

NAMA VDI Standards

Web Service Design

VDI S2S DEX 1.0

DEX MESSAGE TRANSFER



National Automatic Merchandising Association

20 N. Wacker Drive, Chicago IL USA

www.vending.org

© NAMA VDI Task Force

May 1, 2010

Table of Contents

Chapter 1 GetDex/UploadDex Webservices	3
1.1 Background.....	3
1.2 Authentication	3
1.3 Assumptions.....	4
Chapter 2 GetDex message.....	4
2.1 Purpose	4
2.2 Parameters list.....	4
2.3 Return.....	5
Chapter 3 UploadDex message	6
3.1 Purpose.	6
3.2 Parameters list.....	7
3.3 Return.....	8
3.4 UploadDEX XML.....	8
3.5 Return.....	10
Chapter 4 XML Tags and Attributes Descriptions	10
4.1 Tags Descriptions.....	10
4.2 Attributes Description.....	11
Chapter 5 XML Example.....	13
Chapter 6 Future Goals.....	14
VDI Task Force Roster.....	15

Note: VDI S2S DEX 1.0 was drafted by Glenn Butler, Lenny Filatov and Anton Rakushkin of Crane Merchandising Systems with contributions from the entire NAMA VDI Task Force. Document published by NAMA, 20 N.Wacker Drive, Chicago IL 60606 USA. Original Release 1.0 → May 2010.

Posted by VDI Task Force Coordinator: Michael Kasavana, NAMA Endowed Professor, MSU

Chapter 1 GetDex/UploadDex Webservice

1.1 Background

Open system webservice are typically required to transmit DEX payloads from one system, be it a device or application or server, to another. A common use of these webservice is the push transmission of a DEX payload from a **deployed telemetry unit, or server, or application** to a backend enterprise route management system (UploadDex Webservice). The payload document can also be used as a standalone file and be traded using FTP or any other transmission method. Another common use would be an enterprise route management system contacting another system to “pull” DEX files from another server or application (GetDex Webservice).

In both these Webservices, an attempt has been made to keep the DEX payload XML as similar as possible, other features are:

- Arguments that include the DEX provider identification (NAMA identifier)
- Ability to include the customer identifier as a string
- Identification of the device or application transmitting the data
- Versioning
- Optional Compression, with ability to transmit compression meta data (UploadDex only)
- Metadata information including time stamp, result code, and GMT offset
- Transmission of a single DEX record, multiple records in one transmission, or a large compressed payload of multiple records pulled over time, each with their own meta data
- Error Code upload allows transmission of metadata when DEX reads fail
- DexType element to define the type of service initiating the DEX upload (refill, cash-out, archive, current, etc.)

1.2 Authentication

The webservice provider will authenticate the webservice consumer against a username/password entered into SOAP header information (header element: authorization, basic base64 encoded username:password). The username/password is part of basic HTTP basic authentication. The SOAP header information is the same as standard HTTP username/password credentials passed as part of the transfer header

Example (within the HTTP header):

Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

For example, the base64 above decodes to “username:password”.

If the trading partners agree, authentication can optionally be done in a more secure manner using SSL and HTTPS.

Refer to <http://www.w3.org/Protocols/HTTP/1.0/spec.html#BasicAA> for additional details.

1.3 Assumptions

This document is intended to provide a framework and standard for a message. The details surrounding delivery for that message are negotiable by the trading partners. It is possible to trade these messages by means other than the recommended web services method. For example, AS/2, FTP and email are all viable methods for delivering a DEX VDI 1.0 message from one trading partner to the other.

It is a best practice to meet with your trading partner prior to sending any documents to establish a user ID and to negotiate a compression method, a decision on http or https and other variables that may exist.

Chapter 2 GetDex message

2.1 Purpose

A webservice that allows receiving of DEX payloads from Server or application system (webservice provider) by another Server or application system (webservice consumer).

VDIxml GetDex(parameters)

Note: VDIxml here and later refers to artificial data type, which is XML String according to VDI standard.

2.2 Parameters list

This call takes 12 parameters:

Name	Type	Required	Enumeration	Description
TransactionID	String(16)	Yes		Unique request identifier from the device. Used to prevent 'echo' data if a retry sent from the device to the Server. In some cases if device made a call to a server and didn't receive response back, device might repeat the call (echo). TransactionID will ensure that a server would understand if this is repeat call or new one.
ProviderID	String(32)	Yes		Name of the provider of the source data, for example. inOne or MEI sending DEX.

CustomerID	String(32)	Yes		Name of the bottler or operator in which the data belongs, like BestFamilyVending.
ApplicationID	String(32)	Yes		Identifies the provider application which called UploaDex Web Method .
ApplicationVersion	String(8)	Yes		Version of the application which called UploaDex Web Method
DeviceList	Array of strings. Each element String(16)	One of [DeviceList, OutletList] is required		A list of telemetry device IDs. If present the call will only return DEX collected by telemetry devices in the list.
OutletList	Array of strings. Each element String(16)	One of [DeviceList, OutletList] is required		A list of machine identifiers. If present the call will only return DEX collected at machine in the list.
OnOrAfter	Date/time	No		If provided, only DEX collected at or after the specified time can be returned.
OnOrBefore	Date/time	No		If provided, only DEX collected at or before the specified time can be returned.
ReturnSet	String(16)	Yes	FIRST, LAST, ALL	Determines how many DEX records should be returned for each device or machine. Valid values are: FIRST, LAST, ALL
UserData	String	No		User defined data. Not Required.
VDIXMLVersion	String(8)	Yes		Version of VDIXML to send back.

2.3 Return

GetDex message returns XML string, that contains VDIReturn xml and collection of dex records according to values of GetDex parameters:

```
<?xml version="1.0" encoding="utf-8"?>
<VDITransaction VDIXMLVersion="1.0"
  <VDIReturn
    <Code>0</Code>
```

```

        <Message>Success</Message>                                </VDIReturn
<DEXList RecordsCount="2" DEXEncoding="" DEXCompressionType=""
    DEXCompressionParam="">
    <DexTransmission
        DeviceID=""
        OutletID=""
        TransmitTime=""
        GMTOffset="">
        <DexCollection RecordsCount="2">
            <DEX
                ReadDateTime="" GMTOffset="" FileSize=""
                DexReason="" DexType="" ResponseCode="">
                <RawDEX>
                    DXS*9259630001*VA*V1/1*1**100
                    DXE
                </RawDEX>

                <UserData></UserData>
            </DEX>
            <UserData></UserData>
        </DexCollection>
        <UserData></UserData>
    </DexTransmission >
    <DexTransmission ></DexTransmission>
    <UserData></UserData>
</DEXList>
<OtherCollectionsOrLists></OtherCollectionsOrLists>
<UserData></UserData>
</VDITransaction>

```

Chapter 3 UploadDex message

3.1 Purpose

A webservice that allows the transmission of DEX payloads from device or application system (webservice consumer) to another application system serving as the host for the webservice (webservice provider).

XML string UploadDex(parameters)

3.2 Parameters list

This call takes 11 parameters.

Name	Type	Required	Enumeration	Description
TransactionID	String(16)	Yes		Unique request identifier from the device. Used to prevent 'echo' data if a retry sent from the device to the Server. In some cases if device made a call to a server and didn't receive response back, device might repeat the call (echo). TransactionID will ensure that a server would understand if this is repeat call or new one.
ProviderID	String(32)	Yes		Name of the provider of the source data, for example inOne or MEI sending DEX.
CustomerID	String(32)	Yes		Name of the bottler or operator in which the data belongs, like BestFamilyVending.
ApplicationID	String(32)	Yes		Identifies the provider application which called UploadDex Web Method .
ApplicationVersion	String(8)	Yes		Version of the application which called UploadDex Web Method
DEXEncoding	Integer	Yes	0 – 'None' 1 – 'ISO8859-1' 2 – 'UTF-8'	Type of encoding used to encode DEX files. Can be only ISO8859-1 or UTF-8. This is enumeration. Accepted values are:
DEXCompressionType	String(32)	No	0 – 'None' 1 – RAR 2 – ZIP 3 – 7ZIP 4 - Other	Type of Compression used. Can be omitted or 'NONE' to flag that no compression was used.
DEXCompressionParam	String	No		Additional compression information if anything required for decompression.
UserData	String	No		User defined data. Not Required.

VDIXMLVersion	String(8)	Yes	Version of VDIXML sent as VDIXML parameter (1.0 initially)
VDIXML	XML String	Yes	XML String representing collection of DEX Files with additional attributes. See Format Below.

3.3 Return

UploadDex() returns VDI standard return xml string containing error code and text message:

```
<?xml version="1.0" encoding="utf-8"?>
<VDIReturn
  <Code>0</Code>
  <Message>Success</Message>
</VDIReturn
```

3.4 UploadDEX XML

VDIXMLVersion – This is a version of VDIXML. If VDIXML format will ever change VDIXML parsers would need to be changed also. It's unrealistic to think that this can be or will be done by whole industry in the same time. To prevent this situation VDIXMLVersion would identify what parser(version of VDIXML) to use.

VDIXML - XML String representing collection of DEX Files with additional attributes in the following format:

```
<VDITransaction VDIXMLVersion="1.0"
  TransactionReason="UploadDEX" TransactionID="123"
  TransactionTime="2002-05-30T09:00:00"
  ProviderID="RemoteDataVendor" CustomerID="OneFamilyVend"
  ApplicationID="Any" ApplicationVersion="1.021" >
  <DEXList RecordsCount="2" DEXEncoding=""
  DEXCompressionType="" DEXCompressionParam="">
  <DexTransmission DeviceID="12" TransmitTime="" GMTOffSet="">
  <DexCollection RecordsCount="2">
  <DEX ReadDateTime="" GMTOffSet="" FileSize="" DexReason=""
  DexType="" ResponseCode="">
  <RawDEX>
```

```

    DXS*9259630001*VA*V1/1*1**100
    DXE
  </RawDEX>
  <UserData></UserData>
</DEX>
  <DEX ReadDateTime="" GMTOffSet="" FileSize="" DexReason=""
DexType="" ResponseCode="">
  <RawDEX>
    DXS*9259630211*VA*V1/1*1**100
    DXE
  </RawDEX>
  <UserData></UserData>
</DEX>
  <UserData></UserData>
</DexCollection>
  <UserData></UserData>
</DexTransmission >
  <DexTransmission DeviceID="15" TransmitTime="" GMTOffSet="">
  <DexCollection RecordsCount="2">
  <DEX ReadDateTime="" GMTOffSet="" FileSize="" DexReason=""
DexType="" ResponseCode="">
  <RawDEX>
    DXS*9259630001*VA*V1/1*1**100
    DXE
  </RawDEX>
  <UserData></UserData>
</DEX>
  <DEX ReadDateTime="" GMTOffSet="" FileSize="" DexReason=""
DexType="" ResponseCode="">
  <RawDEX>
    DXS*9259630211*VA*V1/1*1**100
    DXE
  </RawDEX>
  <UserData></UserData>
</DEX>
  <UserData></UserData>
</DexCollection>
  <UserData></UserData>
</DexTransmission >
</DexTransmission ></DexTransmission>

```

```

<UserData></UserData>
</DEXList>
<OtherCollectionsOrLists></OtherCollectionsOrLists>
<UserData></UserData>
</VDITransaction>

```

3.5 Return

UploadDEX message returns XML string as standard VDI return structure:

```

<?xml version="1.0" encoding="utf-8"?>
<VDIReturn
  <Code>0</Code>
  <Message>Success</Message>
</VDIReturn

```

Chapter 4 XML Tags and Attributes Descriptions

This applies to both GetDEX and UploadDEX messages.

4.1 Tags Descriptions

Name	Type	Enumeration	Description
<VDITransaction>			This is the Root Element for VDI transaction. All other elements are nested within it
<DEXList>	Collection		This element is start for DEX related data transmission.
<DexTransmission>	Collection		This is an element within <DEXList>. Dex Transmission combines Device information and collection of DEX reads for this Device transmitted from the same device in the same time. In some cases Device may read DEX several times and not being able to send it individually. All DEX reads not sent yet are being combined into DEX transmission. <DexList> may have multiple <DexTransmissions> collections for the same or different devices. Context of each individual <DexTransmission> will be compressed based on DEXEncoding and DEXCompressionParam parameters of UploadDex method or same name attributes of <DEXList> element.
<DexCollection>	Collection		This is an element within <DexTransmission> and

<Dex>	Collection
<RawDEX>	String
<UserData>	String
<OtherCollectionsOrLists>	Collection

represents combination of dex records and their specific information for the same device, read in the different time, but being transmitted in the same time. Each <DEXCollection> may combine information about multiple DEX files. This is an element within <DexCollection>. Each <DEX> collection combines information about single DEX file and has DEX read itself. This is an element within <DEX> and represents **Encoded** Raw Dex. (for simplicity DEX is not encoded in XML sample above). RAWDEX context is going to be encoded based on DEXEncoding parameter of UploadDEX method or DEXEncoding attribute of DEXList element. This is an element within different levels of VDIXML and represents provider specific data. This tag is optional. Just a place holder to show that other elements (like alerts, alarms, etc...) can go here.

4.2 Attributes Description

Tag Name	Attribute Name	Type	Description
VDITransaction	VDIXMLVersion	String(8)	Version of VDIXML used in XML
VDITransaction	TransactionReason	String(32)	Reason for transaction. It can be GetDEX, UploadDex, or any other reason for this particular transaction
VDITransaction	TransactionID	String(16)	Unique identifier for transaction. Used to prevent transmitting or processing of the same transaction.
VDITransaction	TransactionTime	Date/Time	Time when this transaction was transmitted
VDITransaction	ProviderID	String(32)	Name of the provider of the source data, for example inOne or MEI sending DEX.
VDITransaction	CustomerID	String(32)	Name of the bottler or operator in which the data belongs, like BestFamilyVending.
VDITransaction	ApplicationID	String(32)	Identifies the provider application which created transaction
VDITransaction	ApplicationVersion	String(8)	Version of the application which created transaction
DEXList	RecordsCount	Integer	RecordsCount represents a number of transmissions within the list
DEXList	DEXEncoding	Integer (Enumeration)	Type of encoding used to encode

DEXList	DEXCompressionType	String(32)	DEX files. Can be only ISO8859-1 or UTF-8. This is enumeration. Accepted values are: 0 – 'None' 1 – 'ISO8859-1' 2 – 'UTF-8' Type of Compression used. Can be omitted or 'NONE' to flag that no compression was used. For Now only one type is supported: 0 - 'None'
DEXList	DEXCompressionParam	String	Additional compression information if anything required for decompression.
DexTransmission	DeviceID	String(32)	ID of the device that this <DEXTransmission> is for.
DexTransmission	TransmitTime	Date/Time	Date/time of this particular transmission. Value is local date/time of transmission.
DexTransmission	GMTOffset	Integer	offset between local time of transmission place and GMT (LocalTime – GMTTime)
DEX	ReadDateTime	Date/Time	Date/time when Dex was read. This is device local Date/Time.
DEX	GMTOffset	Integer	Offset between device local time and GMT (LocalTime – GMTTime)
DEX	FileSize	Long	size of DEX file in bytes.
DEX	DexReason	Integer (Enumeration)	during what user activity (if any) DEX were taken. Possible values currently defined as: Refill, Service, Cash Out, Archive. List of values can be extended with new types. 0 – Scheduled DEX Read 1 – Service Button Pressed 2 – Door Open 3 – Door Closed 4 – Fulfills Specific Request 5 – Alert/VMC initiated 6 – Dex from external Device (Driver HH or similar) 99 – Other
DEX	DexType	Integer (Enumeration)	'Full' dex was sent/taken or only specific portion of DEX stream. 0 – Full Dex and 99 – Other
DEX	ResponseCode	String(128)	status of DEX read. If DEX read is successful then value should

be OK. In case of unsuccessful read value will be text of the error.

Chapter 5 XML Example

After necessary encoding and encryption final VDI XML supplied to or by the web methods GetDex and UploadDex will look as (depending on actual values):

```
<?xml version="1.0" encoding="utf-8"?>
<VDITransaction VDIXMLVersion="1.0"
    TransactionReason="UploadDEX"
    TransactionID="123"
    TransactionTime="2002-05-30T09:00:00"
    ProviderID="RemoteDataVendor"
    CustomerID="OneFamilyVend"
    ApplicationID="Any"
    ApplicationVersion="1.021" >
  <DEXList RecordsCount="2" DEXEncoding="UTF-16"
    DEXCompressionType="None"
    DEXCompressionParam="" >
    <DexTransmission DeviceID="234432"
      TransmitTime="2002-06-30T11:00:00"
      GMTOffset="-2">
      Dfkjndfkjnvdljksnflkvjdsnflkvnsikjdfnvlkjsdsnlkjdfgdsghgkjtjytrtreds
    </DexTransmission >
    <DexTransmission DeviceID="8765"
      TransmitTime="2002-06-30T12:00:00"
      GMTOffset="-2">
      Dfkjndfkjnvdljksnflkvjdsnflkvnsikjdfnvlkjsdsnlkjdfgdsghgkjtjytrtreds
    </DexTransmission>
    <UserData></UserData>
  </DEXList>
  <OtherCollectionsOrLists></OtherCollectionsOrLists>
  <UserData></UserData>
</VDITransaction>
```

UploadDEX (VDI common return) return:

```
<?xml version="1.0" encoding="utf-8"?>
<VDIReturn
  <Code>1</Code>
  <Message>Cannot get DEX data for device
'00CD123456676878'</Message>
</VDIReturn
```

Chapter 6 **Future Goals**

Members agree that some loosely specified strings could be tightened up using more specific enumerated types. The committee agrees to work on this in future specifications but does not want to impact current implementations of the interface.