NAMA VDI Compliant Standard
Server-to-Server DEX Message
Transfer Standard
VDI S2S-DEX
Version 1.1

National Automatic Merchandising Association
20 N. Wacker Drive, Chicago IL USA
www.vending.org
© NAMA VDI Task Force
Initial Posting → November 2011
# Table of Contents

Chapter 1  DEX Transfer Webservices ................................................................. 3  
1.1 Background ............................................................................................... 3  
1.2 Authentication .......................................................................................... 3  

Chapter 2  GetDex message ............................................................................ 4  
2.1 Purpose ....................................................................................................... 4  
2.2 Parameters list .......................................................................................... 4  
2.3 Example Message ....................................................................................... 5  
2.4 Return ......................................................................................................... 5  

Chapter 3  UploadDex message ....................................................................... 7  
3.1 Purpose ....................................................................................................... 7  
3.2 Parameters list .......................................................................................... 7  
3.3 UploadDEX XML ....................................................................................... 8  
3.4 Return ......................................................................................................... 9  

Chapter 4  XML Tags and Attributes Descriptions ......................................... 9  
4.1 Tags Descriptions ...................................................................................... 9  
4.2 Attributes Description .............................................................................. 10  

VDI Roster (November 2011) .......................................................................... 13
Chapter 1  DEX Transfer Webservices

1.1 Background

Open system webservices 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 webservices is the push transmission of a DEX payload from a deployed telemetry unit, or server, or application to a backend enterprise route management system using an 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 using a GetDex Webservice.

In these Webservices, an attempt has been made to keep the payload XML as similar as possible. Other features include:

- Arguments that include the 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 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 dXNlc3NhbWU6cGFzc3dvcmQ=

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

Refer to [http://www.w3.org/Protocols/HTTP/1.0/spec.html#BasicAA](http://www.w3.org/Protocols/HTTP/1.0/spec.html#BasicAA) for additional details.
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionID</td>
<td>String(16)</td>
<td>Yes</td>
<td></td>
<td>Unique request identifier from the device. Used to prevent ‘echo’ data if a retry is sent from the device to the Server. Server returns an error and rejects the message if a duplicate TransactionID is detected.</td>
</tr>
<tr>
<td>ProviderID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Name of the provider of the source data, for example. inOne or MEI sending DEX.</td>
</tr>
<tr>
<td>CustomerID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Name of the bottler or operator in which the data belongs, like BestFamilyVending.</td>
</tr>
<tr>
<td>ApplicationID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Identifies the provider application which called UploadDex Web Method.</td>
</tr>
<tr>
<td>ApplicationVersion</td>
<td>String(8)</td>
<td>Yes</td>
<td></td>
<td>Version of the application which called UploadDex Web Method. A list of telemetry device IDs. If present the call will only return DEX collected by telemetry devices in the list.</td>
</tr>
<tr>
<td>DeviceList</td>
<td>Array of strings, each element String(16), enclosed in &lt;Item&gt; &lt;Item&gt; tags</td>
<td>No</td>
<td></td>
<td>A list of machine identifiers. If present the call will only return DEX collected at machine in the list.</td>
</tr>
<tr>
<td>OutletList</td>
<td>Array of strings, each element String(16), enclosed in &lt;Item&gt; &lt;Item&gt; tags</td>
<td>No</td>
<td></td>
<td>If provided, only DEX collected at or after the specified time can be returned.</td>
</tr>
<tr>
<td>OnOrAfter</td>
<td>Date/time</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.3 Example Message

Following is an example SOAP 1.2 message body:

```xml
<ws:GetDex xmlns:ws="urn:ExampleVDIService">
  <TransactionID>LIYaSha0nEqhSBLy</TransactionID>
  <ProviderID>ExampleProvider</ProviderID>
  <CustomerID>ExampleCustomerID</CustomerID>
  <ApplicationID>ExampleApp</ApplicationID>
  <ApplicationVersion>1.2.3.4</ApplicationVersion>
  <VDIXMLVersion>1.1</VDIXMLVersion>
  <ReturnSet>LAST</ReturnSet>
  <DeviceList enc:id="DeviceList"
    xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    enc:itemType="xsd:string" enc:arraySize="2">
    <Item>DeviceID1</Item>
    <Item>DeviceID2</Item>
  </DeviceList>
</ws:GetDex>
```

2.4 Return

GetDex message returns an XML string that contains a VDIReturn block and a collection of DEX records according to the values of the GetDex parameters. If neither DeviceList nor OutletList are specified, the server will return DEX records for all data collection points (points of sale) that correspond to the specified CustomerID and ProviderID. If no time range is specified (that is, OnOrBefore and OnOrAfter are both omitted), the server will return only DEX records for the last 48 hours.
Example return XML:

Following is an example GetDex return message:

```xml
<?xml version="1.0" encoding="utf-8"?>
<VDITransaction VDIXMLVersion="1.1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   TransactionReason="GetDEX"
   TransactionID="2byBm1f1MkC7oZyz"
   TransactionTime="2011-07-28T12:34:56"
   ProviderID="ExampleProvider"
   CustomerID="ExampleCustomerID"
   ApplicationID="ExampleApp"
   ApplicationVersion="1.2.33.4">
   <VDIReturn>
     <Code>0</Code>
     <Message>Success</Message>
   </VDIReturn>
   <DEXList DEXEncoding="0"
      DEXCompressionType="" DEXCompressionParam="">
     <DexTransmission DeviceID="DeviceID1"
                    TransmitTime="2011-07-28T11:17:04" GMTOffSet="-5">
       <DexCollection>
         <DEX ReadDateTime="2011-07-28T16:08:00"
              GMTOffSet="0" DexReason="0" DexType="0"
              ResponseCode="OK">
           <RawDEX>
             DXS*9259630001*VA*V1/1*1**100
             DXE*1*1
           </RawDEX>
         </DEX>
       </DexCollection>
     </DexTransmission>
     <DexTransmission DeviceID="DeviceID2"
                    TransmitTime="2011-07-28T11:18:36" GMTOffSet="-5">
       <DexCollection>
         <DEX ReadDateTime="2011-07-28T16:08:00"
              GMTOffSet="0" DexReason="0" DexType="0"
              ResponseCode="OK">
           <RawDEX>
             DXS*9253420001*VA*V1/1*1**100
             DXE*1*1
           </RawDEX>
         </DEX>
       </DexCollection>
     </DexTransmission>
   </DEXList>
</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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionID</td>
<td>String(16)</td>
<td>Yes</td>
<td></td>
<td>Unique request identifier from the device. Used to prevent 'echo' data if a retry is sent from the device to the Server. Server returns an error and rejects the message if a duplicate TransactionID is detected.</td>
</tr>
<tr>
<td>ProviderID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Name of the provider of the source data, for example inOne or MEI sending DEX.</td>
</tr>
<tr>
<td>CustomerID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Name of the bottler or operator in which the data belongs, like BestFamilyVending.</td>
</tr>
<tr>
<td>ApplicationID</td>
<td>String(32)</td>
<td>Yes</td>
<td></td>
<td>Identifies the provider application which called UploadDex Web Method.</td>
</tr>
<tr>
<td>ApplicationVersion</td>
<td>String(8)</td>
<td>Yes</td>
<td></td>
<td>Version of the application which called UploadDex Web Method</td>
</tr>
<tr>
<td>DEXEncoding</td>
<td>Integer</td>
<td>Yes</td>
<td>0 – ‘None’ 1 – ‘ISO8859-1’ 2 – ‘UTF-8’</td>
<td>Type of encoding used to encode DEX files. Can be only ISO8859-1 or UTF-8.</td>
</tr>
<tr>
<td>DEXCompressionType</td>
<td>String(32)</td>
<td>No</td>
<td>0 – ‘None’, or implementation-specific code</td>
<td>Type of Compression used. Can be omitted or ‘NONE’ to flag that no compression was used.</td>
</tr>
<tr>
<td>DEXCompressionParam</td>
<td>String</td>
<td>No</td>
<td></td>
<td>Additional compression information if anything required for decompression.</td>
</tr>
<tr>
<td>UserData</td>
<td>String</td>
<td>No</td>
<td></td>
<td>User defined data, if any.</td>
</tr>
<tr>
<td>VDXXMLVersion</td>
<td>String(8)</td>
<td>Yes</td>
<td></td>
<td>Version of VDXXML sent as VDXXML parameter (1.0 initially)</td>
</tr>
<tr>
<td>VDXXML</td>
<td>XML</td>
<td>Yes</td>
<td></td>
<td>XML String representing collection of DEX Files with additional attributes. See format below.</td>
</tr>
</tbody>
</table>
3.3 UploadDEX XML

**VDIXMLVersion** – This is a version of VDXXML. If VDXXML format will ever change VDXXML 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 VDXXMLVersion would identify what parser (version of VDXXML) to use.

**VDIXML** - XML String representing collection of DEX Files with additional attributes as in the following SOAP 1.2 example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<VDITransaction VDIXMLVersion="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 TransactionReason="UploadDEX"
 TransactionID="2bC7oZzyBM1f1Mk"
 TransactionTime="2011-07-28T12:42:42"
 ProviderID="ExampleProvider"
 CustomerID="ExampleCustomerID"
 ApplicationID="ExampleApp"
 ApplicationVersion="1.2.33.4">
 <DEXList DEXEncoding="0" DEXCompressionParam="">
   <DexTransmission DeviceID="DeviceID1"
     TransmitTime="2011-07-28T11:17:42" GMTOffSet="-5">
     <DexCollection>
       <DEX ReadDateTime="2011-07-28T16:08:00" GMTOffSet="0" DexReason="0" DexType="0" ResponseCode="OK">
         <RawDEX>
           DXS*9259630001*VA*V1/1*1**100
           DXE*1*1
         </RawDEX>
       </DEX>
     </DexCollection>
   </DexTransmission>
 </DEXList>
</VDITransaction>
```
3.4 Return
UploadDEX message returns an XML string in the standard VDI return structure, for example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<VDITransaction VDIXMLVersion="1.1">
  <VDIReturn>
    <Code>0</Code>
    <Message>Success</Message>
  </VDIReturn>
</VDITransaction>
```

Chapter 4 XML Tags and Attributes Descriptions

4.1 Tags Descriptions

4.1.1 Common Tags
These tags apply to all VDI messages:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;VDITransaction&gt;</td>
<td></td>
<td>Yes</td>
<td></td>
<td>This is the Root Element for VDI transaction. All other elements are nested within it</td>
</tr>
<tr>
<td>&lt;UserData&gt;</td>
<td>String</td>
<td>No</td>
<td></td>
<td>This is an optional element within different levels of VDI XML and represents provider specific data.</td>
</tr>
<tr>
<td>&lt;OtherCollectionsOrLists&gt;</td>
<td>Collection</td>
<td>No</td>
<td></td>
<td>Just a place holder to show that other elements (like alarms, etc...) can go here.</td>
</tr>
</tbody>
</table>

4.1.2 DEX Tags
These tags apply to both GetDEX and UploadDEX messages.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;DEXList&gt;</td>
<td>Collection</td>
<td>No</td>
<td></td>
<td>This element is start for DEX related data transmission.</td>
</tr>
<tr>
<td>&lt;DexTransmission&gt;</td>
<td>Collection</td>
<td>No</td>
<td></td>
<td>This is an element within &lt;DEXList&gt;. 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. &lt;DEXList&gt; may have multiple &lt;DexTransmissions&gt; collections for the same or different devices. Context</td>
</tr>
</tbody>
</table>
of each individual <DexTransmission> will be compressed based on DEXEncoding and DEXCompressionParam parameters of UploadDex method. This is an element within <DexTransmission> and 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.

### 4.2 Attributes Description

#### 4.2.1 Common Attributes

These tags apply to all VDI messages.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Attribute Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDITransaction</td>
<td>VDXMLVersion</td>
<td>String(8)</td>
<td>Yes</td>
<td>Version of VDXML used in XML</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>TransactionReason</td>
<td>String(32)</td>
<td>Yes</td>
<td>Reason for transaction. It can be a message type, e.g. “GetDEX”, “UploadDEX”, or any other reason for this particular transaction</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>TransactionID</td>
<td>String(16)</td>
<td>Yes</td>
<td>Unique identifier for transaction. Used to prevent transmitting or processing of the same transaction.</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>TransactionTime</td>
<td>Date/Time</td>
<td>Yes</td>
<td>Time when this transaction was transmitted</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>ProviderID</td>
<td>String(32)</td>
<td>Yes</td>
<td>Name of the provider of the source data, for example inOne or MEI sending DEX.</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>CustomerID</td>
<td>String(32)</td>
<td>Yes</td>
<td>Name of the bottler or operator in which the data belongs, like BestFamilyVending.</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>ApplicationID</td>
<td>String(32)</td>
<td>Yes</td>
<td>Identifies the provider application which created transaction</td>
</tr>
<tr>
<td>VDITransaction</td>
<td>ApplicationVersion</td>
<td>String(8)</td>
<td>Yes</td>
<td>Version of the application which created transaction</td>
</tr>
</tbody>
</table>
### 4.2.2 DEX Attributes

These attributes apply to both GetDEX and UploadDEX messages.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Attribute Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEXList</td>
<td>RecordsCount</td>
<td>Integer</td>
<td>No</td>
<td>RecordsCount represents a number of transmissions within the list.</td>
</tr>
<tr>
<td>DEXList</td>
<td>DEXEncoding</td>
<td>Integer</td>
<td>Yes</td>
<td>Type of encoding used to encode DEX files. Can be only ISO8859-1 or UTF-8.</td>
</tr>
<tr>
<td>DEXList</td>
<td>DEXCompressionType</td>
<td>String(32)</td>
<td>No</td>
<td>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'.</td>
</tr>
<tr>
<td>DEXList</td>
<td>DEXCompressionParam</td>
<td>String</td>
<td>No</td>
<td>Additional compression information if anything required for decompression.</td>
</tr>
<tr>
<td>DexTransmission</td>
<td>DeviceID</td>
<td>String(32)</td>
<td>Yes</td>
<td>ID of the device that this &lt;DEXTransmission&gt; is for.</td>
</tr>
<tr>
<td>DexTransmission</td>
<td>TransmitTime</td>
<td>Date/Time</td>
<td>Yes</td>
<td>Date/time of this particular transmission. Value is local date/time of transmission. Offset between local time of transmission place and GMT (LocalTime – GMTTime)</td>
</tr>
<tr>
<td>DexTransmission</td>
<td>GMTOffSet</td>
<td>Integer</td>
<td>Yes</td>
<td>Offset between device local time and GMT (LocalTime – GMTTime)</td>
</tr>
<tr>
<td>DexCollection</td>
<td>RecordsCount</td>
<td>Integer</td>
<td>No</td>
<td>RecordsCount attribute represents a number of DEX files within the DexCollection.</td>
</tr>
<tr>
<td>DEX</td>
<td>ReadDateTime</td>
<td>Date/Time</td>
<td>Yes</td>
<td>Date/time when Dex was read. This is device local Date/Time. Offset between device local time and GMT (LocalTime – GMTTime)</td>
</tr>
<tr>
<td>DEX</td>
<td>GMTOffSet</td>
<td>Integer</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DEX</td>
<td>FileSize</td>
<td>Long</td>
<td>No</td>
<td>Size of DEX file in bytes</td>
</tr>
<tr>
<td>DEX</td>
<td>DexReason</td>
<td>Integer</td>
<td>Yes</td>
<td>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 – Fulfils Specific Request 5 – VMC initiated 6 – Dex from external Device (Driver HH or similar)</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
<td>Notes</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------</td>
<td>-------------</td>
<td>----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>DexType</td>
<td>'Full' dex was sent/taken or only specific portion of DEX stream.</td>
<td>Integer</td>
<td>Yes</td>
<td>0 – Full Dex 99 – Other</td>
</tr>
<tr>
<td>ResponseCode</td>
<td>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.</td>
<td>String(128)</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Chair – Chris Lilly, Best Vendors Technology Group
Coordinator – Michael Kasavana, Michigan State University/NAMA
NAMA Board Liaison – Mark Stein, Mark Vend

1-Anant Agrawal (Cantaloupe Systems)
2-Mandeep Arora (Cantaloupe Systems)
3-Tammy Baker (Cantaloupe Systems)
4-Louis Beaudoin (Cantaloupe Systems)
5-Rebecca Boyle (Apriva)
6-Glenn Butler (CTO Services)
7-Jim Canter (Crane)
8-Jeff Doerr (Flextronics)
9-Don Finley (MEI Group)
10-Justin Grant (Cantaloupe Systems)
11-Doug Haddon (MEI Group)
12-Dennis Hammer (Seaga Manufacturing)
13-Ron Hoormann (CoinCo)
14-Tomo Howell (Coca-Cola)
15-Tenoah Hunt (Coca-Cola)
16-Glen Johnson (Vendors Exchange)
17-Michael Kasavana (Michigan State University)
18-Mark Kronenberg (CompuVend)
19-Chris Lilly (Best Vendors Technology Group)
20-Dan Mathews (NAMA)
21-Jeff Mayoros (Wittern)
22-Steve Merwarth (Coca-Cola)
23-Bud Nixon (Compass-USA)
24-Chris Norris (Validata)
25-Gene Ostendorf (InOne Technology)
26-Paresh Patel (Courtesy Vending)
27-Warren Philips (Validata)
28-Anton Rakushkin (Crane-Streamware)
29-Mary Rampe (MEI Group)
30-Bill Robertson (Apriva)
31-Cary Sagady (USA Technologies)
32-Mark Stein (Mark Vend)
33-David Saracini (Sprout Retail)
34-Leandro Valdez (Apriva)
35-Bob Williams (Compass-USA)