface since *SAP AII-DC 4.0*3was published in June 2006.

**Additions to SAP AII classic messages**

- *Command*.Message
  - Addition of *StatusPing* Element
  - Addition of *Response* Element
  - Addition of *KillTag* Element
- Notification message:
  - Additional *Observation Data* elements: The *LogicalDeviceID* element was added. Device Controller can now send the SAP AII Device Group ID instead of SAP AII Device ID if they choose to. So, the *ReaderID* element is optional and a customer no longer has to maintain a device below device group in AII. However, either *ReaderID* or *LogicalDeviceID* must be specified. If both are specified then the device must belong to the device group in AII.