



# WorkXpress API Documentation



## Table of Contents

<i>Description of use for the WorkXpress API</i> .....	3
Exposed API Functions.....	4
Compatibility Level.....	4
Authentication.....	5
Function: LookupData .....	6
Function: AddItem .....	11
Function: UpdateItem.....	15
Function: ExecuteAction .....	20
<i>Map Builder</i> .....	23
<i>Action: Third Party Web Service</i> .....	24
<i>WorkXpress API Data Formats</i> .....	25
<i>Display Format Parts</i> .....	31

## Description of use for the WorkXpress API

The WorkXpress API exposes the complete WorkXpress Engine using only four simple functions. These functions include:

- LookupData
- AddItem
- UpdateItem
- ExecuteAction

Each exposed function has the same three simple parameters and returns response XML containing the data requested. These parameters include:

- API version
- Authentication Code
- Request XML.

Before each call is made, the client program must assemble an appropriate Request XML string. Then, a connection to the WorkXpress API must be made through the SOAP WSDL. A SOAP connection object must be instantiated, and finally, the call is made.

This call returns a Response XML document, which needs to be parsed by the client program, to extract its requested data.

A complete specification of the request and response XML documents for each exposed API function follows. All code examples are in PHP and will need to be modified for the appropriate language.

For PHP developers there is a PEAR package that makes communicating with WorkXpress easier. For more information, please see <http://www.workxpress.com//sites/default/files/Services%20WorkXpress.pdf>.

## Exposed API Functions

In most programming languages making an API call requires the client application to first instantiate a connection to the server, to a specific WSDL file. The WSDL file defines what functions and data structures are available for consumption through the API.

The URL for the WorkXpress WSDL is:

<http://example.workxpress.com/api/api.php?wsdl>

Once the object is instantiated against the WorkXpress WSDL, all of the functions detailed below are available on the connection object, and can be called like any other function in the client language.

*Example:*

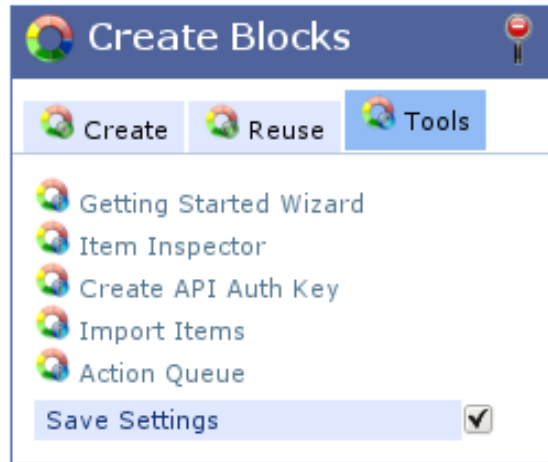
```
$soap = new  
SoapClient('http://example.workxpress.com/api/api.php?wsdl');  
$response = $soap->UpdateItem(1, $auth_code, $xml);
```

## Compatibility Level

**This document is written for compatibility level 1.**


## Authentication

Before you can use the WorkXpress API, an authentication key must be generated for the Application. To generate an auth key on any project, click the “Crete API Auth Key” from the Tools tab in the Block Creator:



The build tools are not available on testing and production applications. However, this page can be accessed via [http://example.workxpress.com/im\\_tools/create\\_auth\\_key.php](http://example.workxpress.com/im_tools/create_auth_key.php).

Once the page loads, three Fields will be rendered. A user to associate the key with must be provided as well as that user's password. Once the Layout has been saved with the appropriate data, the Layout will reload and the Auth Key Field will be an editable Text Area with the authentication key. Auth keys cannot be retrieved. If an auth key is lost it must be regenerated from within WorkXpress.



User to associate the key with	HarrisData
Password of the selected user	
Auth Key	
<input type="button" value="Save"/>	

## Function: LookupData

LookupData is a function for reading from the WorkXpress database. The Request XML defines what Item Types and Relations should be looked up, as well as the Fields to retrieve from each Item/Relation. The engine reads this definition, looks up the data requested and returns a similar structure containing the requested data.

**Request XML** – Every XML document contains a root node, with blocks of nested child nodes. Each node opens, contains attributes and child nodes, and then closes. Below is a description of the node types, their attributes and the valid child node types.

**<wxRequest></wxRequest>** - The root node for the Request XML document.

**<dataSet></dataSet>** - Contained inside of the wxRequest node, the dataSet node contains the information required for a single lookup. You may have as many data sets as you want, allowing you to combine many lookups into one call.

*dataSet Attributes:*

reference	String	An identifier that will be returned with the response to identify different data sets.
-----------	--------	--

**<items></items>** - Contained inside of the dataSet node, the items node contains the information required to identify the item types that you want to lookup, and tells the API it will be working with items.

**<item></item>** - Contained inside of the items node, the item node identifies a single item within the data set.

*item Attributes:*

itemId	String	The id of the Item to lookup.
--------	--------	-------------------------------

**<map></map>** - Contained inside of the items node, the map node contains the information required to lookup each item type, including the item type id and any filters.

**<definition></definition>** - Contained inside of the map node, the definition node holds the XML definition of the map. The XML must have any HTML entities encoded; this is done in PHP by passing the XML to htmlentities().

*\* For more information on building maps, see the “Map Builder” section below.*

**<fields></fields>** - Contained inside of the dataSet and relation nodes, the fields node is the parent node for the fields to retrieve from.

**<field></field>** - Contained inside of the fields node, the field node defines a single Field to retrieve data from.

*field Attributes:*

fieldId	String	Id of the Field.
reference	String	An identifier that will be returned with the response to identify each Field.

**<format></format>** - Contained inside of the field node, the format node defines the format of the data to return for the selected Field. This node may hold a string of display format parts used to define a format. If no format is provided, the full Field value will be returned. For more information please see the section on display format parts at the end of this document.

*format Attributes:*

type	String	<p>There are three different options for the format of the data:</p> <p>html – Includes any HTML used when rendering the non-editable Field inside of the WorkXpress Application.</p> <p>stored – The format of the Field as it is stored in the WorkXpress database.</p> <p>text – Returns the value as plain text. This is the recommended format.</p>
------	--------	--

**<relations></relations>** - Contained inside of the dataSet node, the relations node is the parent node for any relationships to look up for the items that were previously defined.

**<relation></relation>** - Contained inside of the relations node, the relation node defines a single Relation to lookup.

*relation Attributes:*

relationType	String	Id of the Relation Type.
from	String	<p>Defines which side of the Relation to start from. Valid values are:</p> <p>base – The base side of the Relation Type.</p> <p>target – The target side of the Relation Type.</p>

reference	String	An identifier that will be returned with the response to identify each Relation Type.
-----------	--------	---

**Response XML** - Below is a description of the valid nodes returned in the Response XML from a LookupData request.

**<wxResponse></wxResponse>** - The root node for the Response XML document.

**<callStatus></callStatus>** - Contained inside of the wxResponse node, the callStatus node contains the status of the SOAP call.

*callStatus Attributes:*

status	String	The call's status. Values include success and failure.
--------	--------	--

**<compatibilityLevel></compatibilityLevel>** - Contained inside of the wxResponse node, the compatibilityLevel node contains the version of the API that was used.

**<dataSet></dataSet>** - Contained inside of the wxResponse node, one data set is returned for each data set in the request document.

*dataSet Attributes:*

reference	String	The reference that was defined for the data set in the request document.
-----------	--------	--

**<item></item>** - Contained inside of the dataSet node, the item node identifies a single item within the data set.

*item Attributes:*

ItemId	String	The id of the current item.
--------	--------	-----------------------------

**<field></field>** - Contained inside of the item or relation node, the field node contains data about a single field on the current Item or Relation.

*field Attributes:*

ItemId	String	Id of the Field.
reference	String	The reference that was defined for the field in the request document.

**<value></value>** - Contained inside of the field node, the value node holds the value for the current Field.

**<relation></relation>** - Contained inside of the item node, the relation node contains data about a single relationship from the current item.



*relation Attributes:*

reference	String	The reference that was defined for the Relation Type in the request document.
id	String	Id of the current Relationship.
relationType	String	Relation Type id of the current Relationship.
baseItemId	String	Item Type id of the base Item.
baseItemType	String	Item Type id of the base Item.
targetItemId	String	Item Type id of the target Item.
targetItemType	String	Item Type id of the target Item.

**Examples:**

*An example of a basic LookupData Request might use the following XML :*

```

<wxRequest>
  <dataSet reference="accounts">
    <items>
      <map>
        <definition>
          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;wxQuery
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="wxQuery.xsd"
            id="root"&gt;&lt;data
              for="root"&gt;&lt;item/&gt;&lt;/
              data&gt;&lt;startingTypes&gt;&lt;startingType&
              gt;a35234&lt;/startingType&gt;&lt;/
              startingTypes&gt;&lt;/wxQuery&gt;
          </definition>
        </map>
      </items>
    <fields>
      <field fieldId="a66969" reference="name">
        <format type="text" />
      </field>
    </fields>
    <relations>
      <relation relationType="a36495" from="base"
        reference="account_to_contact">
        <fields>
          <field fieldId="a36498" reference="position">
            <format type="text" />
          </field>
        </fields>
      </relation>
    </relations>
  </dataSet>
</wxRequest>

```

Corresponding response XML:

```
<wxResponse>
  <callStatus status="success" />
  <compatibilityLevel>1</compatibilityLevel>
  <dataSet reference="accounts">
    <item itemId="u7324">
      <field fieldId="a66969" reference="name">
        <value>WorkXpress</value>
      </field>
      <relation reference="account_to_contact" id="u7437"
        relationType="a36495"
        baseItemTypeId="a35234" baseItemId="u7324"
        targetItemTypeId="a35334"
        targetItemId="u7436">
        <field fieldId="a36498" reference="position">
          <value>Developer</value>
        </field>
      </relation>
    </item>
```

## Function: AddItem

AddItem is a function for creating new Items inside of WorkXpress. When adding Items through the WorkXpress API, any appropriate Item, Field and Relation Actions will be executed.

**Request XML** - Below is a description of the node types, their attributes and the valid child node types supported for an AddItem request.

**<wxRequest></wxRequest>** - The root node for the Request XML document.

**<dataSet></dataSet>** - Contained inside of the wxRequest node , the dataSet node contains the information required for a item add request. You may have as many data sets as you want, allowing you to combine many add items requests into one call.

*dataSet Attributes:*

reference	String	An identifier that will be returned with the response to identify different data sets.
-----------	--------	--

**<item></item>** - Contained inside of the dataSet node, the item node contains the Item Type id of the Item being added.

*item Attributes:*

itemTypeId	String	The Item Type id of the item being created.
------------	--------	---

**<fields></fields>** - Contained inside of the dataSet and relation nodes, the fields node is the parent node for the Fields to set on the Item or Relation previously defined.

**<field></field>** - Contained inside of the fields node, the field node defines a single Field to store data into.

*field Attributes:*

fieldId	String	Id of the Field.
---------	--------	------------------

**<value></value>** - Contained inside of the field node, the value node holds the value to store into the Field.

**<relations></relations>** - Contained inside of the dataSet node, the relations node is the parent node for any Relationships that should be added when the Item is added.

**<relation></relation>** - Contained inside of the relations node, the relation node defines a single Relation to create.

*relation Attributes:*

action	String	The action to perform on the Relationship. This should be set to “add” when adding a Relationship.
oppositeItemId	String	The Item Id of the Item that you wish to relate the Item you are adding to.
reference	String	An identifier that will be returned with the response to identify each Relationship that was created.
relationType	String	Id of the Relation Type you would like to create.
startingSide	String	Defines which side of the Relation the Item being added will be on. Valid values are:  base – The new Item is on the base side.  target – The new Item is on the target side.

**Response XML** - Below is a description of the valid nodes returned in the Response XML from an AddItem request.

**<wxResponse></wxResponse>** - The root node for the Response XML document.

**<callStatus></callStatus>** - Contained inside of the wxResponse node, the callStatus node contains the status of the SOAP call.

*callStatus Attributes:*

status	String	The call's status. Values include success and failure.
--------	--------	--

**<compatibilityLevel></compatibilityLevel>** - Contained inside of the wxResponse node, the compatibilityLevel node contains the version of the API that was used.

**<dataSet></dataSet>** - Contained inside of the wxResponse node, One data set is returned for each data set in the request document.

*dataSet Attributes:*

reference	String	The reference that was defined for the data set in the request document.
-----------	--------	--

**<item></item>** - Contained inside of the dataSet node, defines an Item that was added to the WorkXpress application.

*item Attributes:*

itemId	String	Id of the Item that was added.
ItemTypeid	String	Item Type id of the Item that was added.

**<relation></relation>** - Contained inside of the item node, the relation node defines a Relation that was added to the WorkXpress application.

*relation Attributes:*

reference	String	The reference that was defined for the Relation Type in the request document.
relationId	String	Id of the Relationship.

**Examples:**

*An example of an AddItem request might use the following XML:*

```
<wxRequest>
  <dataSet reference="workxpress">
    <item itemTypeId="a35234" />
    <fields>
      <field fieldId="a66969">
        <value>WorkXpress</value>
      </field>
      <field fieldId="a36314">
        <value>http://www.workxpress.com</value>
      </field>
    </fields>
    <relations>
      <relation action="add" oppositeItemId="u7436"
reference="account_to_contact"
        relationType="a36495" startingSide="base">
        <fields>
          <field fieldId="a36498">
            <value>Developer</value>
          </field>
        </fields>
      </relation>
    </relations>
  </dataSet>
</wxRequest>
```

*Corresponding response XML:*

```
<wxResponse>
  <callStatus status="success" />
  <compatibilityLevel>1</compatibilityLevel>
  <dataSet reference="workxpress">
    <item itemId="u7563" itemTypeId="a35234">
      <relation reference="account_to_contact" relationId="u7564"
    />
    </item>
  </dataSet>
</wxRequest>
```

## Function: UpdateItem

UpdateItem is called to perform a number of different tasks on existing Items in WorkXpress. These tasks include:

- Set Fields on Items & Relations
- Recycle Items & Relations
- Restore previously recycled Items & Relations
- Delete Items & Relations
- Create Relations to Items

The Request XML defines Fields to store, as well as Relations to create. The engine reads this definition, performs its' tasks and then returns an item node for each Item effected, along with relation nodes for each Relation that was added or updated. Actions attached to any Items, Fields or Relations affected by the call will be run.

**Request XML** - Below is a description of the node types, their attributes and the valid child node types supported for an UpdateItem request.

**<wxRequest></wxRequest>** - The root node for the Request XML document.

**<dataSet></dataSet>** - Contained inside of the wxRequest node, the dataSet node contains the information required to perform the requested operation. You may have as many data sets as you want, allowing you to combine many operations into one call.

*dataSet Attributes:*

reference	String	An identifier that will be returned with the response to identify different data sets.
action	String	The operation to perform on the Item. Valid values are:  delete – Deleted Items are completely removed from WorkXpress and cannot be retrieved.  recycle – Recycled Items are <i>not</i> removed from WorkXpress and can be restored.  restore – Restores a previously recycled item.  update – Updates an

		existing Item.
--	--	----------------

**<items></items>** - Contained inside of the dataSet node, the items node contains the information required to identify the Items you wish to update.

**<item></item>** - Contained inside of the items node, the item node identifies a single item in the data set.

*item Attributes:*

itemId	String	The id of the Item to perform the operation on
--------	--------	--

**<map></map>** - Contained inside of the items node, the map node contains the information required to lookup each item type, including the item type id and any filters.

**<definition></definition>** - Contained inside of the map node, the definition node holds the XML definition of the map. The XML must have any HTML entities encoded; this is done in PHP by passing the XML to htmlentities().

*\* For more information on building maps, see the "Map Builder" section below.*

**<fields></fields>** - Contained inside of the dataSet and relation nodes, the fields node is the parent node for the Fields to be updated.

**<field></field>** - Contained inside of the fields node, the field node defines a single Field to retrieve data from.

*field Attributes:*

fieldId	String	Id of the Field
---------	--------	-----------------

**<value></value>** - Contained inside of the field node, the value node holds the value to store into the Field.

**<relations></relations>** - Contained inside of the dataSet node, the relations node is the parent node for any Relationships that should be added or updated.

**<relation></relation>** - Contained inside of the relations node, the relation node defines a single Relation to add or update.

*relation Attributes:*

action	String	The action to perform on the Relationship. Valid values are:  add – Creates a new Relationship.  update – Updates an existing
--------	--------	---



		<p>Relationship.</p> <p>delete – Deleted Relationships are completely removed from WorkXpress and cannot be retrieved.</p> <p>recycle – Recycled Relationships are <i>not</i> removed from WorkXpress and can be restored.</p> <p>restore – Restores a previously recycled Relationship.</p>
oppositeItemId	String	The Item Id of the Item that you wish to relate the Item you are updating to. If the action is not set to add, this will be used to find an existing Relationship.
reference	String	An identifier that will be returned with the response to identify each Relationship that was created or updated.
relationType	String	Id of the Relation Type you would like to create.
startingSide	String	<p>Defines which side of the Relation the Item being updated will be on. Valid values are:</p> <p>base – The Item will be on the base side.</p> <p>target – The Item will be on the target side.</p>

**Response XML** - Below is a description of the valid nodes returned in the Response XML from an UpdateItem request.

**<wxResponse></wxResponse>** - The root node for the Response XML document.

**<callStatus></callStatus>** - Contained inside of the wxResponse node, the callStatus node contains status of the SOAP call.

*callStatus Attributes:*

status	String	The call's status. Values include success and failure.
--------	--------	--

**<compatibilityLevel></compatibilityLevel>** - Contained inside of the wxResponse node, the compatibilityLevel node contains the version of the API that was used.

**<dataSet></dataSet>** - Contained inside of the wxResponse node, one data set is returned for each data set in the request document.

*dataSet Attributes:*

reference	String	The reference that was defined for the data set in the request document.
-----------	--------	--

**<item></item>** - Contained inside of the dataSet node, defines an Item that was updated.

*item Attributes:*

itemId	String	Id of the Item that was updated
--------	--------	---------------------------------

**<relation></relation>** - Contained inside of the item node, the relation node defines a Relation that was added or updated.

*relation Attributes:*

Reference	String	The reference that was defined for the Relation in the request document.
relationId	String	Id of the Relationship.

**Examples:**

*An example of a basic UpdateData request might use the following XML:*

```
<wxRequest>
  <dataSet action="update" reference="account">
    <items>
      <item itemId="u7563" />
    </items>
    <fields>
      <field fieldId="a66969">
        <value>WorkXpress</value>
      </field>
    </fields>
    <relations>
      <relation action="update" oppositeItemId="u7436"
        reference="account_to_contact" relationType="a36495"
        startingSide="base">
        <fields>
          <field fieldId="a36498">
            <value>Intern</value>
          </field>
        </fields>
      </relation>
    </relations>
  </dataSet>
</wxRequest>
```

*Corresponding response XML:*

```
<wxResponse>
  <callStatus status="success" />
  <compatibilityLevel>1</compatibilityLevel>
  <dataSet reference="account">
    <item itemId="u7563">
      <relation reference="account_to_contact" relationId="u7564" />
    </item>
  </dataSet>
</wxRequest>
```

## Function: ExecuteAction

ExecuteAction is called to run Actions that already exist in the WorkXpress Application on a set of Items. The request XML defines Items and individual Actions to execute. The engine reads this definition, performs its' tasks and then returns each Item that the Action was run on.

**Request XML** - Below is a description of the node types, their attributes and the valid child node types supported for a ExecuteAction request.

**<wxRequest></wxRequest>** - The root node for the Request XML document.

**<dataSet></dataSet>** - Contained inside of the wxRequest node , the dataSet node contains the information required to perform the requested operation. You may have as many data sets as you want, allowing you to combine many requests into one call.

*dataSet Attributes:*

reference	String	An identifier that will be returned with the response to identify different data sets.
-----------	--------	--

**<items></items>** - Contained inside of the dataSet node, the items node contains the information required to identify the Items you wish to run the Actions on.

**<item></item>** - Contained inside of the items node, the item node identifies a single item in the data set.

*item Attributes:*

itemId	String	The id of the Item to run the Action on.
--------	--------	--

**<map></map>** - Contained inside of the items node, the map node contains the information required to lookup each item type, including the item type id and any filters.

**<definition></definition>** - Contained inside of the map node, the definition node holds the XML definition of the map. The XML must have any HTML entities encoded; this is done in PHP by passing the XML to htmlentities().

*\* For more information on building maps, see the "Map Builder" section below.*

**<actions></actions>** - Contained inside of the dataSet node, the actions node defines the Actions that should be run on the items that were previously defined.

**<action></action>** - Contained inside of the actions node, the action node defines a single Action to be executed.

*action Attributes:*

actionId	String	Id of the Action
----------	--------	------------------

**Response XML** - Below is a description of the valid nodes returned in the Response XML from an ExecuteActions request.

**<wxResponse></wxResponse>** - The root node for the Response XML document.

**<callStatus></callStatus>** - Contained inside of the wxResponse node, the callStatus the status of the SOAP call.

*callStatus Attributes:*

status	String	The call's status. Values include success and failure.
--------	--------	--

**<compatibilityLevel></compatibilityLevel>** - Contained inside of the wxResponse node, the compatibilityLevel node contains the version of the API that was used.

**<dataSet></dataSet>** - Contained inside of the wxResponse node, one data set is returned for each data set in the request document.

*dataSet Attributes:*

reference	String	The reference that was defined for the data set in the request.
-----------	--------	---

**<item></item>** - Contained inside of the dataSet node, defines a single Item that the Actions were run on.

*item Attributes:*

itemId	String	Id of the Item.
--------	--------	-----------------

**Examples:**

*An example of a basic ExecuteAction request might use the following XML:*

```
<wxResponse>
  <dataSet reference="accounts">
    <items>
      <item itemId="u3541" />
      <item itemId="u511" />
    </items>
    <actions>
      <action actionId="a314558" />
    </actions>
  </dataSet>
</wxResponse>
```

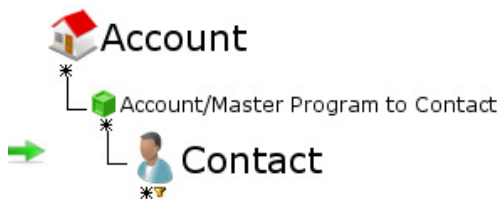
*Corresponding response XML:*

```
<wxResponse>
  <callStatus status="success" />
  <compatibilityLevel>1</compatibilityLevel>
  <dataSet reference="accounts">
    <item itemId="u3541" />
    <item itemId="u511" />
  </dataSet>
</wxResponse>
```

## Map Builder

WorkXpress provides a map builder to make requests easier to build. To access the map builder, click the “Build Maps For API Calls” link on the Tools tab of the Block Creator. The map builder works like a normal data lookup; however, after modifying any part of the map the XML definition is reloaded and displayed. The XML is rendered in two different formats: XML is the normal XML with all HTML entities visible, Encoded XML is the XML passed through PHP's htmlentities(). Encoded XML is the clean version of the map definition that must be used in any API call.

 The starting context is All Items of a type ([Change](#))






Preview	
LACROIX, BRIAN Braithwaite, Brian WEIST, BOB	Contact
XML	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;wxQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="wxQuery.xsd" id="root"&gt;&lt;startingTypes&gt; &lt;startingType&gt;a5543&lt;/startingType&gt;&lt;/startingTypes&gt;&lt;paramGroup id="wx4a6476f43e03c"&gt;&lt;join&gt;and&lt;/join&gt;&lt;toRelation id="wx4a6476f440284"&gt; &lt;relationTypes&gt;&lt;relationType&gt;&lt;relationTypeId&gt;a15151&lt;/relationTypeId&gt; &lt;from&gt;target&lt;/from&gt;&lt;/relationType&gt;&lt;/relationTypes&gt;&lt;join&gt;and&lt;/join&gt; &lt;requirement&gt;must&lt;/requirement&gt;&lt;paramGroup id="wx4a6476f4405df"&gt; &lt;join&gt;and&lt;/join&gt;&lt;toItem id="wx4a6476f440660"&gt;&lt;itemTypes&gt;&lt;itemType&gt; &lt;itemTypeId&gt;a5614&lt;/itemTypeId&gt;&lt;relationTypeId&gt;a15151&lt;/relationTypeId&gt; &lt;to&gt;base&lt;/to&gt;&lt;/itemType&gt;&lt;/itemTypes&gt;&lt;requirement&gt;must&lt;/requirement&gt;</pre>
Encoded XML	<pre>&amp;lt;?xml version="1.0" encoding="UTF-8"?&amp;gt; &amp;lt;wxQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="wxQuery.xsd" id="root"&amp;gt;&amp;lt;startingTypes&amp;gt; &amp;lt;startingType&amp;gt;a5543&amp;lt;/startingType&amp;gt;&amp;lt;/startingTypes&amp;gt;&amp;lt;paramGroup id="wx4a6476f43e03c"&amp;gt;&amp;lt;join&amp;gt;and&amp;lt;/join&amp;gt;&amp;lt;toRelation id="wx4a6476f440284"&amp;gt;&amp;lt;relationTypes&amp;gt;&amp;lt;relationType&amp;gt;&amp;lt;relationTypeId&amp;gt;a15151&amp;lt;/relationTypeId&amp;gt; &amp;lt;from&amp;gt;target&amp;lt;/from&amp;gt;&amp;lt;/relationType&amp;gt;&amp;lt;/relationTypes&amp;gt;&amp;lt;join&amp;gt;and&amp;lt;/join&amp;gt; &amp;lt;requirement&amp;gt;must&amp;lt;/requirement&amp;gt;&amp;lt;paramGroup id="wx4a6476f4405df"&amp;gt; &amp;lt;join&amp;gt;and&amp;lt;/join&amp;gt;&amp;lt;toItem id="wx4a6476f440660"&amp;gt;&amp;lt;itemTypes&amp;gt;&amp;lt;itemType&amp;gt; &amp;lt;itemTypeId&amp;gt;a5614&amp;lt;/itemTypeId&amp;gt;&amp;lt;relationTypeId&amp;gt;a15151&amp;lt;/relationTypeId&amp;gt;</pre>

## Action: Third Party Web Service

Within the WorkXpress Engine, there is a Web Service Action Type, found under the Third Party category. This Action Type allows the Application to make a call to any SOAP web service with a publicly accessible WSDL. After entering the WSDL location, WorkXpress will read the WSDL to determine the functions that are available to be called and display a select box with those options. Upon an option being chosen, WorkXpress will then display the input parameters with a Formula to populate each one. The output parameters will be displayed below the input parameters, with a map to choose a storage location for each one. These parameters must be simple types such as strings, numbers and booleans.

Third Party

Webservice




Get the USD to Euro conversion rate.

Enter the URL to a Web Service (Ent



Choose a Service to call

**This function wants to know (Input Parameters) :**

ConversionRate-FromCurrency :  USD

ConversionRate-ToCurrency :  EUR

**And it will tell you (Output) :**

ConversionRateResponse-ConversionRateResult :  Conversion Rate on  SO 1000 (a  Sales Order)

Set SOAP Client Timeout  
*In Seconds*

For more information on the Web Service Action Type, please see <http://www.workxpress.com/training/third-party-webservice>.



## WorkXpress API Data Formats

The WorkXpress Engine tries hard to store data in simple, easy, logical formats. Below is a description of some of the less straightforward Field Types, and what format their data is expected in.

### Item Pickers

An Item Picker is a Field Type that stores a reference to another Item. These fields can be used to manage Relationships, create links to other Items or simply store a reference. For more information on Item Pickers, please visit <http://www.workxpress.com/training/item-picker-select-one>.

Item Pickers are split into two storage types: single and multi. Single Item Pickers can only store a single Item in the format below:

`<itemTypeId>|<itemId>` (ex. e8|u1)

Multi Item Pickers use a similar format with the addition of a comma to delimit multiple values:

`<itemTypeId>|<itemId>,<itemTypeId>|<itemId>` (ex. e8|u1,e8|u58)

### Select

Select Fields use Items known as Select Options to populate their available values. These Fields store the id of the Select Option(s) that was selected, much like Item Pickers. However, attempting to store the title or alternate title of a valid Select Option will result in the value being converted to the proper Select Option id before storage. This allows you to pass in the value "Pounds" instead of having to know the Select Option id. For more information on Select Fields, please visit <http://www.workxpress.com/training/select-one>.

Much like Item Pickers, Select Fields are split into single and multi. Single Select Fields can only store a single Select Option using the format below:

`e11|<selectOptionId>` (ex. e11|a36789) or  
`<title>` (ex. Pounds) or  
`<altTitle>` (ex. lbs)

Multi Select Fields use a similar format with the addition of a comma to delimit multiple values:

```
e11|<selectOptionId>,e11|<selectOptionId> (ex. e11|a36789,e11|a36791)
or
<title>,<title> (ex. Pounds,Feet) or
<altTitle>,<altTitle> (ex. lbs,ft)
```

### Check Box

A Check Box Field works much like a boolean value, it is either on or off. When a Check Box is “on”, the value is stored as a 1. When the Field is “off”, the value is blank. To turn a Check Box Field on, any non-empty value may be passed in to the API. To turn a Check Box Field off, the value should be empty. For more information on Check Box Fields, please visit <http://www.workxpress.com/training/checkbox>.

### Date, Time, and Date Time

All Date Fields are stored as a Unix time stamp. This number represents the number of seconds since the Unix Epoch (January 1, 1970). In PHP, you can convert any standard date or date time format into a Unix time stamp:

```
// convert a date value
$date = '01/19/1985';
$time_stamp = strtotime($date);

// convert a date time value
$date = '01/19/1985 11:35 AM';
$time_stamp = strtotime($date);

// convert a time value
$time = '11:35 AM';
$time_stamp = strtotime('January 1, 1970 '.$time);
```

For more information on Date Fields, visit <http://www.workxpress.com/training/date>.

### File Attachment

File Attachment Fields are what is known as a multi-part Field. These Fields are stored as XML that separate different parts of the full value. In order to set a File Attachment Field through the API you will need to base 64 encode the binary file and place that into the encoded file part of the XML. In PHP you would use the following code:

```
$file_path = '/path/to/file.odf';
$encoded_file = base64_encode(file_get_contents($file_path));
```

For more information on File Fields, please visit <http://www.workxpress.com/training/file-attachment>.

File Attachment Fields use the following XML format:

`<multi_part_field></multi_part_field>` - Root node for all multi-part Fields.

`<part></part>` - Defines the value for a single part of the Field.

*Attributes:*

id	String	<p>The name of the part. Valid values include:</p> <p>filename – The name to be given to the file, including extension.</p> <p>mime_type – The file's mime type.</p> <p>size – The file's size in bytes.</p> <p>encoded_file – The base 64 encoded file.</p>
----	--------	--

Example:

```
<?xml version="1.0"?>
<multi_part_field>
  <part id="filename">image.png</part>
  <part id="mime_type">image/png</part>
  <part id="size">364544</part>
  <part id="encoded_file">encodedFile</part>
</multi_part_field>
```

**Address**

Address Fields are also multi-part Fields. Address Fields can be one of two different types: US and International. This type will need to be defined when setting an Address Field through the API. For more information on Address Fields, please see <http://www.workxpress.com/training/address>.

Address Fields use the following XML format:

`<multipart_field></multi_part_field>` - Root node for all multi-part Fields.

`<part></part>` - Defines the value for a single part of the Field.

*Attributes:*

<p>id</p>	<p>String</p>	<p>The name of the part. Valid values include:</p> <p>street – The value for the first street part.</p> <p>street2 – The value for the second street part.</p> <p>street3 – The value for the third street part, only used for International addresses.</p> <p>city – The value for the city.</p> <p>state – The value for the state, up to three characters for International and two characters for United States.</p> <p>zip_code – The value for the postal code. Should be numeric for United States addresses.</p> <p>country – The value for the country. Should be the countries full name or the ISO 3166-1 alpha-3 formatted country code (see <a href="http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3">http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3</a>).</p> <p>type – Should be either International or United States.</p> <p>sort_value – Which street value should be used to sort this field. Should be either street, street2 or street3. Only used for International address.</p>
-----------	---------------	---

**Example:**

```
<?xml version="1.0"?>
<multi_part_field>
  <part id="street">Ostvorstadt</part>
  <part id="street2">Hauptstraße 5</part>
  <part id="street3"></part>
  <part id="city">Musterstadt</part>
  <part id="state"></part>
  <part id="zip_code">01234</part>
  <part id="country">Germany</part>
  <part id="type">International</part>
  <part id="sort_value">street2</part>
</multi_part_field>
```

**Phone Number**

Phone Number Fields are also multi-part. Like Address Fields, Phone Numbers can be either International or United States. For more information on Phone Number Fields, visit <http://www.workxpress.com/training/phone-number>.

Phone Number Fields use the following XML format:

**<multipart\_field></multi\_part\_field>** - Root node for all multi-part Fields.

**<part></part>** - Defines the value for a single part of the Field.

**Attributes:**

id	String	<p>The name of the part. Valid values include:</p> <p>area_code – The value for the area code. Should be three digits for United States and up to five alphanumeric characters for International.</p> <p>prefix – For United States phone numbers, this is the three digits immediately following the area code. This is not used for International numbers.</p> <p>line_number – For United States phone numbers, this is the last four digits of the number. For International phone numbers, this is the entire number after the area code.</p>
----	--------	--

		<p><b>extension</b> – The value for the extension (if any).</p> <p><b>country_code</b> – The country calling code for International Phone Numbers. Should be 1 for United States,</p> <p><b>type</b> – Should be either International or United States.</p>
--	--	---

**Example:**

```
<?xml version="1.0"?>
<multi_part_field>
  <part id="area_code">717</part>
  <part id="prefix">609</part>
  <part id="line_number">0029</part>
  <part id="extension">123</part>
  <part id="country_code">1</part>
  <part id="type">United States</part>
</multi_part_field>
```

## Display Format Parts

WorkXpress provides several display format parts available through the API. These parts can be used to format Field values from the LookupData function. For example, say you want to format an address as follows:

453 Lincoln Street - Carlisle, PA 17013

You could use a display format part as the value for the format node as follows:

```
<format type="text">Street - City, State ZipCode</format>
```

You can use display format parts for either html or text formatted Fields. When using the stored format, display format parts are ignored. Below is a list of the display format parts available for each Field Type.

### Address

- **Type** – The type of address, either United States or International
- **Street** – The first street input
- **Street2** – The second street
- **Street3** – The third street of an International address
- **City**
- **State**
- **ZipCode** – The postal code
- **Country**

### Currency

- **NumberOnly** – Returns the value without the preceding dollar sign (\$)

### Phone

- **Type** – The type of phone number, either United States or International
- **CountryCode**
- **AreaCode**
- **Prefix** – The three digits immediately following the area code in a United States phone number
- **LineNumber** – The remaining digits of the phone number
- **Extension**

### Select – Select One

- **AltTitle** – Returns the alternate title of the Select Option currently stored as the Field's value
- **WithOther** – Returns the title of the Select Option along with the value of the "Other" Field associated with the Select Option separated by a hyphen (if one exists).
- **OtherOnly** – Returns only the value of the "Other" Field associated with the Select Option.

### Check Box

- **Checked** – Returns the word “CHECKED” if the Field is checked and “UNCHECKED” if it is not.
- **FieldLabel** – Returns the Field's label if it is checked and nothing if it is not.
- **WithOther** – Returns the same as the “Checked” display format part with the value of the “Other” Field associated with the Field's current value separated by a hyphen (id one exists).
- **OtherOnly** – Returns only the value of the “Other” Field associated with the Field's current value

### Date and Date Time

- **AMPMLowercase** – Returns the appropriate am/pm value.
- **AMPMUppercase** – Returns the appropriate AM/PM value.
- **DayOfMonth** - Day of the month without leading zeros
- **DayOfMonthSuffix** – English ordinal suffix for the day of the month, 2 characters
- **DayOfMonthLeadingZero** – Day of the month, 2 digits with leading zeros
- **DayOfWeek** – A full textual representation of the day of the week
- **DayOfWeekAbbreviated** – A textual representation of a day, three letters
- **DayOfWeekNumber** – ISO-8601 numeric representation of the day of the week
- **DayOfYear** – The day of the year (starting from 0)
- **Hour** – 12-hour format of an hour without leading zeros
- **Hour24** – 24-hour format of an hour without leading zeros
- **Hour24LeadingZero** – 24-hour format of an hour with leading zeros
- **HourLeadingZero** – 12-hour format of an hour with leading zeros
- **Minute** - Minutes with leading zeros
- **MonthName** – A full textual representation of a month
- **MonthNameAbbreviated** – A short textual representation of a month, three letters
- **MonthOfYear** – Numeric representation of a month, without leading zeros
- **MonthOfYearLeadingZero** – Numeric representation of a month, with leading zeros
- **Second** – Seconds, with leading zeros
- **Timestamp** – Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)
- **WeekOfYear** – ISO-8601 week number of year, weeks starting on Monday
- **Year** – A full numeric representation of a year, 4 digits
- **YearShort** – A two digit representation of a year

### File

- **DownloadURL** – URL to download the file. If the Field is *not* set to allow the File to be public, this URL will require a login.
- **Filename** – The name of the file, including extension
- **FileSize** – The size of the file in bytes
- **Height** – The height in pixels if the file is an image
- **Image** – Returns an HTML image tag if the file is an image
- **MimeType** – The file's mime type



- **Thumbnail** – Returns an HTML image tag to the file's thumbnail if it is an image
- **ThumbnailURL** – URL to retrieve the thumbnail if the file is an image. If the Field is *not* set to allow the File to be public, this URL will require a login.
- **Width** – The width in pixels if the file is an image

### **Social Security Number**

- **LastFour** – Returns only the last four digits of the social security number