VoiceCollect®

Advanced Configuration

- Field Mapping
- Advanced Static routing
- Field Formatting

VDS-II

Version 2006

VoiceCollect GmbH | Justus-von-Liebig-Str. 5 | 61352 Bad Homburg | Germany
voicecollect@voicecollect.de

Table of Contents

1 Document History

Version	Date	**Nom**	**Description of changes**
1.0	2015-01-06	EV	Creation
1.1	2015-03-23	EV	Initial release
1.2	2015-03-27	EV	Peer review with few changes
1.3	2015-10-13	EV	Added "field formatting" feature
1.4	2015-11-13	EV	Added some notices
1.5	2017-04-05	AF, EV	Update of field mapping. Fixed XML examples,
			removed 'dtmf'.
1.6	2017-06-08	AF	Added database maximum field length
			Added extended mapping (field concatenation)
			available for any VDS using vep-adapter 2.2.2 and
			above
1.7	2017-08-30	AF	extended mapping (use separator as default)
			available for any VDS using vep-adapter 2.2.6 and
			above
1905	15.05.2019	PE	VC Modification &Adaptation
2005	2020-05-05	VR	Supported Property list updated Field Mapping and Field Formatting section

2 Acronyms and Abbreviations

CRD	Call Related Data
RTP	Real-time Transport Protocol (RFC3550)
RTSP	Real Time Streaming Protocol (RFC2326)
VDS	VoIP Decoding System (VoiceCollect)
REGEXP	Regular Expression
VEP	VoIP Export Protocol (VoiceCollect)

3 Introduction

This document describes three advanced features of VDS:

- **CRD field Mapping:**

  Map incoming data fields to some predefined VEP fields (recorder database fields) by configuration.

  New incoming fields could be mapped without having to change the code and any specific customer requirement, like which field is displayed where, could be easily fulfilled.

- **Static Routing:**

  Static assignment of calls to specific recorder channel could be done on VEP fields ordered by priority.

  This make possible to assign a list of channel to a single static criteria, or have several criteria checked for assignment.

  Use of regular expressions for matching complex input data

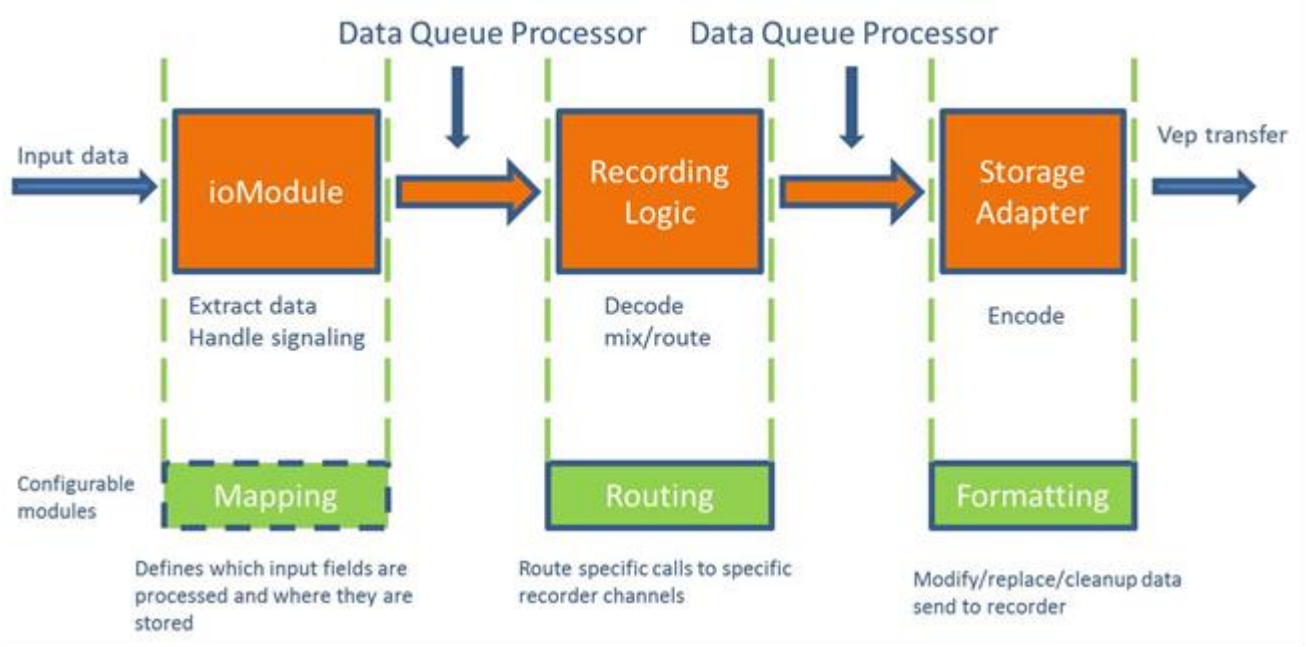
- **Field Formatting:**

  CRD Fields sent to the database through the VEP (voice export protocol) can be modified/replaced to suit customer needs.

  It uses regular expression to find which part of the data needs to be formatted. For example:

  Clean up sip uri: sip:2345@127.32.56.9 formatted to "2345"

4 Overview of VDS design



5 Field Mapping

VDS-II now offers the possibility to map specific incoming data fields into some VEP fields (fields for the

Several of them are mapped internally by the VDS, but we wanted the possibility to be able to change the mapping for some of them, hence to change their location in displayed record data on the Recorder.

It makes it possible to add new incoming fields to some VEP field or change the actual destination field to another one.

If the customer after some months receive a new field like **clientId,** adding it to that configuration, we could for example map it directly to one of the "AUX" fields of the Recorder without having to modify the code.

Customer wants for some reason that one field stored into "AUX1" field of the Recorder to be stored now into "short comment" field instead for having all data in the display page, this can be achieved easily with that configuration without having to modify the code.

The field mapping could be used for mapping customer fields to some VEP fields and then used for static routing to suit customer specific requirement for channel assignment without any change in the code.

The file mapping authorizes the concatenation of incoming fields for the construction of sophisticated mapping rules.

A new bean needs to be added into the spring-config.xml file:

```
<bean id="mappingLogic" class="com.atissystems.recorder.vds.mapping.MappingConfiguration"init-method="init"/>
```

And it has to be referenced in the ioModule:

```
<property name="maplogic" ref="mappingLogic"/>
```

Any change in the spring-config.xml file, needs a restart of VDS

The file crd-configuration.xml contains the mapping:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XmlMappingConfigurationList>
<XmlMappingincomingFieldName="ConnectedNr,CalledNr"vepFieldName="bparty" />
<XmlMappingincomingFieldName="FrequenceId,CallingNr"vepFieldName="aparty" />
<XmlMappingincomingFieldName="CallRef+':'+CallID"vepFieldName="caseId"/>
<XmlMappingincomingFieldName="Operator+'@'+Position,Operator" vepFieldName="userId" />
</XmlMappingConfigurationList>
```

The incomingFieldName is case sensitive so the incoming user field need to be spelled carefully

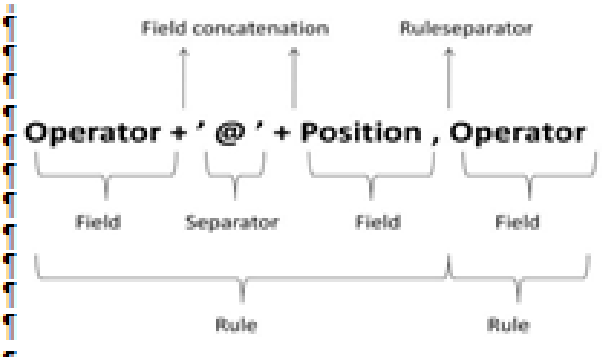
Any change in crd-configuration.xmltakes effect on the next call. There is no need to restart the application.

The ***XmlMappingincomingFieldName*** contains a list of **Rules** which are separated with a colon (no space allowed).

A rule is built with **Separators** and **Fields** which are separated by the **+** sign.

A **Field** is a case sensitive string

A **Separator** is a string delimitated by single quotes (‘)



Following characters are forbidden in Separators: column (,), simple-quote ('), double-quote ("), ampersand (&), plus sign (+).

It is possible to assign several rules for the same VEP field, the order they are enumerated gives a priority order. They need to be separated by a colon (no space).

A rule is valid if all fields which compose the rule are present.

For vep field **bParty** from the xml example above

- if **CalledNr** and **ConnectedNr** are present, then only **ConnectedNr** will be copied to **bParty** field.

- if only **CalledNr** is present, then **bParty** contains **CalledNr.**

- if only **ConnectedNr** is present, then **bParty** contains **ConnectedNr.**

```
<XmlMappingincomingFieldName="Operator+'@'+Position,Operator" vepFieldName="userId"
```

With the example above, let's assume Operator=OP1 and Position= CWP1

- if **Operator** and **Position** are present then userId = OP1@CWP1 (first rule apply)

- if only **Operator** is present first rule fails (Position is missing), second rule is applied then userId = OP1

- if only **Position** is present both rules fail userID is empty. To prevent it, we could add Position as a third rule.

```
<XmlMappingincomingFieldName="Operator+'@'+Position,Operator,Position"
```

- in this case, if only **Position** is present, the two first rules fail but third one is valid: userId=CWP1

To prevent empty field a default value could be inserted

```
<XmlMappingincomingFieldName="Operator+'@'+Position,Operator,Position,'Unknown'"
```

- if **Position** and **Operator** are missing all 3 first rules fail and the "separator" **Unknown** will be used (simple quote are needed) userId = Unknown
-

Only the following VEP fields are sent to the Recorder database:

aParty, bParty, userId, actionId, caseId, shortComment, vdsComment, sessionId, aux1, aux2, aux3, aux4, aux5, aux6, aux7, aux8, aux9, aux10.


aParty, bParty and userId fields are limited to 32 characters.


In addition, the internal field routingIpAddr can be mapped. Even though it is not sent to the Recorder database, it can be used for static routing (Refer Section 6).

The supported configurable VEP fields are listed below.

- aParty
- bParty
- userId

- aux1
- aux2
- aux3
- aux4
- aux5
- aux6
- aux7
- aux8
- aux9
- aux10
- caseId
- actionId
- dtmf
- routingIpAddr
- shortComment
- vdsComment
- sessionId

6 Advanced static routing

The "routing logic" module is responsible for finding the Recorder channel on which to record a call, depending on the received call related data stored in VEP fields and according to the static routing configuration.

The new static routing use regexp (regular expressions) for the filtering of received call related data, which allows powerful matching and the delivery of a list of static channels instead of only one.

The list of criteria for call related data filtering is created using a specific matching order, which gives a priority among received call data to find a match and then route.

The names and matching order of VEP fields to be used for routing are defined in the file `spring-confg.xml`, while the routing table is defined in the file `routing-configuration.xml`.


Assignment of call related data to VEP fields can be changed with the field mapping feature (see chapter 1).

6.1 Static routing table

Here follows an example of the static routing table in the file **`routing-configuration.xml`** :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<routingTable>
<location>Fontaines</location>
<recorderList>
<recorder name="R0001">
<channelList>
<channel id="1" filter="555123"/>
<channel id="2" filter="555456"/>
<channel id="3" filter="bob@123.com"/>
<channel id="4" filter="mary"/>
<channel id="5" filter="^[0-9]*:555789$"/>
```

```
<channel id="6" filter="*"/>
<channel id="7" filter="*"/>
<channel id="8" filter="*"/>
<channel id="9" filter="/"/>
<channel id="10" filter="/"/>
</channelList>
</recorder>
</recorderList>
</routingTable>
```


Any change in ***routing-configuration.xml*** takes effect on the next call. There is no need to restart the application

`filter="*"` means channels assigned to dynamic channel pool

`filter="/"` means channels assigned to blocked channels

Any other value for `filter` will put corresponding `channel` into the static channel pool. When the routing logic will receive any data matching the value in `filter` (for example a phone number); it will tell the Recorder to use that specific channel for recording the call.

As explained in next section, more complex matching patterns can be used for the definition of a `filter`.


Static routing takes the values of the call related data in their complete form. The "Field Formatting" feature described in section 5 has no effect on static routing

6.2 Regexp

The "regular expression" (regexp in short) is a syntax which provides powerful pattern matching.

The syntax is quite hard and mistakes easily made.

We will just describe here the syntax which will be used 99% of the time.

Example of received values:
`123 - xx123yy - 12345 - 54123 - bob@123.com`

Partial Matching

`filter="123"` means any field which contains `123`
All the above fields will match

Starts With

```
filter="^123" means all the fields which starts with 123
and12345 will match
```

Ends With

```
filter = "123$" means all the fields which ends with 123
123and 54123 will match
```

Exact Matching

```
If you combine the last two
filter = "^123$" means all fields which match exactly 123
123 only will match
```


The filter"192.168.163.1" will match 192.168.163.1 but it will match 192.168.163.1x or 192.168.163.1xx. Use the exactMatching property (see below) or the meta-characters ^ and $.

Special characters

As seen above, some characters such as "^" and "$"have special meaning in regexps (they are called meta-characters). They should be "escaped" if you want to match them "as is" in your filter. Escaping a meta-character is done by preceding it with "\".

The dot

The "." matches any single character.

Be careful if you want to match an ip address and you use `"192.168.123.15"`as the filter. This will match `"192A168-123,15"` as well as `"19201680123015"` and`"192.168.123.15"`because`"."` is interpreted as a special character replacing any single character.

If you want to match the character ".", you must use "\" to escape it, for example:

`"192\.168\.123\.15"`where all dots are escaped and are treated as characters and not meta-characters.

The star

The `"*"` matches any preceding character 0 or more time

`"192*"` matches`19`, `192`, `192`, `1922`,etc…

`"19*2"` matches`12`, `192`, `1992`, `19992`, etc…

Like for the ".", the `"*"` must be escaped if it must be used as a character in a filter.

Regular expressions are extremely powerful and complicated. Using more complex expressions than the ones above should be done with care, otherwise routing assignment will fail or it will be totally wrong.

Learn more on "regexps" on Wikipedia: http://en.wikipedia.org/wiki/Regular_expression

You can test your regexps with online tools: http://regex101.com

In order to simplify the static routing table, VDS offers a configuration option to always perform exact matching on the static routing filters. By default, the routing logic performs partial matching (see examples above). When the option is set, it will force the routing logic to look up for exact matches of the filters without the later having to be prepended with "^" and have "$" appended.

The bean **logicModule** in spring-config.xml must have the property **exactMatching** set to true:

```
<bean
name="logicModule"class="com.atissystems.RTSPRecorderServer.RoutingLogicEurocaeRtspActive"
init-method="init">

<property name="router" ref="routingLogic" />

<property name="exactMatching" value="true" />

</bean>
```

Note that other regexp rules still apply.


Any change in spring-config.xmlneeds a VDS-II restart.

6.3 Matching Order

The routing logic module can be initialized with a matching order.

This matching order gives a priority on internal VEP fields to be looked up in the static routing table.

The default fields are **userId**, **aParty**, **bParty,** in this order.

When trying to find a suitable channel for static routing, the routing module will first try to find a channel where the filter matches the value of **userId,** then a channel which matches the value of **aParty**and then a channel which matches the value of **bParty**.

The available criterion are :
aParty, bParty, dtmf, userId, actionId, caseId, shortComment, aux1, aux2, aux3, aux4,

aux5, aux6, aux7, aux8, aux9, aux10, routingIpAddr.

The bean **logicModule** in spring-config.xml must have the property **matchingOrder** set to the comma-separated list of fields which values must be looked up for matches in the static table:

```
<bean
name="logicModule"class="com.atissystems.RTSPRecorderServer.RoutingLogicEurocaeRtspActive"
init-method="init">

<property name="router" ref="routingLogic" /> <property name="matchingOrder"
value="aParty,bParty" /> </bean>
```

Note that every criterion can be used only once.


Any change in spring-config.xml needs a VDS-II restart.

6.4 Preferred channels

In older versions, the routing logic module was stopping its channel lookup immediately after having found a match for a static channel. Now the routing module will go through all entries in the static routing table and it will try to match them all.

Each channel found is added to a list of preferred channels (with same priority as the matching order).

That list will be used by the VDS to route the call to the recorder.

The preferred channel list contains only channels which have matched one of the filters and it is ordered by matching order.

7 Field formatting

A new feature has been added which permits to modify the format of the data which is sent to the Recorder.

It can be applied on the same fields used for matching order in the static routing (cf 4.2).

aParty="+41 79 852 3656" can be for example truncated to last 4 digits to have a better reading .

shortComment="CALL_FORWARDING" can be translated in another language by the user.

The formatting feature has to be set into the spring-config.xml file in the storage Adapter bean

```
<bean id="storageAdapter" class="com.atissystems.recorder.vds.core.RecordingAdapter"init-
method="init">
<constructor-arg index="0"><ref bean="mixer"/></constructor-arg> <property
name="configurationPort" value="8510" />
<property name="dataPort" value="8540" /> <property name="router" ref="routingLogic" />
<property name="listeningIp" value="172.16.240.91" /> <property name="formatter"
ref="vepFormatter" /> </bean>
<bean id="vepFormatter" class="com.atissystems.recorder.vds.formatting.VepFormatter"init-
method="init" />
```


Any change in spring-config.xml needs a VDS-II restart.

A new empty file, formatting-configuration.xml, will be created.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FormattingRules>
</FormattingRules>
```

You can add Formatters in this file.


Any change in formatting-configuration.xml takes effect on the next call. There is no need to restart the application.

Formatters are defined with 3 fields:

vepFieldName: list of VEP parameters to be modified separated by comma (no space allowed)

matchingRule: regular expression defining what to be replaced

replacedBy: replacement expression

Example:

```
<Formatter vepFieldName="aParty,bParty,userId"matchingRule="(\+4181288)"replacedBy="" />
```

The formatter will be applied to the list of parameter defined in vepFieldName.


The matchingRule parameter is a Regular Expression.


The field replacedBy contains the replacement string but it can contain matching groups too.

Several formatters can be set for the same field they will be applied in the order they are defined and they are cumulative.

For example:
We are receiving a phone number with the following format +41 79 852 3656

We would like to remove the country code +41 and replace it with 0 and keep the last 4 digits when calls are coming from 79 852.We prepare two rules:

```
<Formatter vepFieldName="aParty"matchingRule="(\+41)"replacedBy="0" />
<Formatter vepFieldName="aParty"matchingRule="(0 79 852)[0-9]{4}"replacedBy="" />
```

The parenthesis in the regular expression denotes a matching group.As it is the only group, only this part will be evaluated and affected by the replacement operation.

The "+" sign being a regexp meta-character.Wemustescape it by prepending "\".

Rule 1 will make aParty="+41 79 852 3656"to be transformed to "0 79 852 3656".

Rule 2 will then be applied to aParty. The regular expression "(0 79 852)[0-9]{4}"matches 0 79 852 followed by 4 times any digit from 0 to 9 and assign 0 79 852 to group 1. Any matching group is replaced by empty string.

aParty="0 79 852 3656" will be transformed to "3656".

Following `formatting-configuration.xml` example file shows same formatting rules applied to the fields for both the calling and the called parties:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FormattingRules>
<Formatter vepFieldName="aParty,bParty"matchingRule="(\+41)"replacedBy="0" />
<Formatter vepFieldName="aParty,bParty"matchingRule="(0 79 852)[0-9]{4}"replacedBy="" />
</FormattingRules>
```

It works with string too

If the field shortComment contains some English word like call forward it could be replaced by another language.

Example :

```
<Formatter vepFieldName="shortComment"matchingRule="(call
forward)"replacedBy="Anrufweiterleitung" /> The field which match is replaced immediately.
```

```
<Formatter vepFieldName=shortComment"matchingRule="(call
forward)"replacedBy="Anrufweiterleitung" /> <Formatter
vepFieldName="shortComment"matchingRule="(Anrufweiterleitung)"replacedBy="call forward" />
```

Adding 2 formatter like the above will not change anything besides consuming cpu. "call forward" in shortComment is changed to "Anrufweiterleitung" "Anrufweiterleitung" inshortComment is changed to "call forward"

Similarly, if we need to format the **Call Id parameter** the below formatter should be added in the configuration XML file

```
<Formatter vepFieldName="callId" matchingRule="(.*)@.*" replacedBy="$1" />
```

This will strip the characters after the @ Symbol.

For Example, if the Call-ID*: 291882fc4133aa6b1095a808103c416c@192.168.100.131:5060* comes like this from the input system then it will be trimmed as **291882fc4133aa6b1095a808103c416c**

The supported configurable VEP fields are listed below.

- aParty
- bParty
- userId
- aux1
- aux2
- aux3
- aux4
- aux
- aux6
- aux7
- aux8
- aux9
- aux10
- caseId
- actionId
- dtmf
- routingIpAddr
- shortComment
- callId

8 Combining features

When we combine regexp filtering with matching order and the preferred channels list, we can reach a real powerful static assignment.

For example we could use same filter on several channels which will, if the filter is correctly set, assign a group of channels to one specific field value.

This combination can be even more powerful when we can map customer input data field to specific VDS fields without having to modify the code.

The features described in this document are processed in the following order
Mapping → Routing → Formatting
Ensure mapping is correct before to prepare static routing.

***** END OF DOCUMENT *****