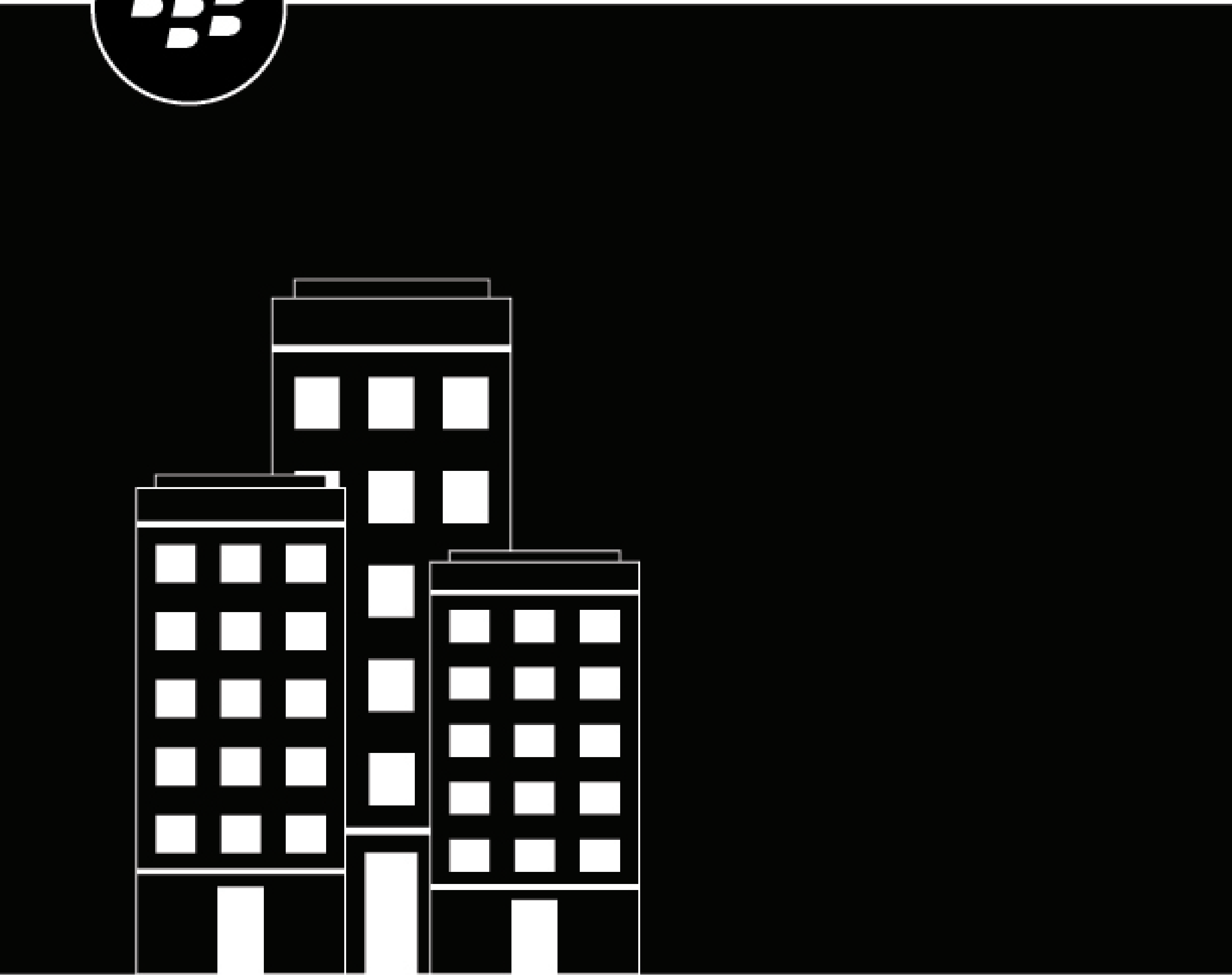


BlackBerry AtHoc

User Sync Client Guidelines

7.6

Contents

What is the BlackBerry AtHoc User Sync Client?

The BlackBerry AtHoc User Sync Client is a command line tool that enables you to synchronize user information from an LDAP data source or CSV file to the BlackBerry AtHoc system. Synchronization of user contact information with an external source ensures that user information in the BlackBerry AtHoc system is up to date with external sources and critical alerts are delivered to intended users.

The BlackBerry AtHoc User Sync Client replaces the LDAP Data Integration Module and CSV Importer Tool that were provided by BlackBerry AtHoc for synchronizing user information. Instead of providing two separate tools, the functionality of these tools is merged into a single tool.

Supported upgrade paths from LDAP and CSV Importer Tool

LDAP 1.2.7

- If you are moving from LDAP 1.2.7, which uses the SDK, you need to migrate to the User Sync Client version 1.0.0. For instructions, see Appendix B: Migrate from LDAP Sync Client version 1.2.7 to BlackBerry AtHoc User Sync Client.

LDAP 2.0

- If you are moving from the most recent version of LDAP (version 2.0) the configuration is fully compatible.

CSV Importer Tool

- If you are moving from the CSV Importer Tool, you need to follow the configuration steps detailed in this document. See Appendix C: Differences between the CSV Importer Tool and the BlackBerry AtHoc User Sync Client.

User synchronization process overview

The BlackBerry AtHoc User Sync Client is a Windows console application which can be executed by a Windows domain user or a Windows task scheduler. Using sets of configuration files, the user sync client tool can obtain data from a CSV file or an LDAP server, perform necessary transformation using XSL files, and synchronize data with the BlackBerry AtHoc System and optionally send out an email using SMTP configuration to specified recipients.

The BlackBerry AtHoc User Sync Client can be scheduled by the Windows task scheduler to execute at a specified time interval or manually executed by a user.

The tool processes data using three components; the data adapter, data processor, and data synchronizer.

Data adapter

The data adapter is a component of the user sync client that can access CSV files kept in a local folder or remote LDAP server accessible from where the tool is run. The data adapter takes configuration inputs and outputs an XML file which is then used by the Data processor.

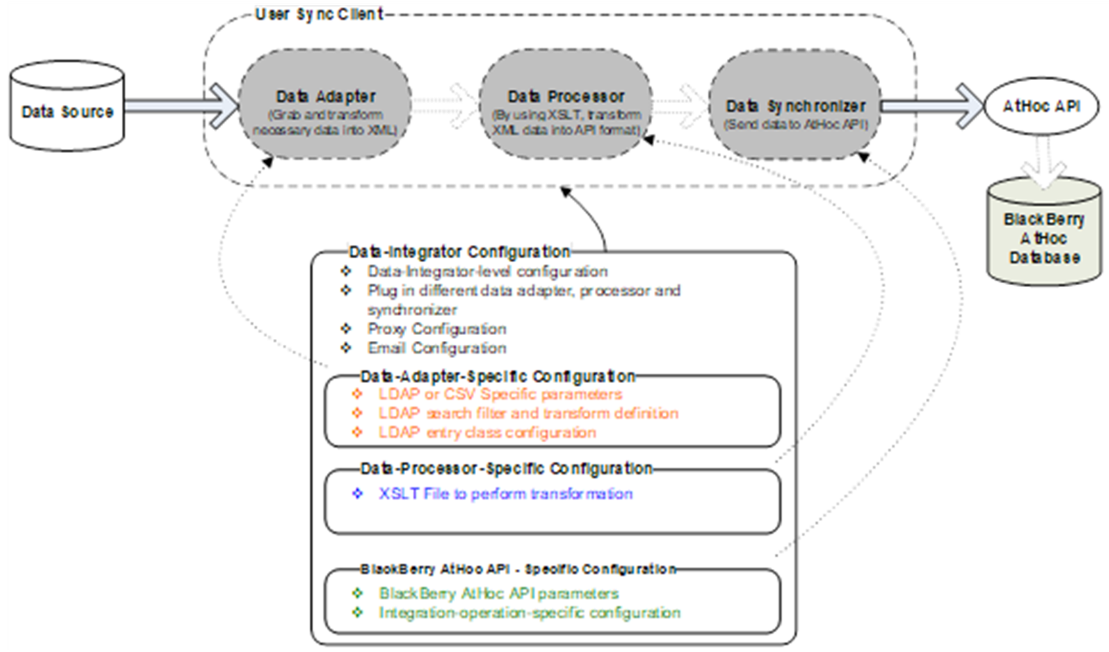
Data processor

The data processor is a component of the user sync client that consumes the intermediate XML file generated by the data adapter and XSL file that you provide to map necessary fields from the data source to BlackBerry AtHoc user fields. You can write different transformation logic in an external XSL file to meet your requirements. This step also generates an XML file suitable for use by the data synchronizer.

Data synchronizer

The data synchronizer is a component of the user sync client that consumes the intermediate XML file generated by the data processor and prepares it to be sent to BlackBerry AtHoc web API interface.

The following diagram shows an overview of the synchronization process with configuration interaction:


User Sync Client
Data Source
Data Adapter
(Grab and transform necessary data into XML)
Data Processor
(By using XSLT, transform XML data into API format)
Data Synchronizer
(Send data to AtHoc API)
AtHoc API
BlackBerry AtHoc Database
Data-Integrator Configuration
Data-Integrator-level configuration
Plug in different data adapter, processor and synchronizer
Proxy Configuration
Email Configuration
Data-Adapter-Specific Configuration
LDAP or CSV Specific parameters
LDAP search filter and transform definition
LDAP entry class configuration
Data-Processor-Specific Configuration
XSLT File to perform transformation
BlackBerry AtHoc API - Specific Configuration
BlackBerry AtHoc API parameters
Integration-operation-specific configuration

Supported data sources

The BlackBerry AtHoc User Sync Client supports LDAP servers and CSV files as data sources. When you use an LDAP server as a data source, you can synchronize users, hierarchies, and distribution lists from LDAP to BlackBerry AtHoc. A CSV data source supports synchronizing only user data. The following table lists the capabilities for LDAP and CSV data sources.

Capability	LDAP data source	CSV data source
User data	Yes	Yes
Organization hierarchy structure	Yes	No
Distribution lists	Yes	No
Distribution list user membership	Yes	Yes (as part of user data)
Nested distribution lists	Yes	No
Proxy configuration	Yes	Yes
Send status email	Yes	Yes
Multiple rounds of integration	Yes	Yes
Support for enterprise user management	Yes	Yes

System requirements

Operating system

- Windows Vista and higher client operating system, or Windows Server 2008 SP2 or higher server operating system
- .NET 4.5 or higher installed

BlackBerry AtHoc system

- 7.6 or later release

LDAP directory

- For full search capability, the LDAP directory must support the Paged Results Control of LDAP v3 (OID: 1.2.840.113556.1.4.319).
- An LDAP directory that does not support Paged Results Control depends on the search result size limit setting of the directory server. For example, if the number of users in a single OU exceeds the size limit, the LDAPSync module does not obtain all users in the OU.
- To synchronize LDAP groups, the LDAP server must support the Attribute Scoped Query Control of LDAP v3 (OID: 1.2.840.113556.1.4.1504).

Install and set up the BlackBerry AtHoc User Sync Client

To install and set up the BlackBerry AtHoc User Sync Client, complete the tasks in the following sections.

Provision the BlackBerry AtHoc API client

Data Synchronization in the BlackBerry AtHoc User Sync Client uses the BlackBerry AtHoc API to synchronize users. The client is authenticated using industry standard OAuth 2.0 protocol before allowing access to call APIs. You must have a Client ID and Client Secret provisioned for your instance of the BlackBerry AtHoc User Sync Client.

Before you begin:

You must have organization administrator, enterprise administrator, or system administrator permissions to provision the BlackBerry AtHoc API client. You must have system administrator permissions to enable a provisioned application.

This task is not required if you already have a Client ID and Client Secret.

To provision the BlackBerry AtHoc API client, complete the following steps:

1. Log in to the BlackBerry AtHoc management system as an organization administrator, enterprise administrator, or system administrator.
2. In the navigation bar, click the ⚙ (Settings) icon.
3. In the System Setup section, click **API Applications**. The API Applications window opens.
4. Click **New**. The New API Application window opens.
5. Enter a name for the API integration.
6. (System administrators only) Next to Status, select the **Enabled** check box.
7. In the Authentication section, select the Password Grant Type.
8. Click **Save**. A Success message appears that includes the Client ID and Client Secret.
9. Take note of the displayed Client ID and Client Secret. It is displayed only once and will need to be regenerated if lost.

Set up an organization code in the BlackBerry AtHoc system

The BlackBerry AtHoc User Sync Client needs the organization code to synchronize users to a specific organization in the BlackBerry AtHoc System. Complete the following task to set up an organization code for your organization in the BlackBerry AtHoc management system. This organization code is not propagated to PSS. If you already have an organization code in PSS, use that one to complete this task.

Note: This task is not required if an organization code for your organization has already been provided to you.

To set up an organization code for your organization in the BlackBerry AtHoc management system, complete the following steps:

1. Log in to the BlackBerry AtHoc management system as a System Administrator.
2. Switch to the specific organization.
3. Go to **Settings** > **General Settings**.
4. In the Organization Details section, enter the organization code. Do not use spaces.
5. Click **Save**.

Configure the BlackBerry AtHoc system

To configure the BlackBerry AtHoc system, complete the following tasks:

1. Create all custom fields to be synchronized and assign unique common names to them.
2. Create a user with operator permissions.
3. Give the new user SDK User and End User Manager roles.

Install the BlackBerry AtHoc User Sync Client

To install the BlackBerry AtHoc User Sync Client, complete the following tasks:

1. Obtain the latest User Sync Client MSI from BlackBerry AtHoc Customer Support.
2. Run the MSI as an administrator.
3. Follow the prompts to install the BlackBerry AtHoc User Sync Client in the directory of your choice.

Once installed, the BlackBerry AtHoc User Sync Client has the following structure in the installed directory:

1. ***bin*** **folder**—This folder has the libraries (DLLs) needed for executing the BlackBerry AtHoc User Sync Client.
2. ***sample*** **folder**—This folder has two sub folders: **CSV** and **LDAP**. Each sub folder has a configuration file and several sample files for your reference. You can use these sample files to configure the BlackBerry AtHoc User Sync Client for either the CSV or LDAP data sources.
3. ***tools*** **folder**—This folder contains other useful tools.
4. ***AtHocDataIntegration.config file***—This is the main configuration file for the BlackBerry AtHoc User Sync Client where you can specify the data adapter, data processor, and data synchronizer configurations.
5. ***AtHocDataIntegrator.exe.config*** **file**—This is the configuration file used for the executable. Do not modify this file.
6. ***AtHocDataIntegrator.exe*** **executable**—This is the main executable file. You can run this file manually or through Windows Task Scheduler. Before running this file, ensure that your configurations and XSLT files are in place.
7. ***Sample.xslt*** **XSL file**—This is a file that is used to transform the XML generated by the data adapter to XML suitable for the data processor. This sample file is for the LDAP data adapter. A sample file for CSV is present in sample/csv folder. You must replace this file with the proper XSL file that matches your transformation requirements.

Execute the BlackBerry AtHoc User Sync Client

This section explains how to configure the data integration configuration file before executing the BlackBerry AtHoc User Sync Client. This section describes the purpose and results of using the three major components of the BlackBerry AtHoc User Sync Client and provides sample XML code.

Configure BlackBerry AtHoc User Sync Client components

The BlackBerry AtHoc User Sync Client consists of the following components:

1. **Data adapter**—Accesses the data source, either CSV or LDAP, to obtain information that is specified by the configuration.
2. **Data processor**—Uses the XSLT to transform XML data into a BlackBerry AtHoc-friendly format and prepares data for the web API data synchronizer.
3. **Web API data synchronizer**—Sends data to the BlackBerry AtHoc web API. User data is split into several web API packages to adapt the capability of the web API interface.

Complete the following steps to configure the User Sync Client components:

1. If required, configure system wide settings for SMTP email and Proxy settings in the configuration file.
2. Set the log file folder path. Decide if you want to save the interim XML files. You should use the “false” value for the <deleteInterimFiles> node for first time integration.
3. Disable all sections by setting the “enabled” attribute to “false”. This configuration enables you to test each component separately by enabling them as required.
4. Configure the data adapter section of the integration file under the <dataAdapter> node. Set the enable attribute on the <dataAdapter> node to “true”. Configure any other nodes under this node according to the requirements.
5. Run the AtHocDataIntegrator.exe file.
6. Check the XML output file produced by the data integrator.
7. If everything looks good in the XML output, create a new XSLT file to write the transformation logic for the file generated by the data integrator.
8. Specify the file path to the XSLT file.
9. Run the AtHocDataIntegrator.exe file again.
10. Check the XML output file produced by data processor.
11. Configure <dataSynchronizer> node according to requirements.
12. Set the “enable” attribute to “true.”
13. Run the AtHocDataIntegrator.exe again.
14. If everything is correct, the users, hierarchies, and distribution lists should be synchronized correctly.

After you finish:

For more information about configuring the BlackBerry AtHoc User Sync Client for a CSV or LDAP data source, see Configure the data integration file.

Test the output of each component

You can test output of each component by setting <deleteInterimFiles> node to false and incrementally enabling each section by setting enable=true. This outputs the interim XML files in the folder you specified for <tempFolderPath> under the <systemSettings> node.

Get status by email

The User Sync Client can send email after each execution using the SMTP server details you provide. You need following information to send an email:

- SMTP server address
- Username and password for the SMTP server
- Recipients email addresses
- The email address from which you want to send an email
- The name of the sender

You can configure these options under the <userMailService> node under <systemSettings> in the configuration file.

Logging

Each execution of the BlackBerry AtHoc User Sync Client generates a log file which is named using the timestamp and the .log extension. The log file is placed in the temporary folder that is specified by the configuration. This log file contains detailed processing information. In addition, the LDAP module generates a system event log entry to report process summary and error information. The source of the event log entry is AtHoc::DataIntegration.

Configure the data integration file

The BlackBerry AtHoc User Sync Client uses an XML-based configuration file, AtHocDataIntegration.config, to configure different components of the client. This file is usually kept in the same folder as the AtHocDataIntegrator.exe file. You can change the location of this file. If you want to load the file from a different location, provide the full path of the file as an argument when you run the AtHocDataIntegrator.exe file.

Configuration overview

The following code outlines the structure of the configuration file. Each node in this XML file is described in detail below.

```
<AtHocDataIntegration>

  <systemSettings>
    <tempDataPath>tempdata/</tempDataPath>
    <deleteInterimFiles>true|false</deleteInterimFiles>
    <!—Other system settings goes here -->
  </systemSettings>

  <integrations>

    <!-- One round integration -->
    <integration>

      <dataAdapter assembly="AtHoc.DataIntegration.dll"

 class="AtHoc.DataIntegration.Adapter.ActiveDirectory.AdDataAdapter"
                   outputFile="adapter.xml">

        <!-- Data-Adapter-Specific configuration -->

      </dataAdapter>

      <dataProcessor assembly="AtHoc.DataIntegration.dll"
                     class="AtHoc.DataIntegration.Processor.XsltDataProcessor"
                     inputFile="adapter.xml"
                     outputFile="processor.xml">

        <!-- Data-Processor-Specific configuration -->

      </dataProcessor>

      <dataSynchronizer assembly="AtHoc.DataIntegration.dll"

 class="AtHoc.DataIntegration.Synchronizer.Sdk.SdkDataSynchronizer"
                        inputFile="processor.xml">
        <!-- Data-Synchronizer-Specific configuration -->

      </dataSynchronizer>

    </integration>

    <!-- Another round integration -->
    <integration>
```

```
      <!-- ... -->
    </integration>

  </integrations>

</AtHocDataIntegration>
```

Configuration node descriptions

<AtHocDataIntegration>—This is the root node of the configuration file. All other nodes must be kept inside this node.

<systemSettings>—This is the node where you can specify general settings that are used by the User Sync Client. The <systemSettings> node has following sub nodes:

- **<tempDataPath>**—Specifies the temporary path to store logs and interim files.
- **<deleteInterimFiles>**—Indicates whether to delete interim files after integration.
- **<proxy>**—Used to configure proxy settings. The User Sync Client needs to access Web API URLs hosted on the cloud (for cloud deployments) to successfully run the synchronization. If the organization policy requires the use of a proxy server for outbound connections, you can configure those settings here. The <proxy> node has following sub nodes:
    - **<url>**—The URL of the proxy server provided to you by your organization administrator.
    - **<port>**—The port number used for the proxy server provided to you by your organization administrator.
    - **<username>**—The username for the proxy server user account. This username is provided to you by your organization administrator.
    - **<password isEncrypted=”false”>**—The password for the proxy server user account, provided to you by your organization administrator.
- **<useMailService>**—This node is used to configure an SMTP mail server for sending status emails. The <useMailService> node has following sub nodes:
    - **<smtpServer>**—Mail server address, provided to you by your organization administrator.
    - **<username>**—Username for the mail server used to send an email.
    - **<password>**—Password for the mail server used to send an email.
    - **<recipient>**—Semicolon-separated list of intended email receipients.
    - **<fromAddress>**—Email address used to send email.
    - **<fromName>**—Name that appears as the sender of an email.

<integrations>—The <integrations> node contains integration sections that consist of configurations for the data adapter, processor, and synchronizer.

<integration>—The <integration> node has following sub nodes:

- **<dataAdapter>**—The <dataAdapter> node contains data adapter-specific configuration.
    - The “assembly” attribute specifies the .NET assembly where the data adapter is located.
    - The “class” attribute specifies the full class name of the data adapter.
    - The optional “outputFile” attribute specifies the name of XML file the data adapter exports data to.
- **<dataProcessor>**—The <dataProcessor> node contains data processor-specific configuration.
    - The “assembly” attribute specifies the .NET assembly where the data processor is located.
    - The “class” attribute specifies the full class name of the data processor.
    - The optional “inputFile” attribute specifies the file the data processor reads data from.
    - The optional “outputFile” attribute specifies the file the data processor exports data to.
- **<dataSynchronizer>**—The <dataSynchronizer> node contains data synchronizer-specific configuration.
    - The “assembly” attribute specifies the .NET assembly where the data synchronizer is located.

- The “class” attribute specifies the full class name of the data synchronizer.
- The optional “inputFile” attribute specifies the file the data synchronizer reads data from.

Note: The “inputFile” and “outputFile” attributes are useful for testing a configuration but are not recommended for production. These attributes can specify a relative or absolute path. If the attributes are not specified, the User Sync Client generates file names based on the timestamp and places the files in the temporary data folder or it uses the filename from the previous step.

Tip: Configure multiple integration sections to synchronize data from multiple sources.

Data adapter configuration

The User Sync Client supports CSV and LDAP data sources for user synchronization. Data adapter configurations are different for the CSV data adapter and the LDAP data adapter. The following sections describe configurations for each data adapter.

LDAP data adapter configuration

Tip: Some parameter values can contain characters that are illegal in XML. This results in errors that may not indicate the illegal character but that will indicate the line number. In this case, surround the parameter value on that line with a CDATA section: `<parameter_name><![CDATA[value]]></parameter_name>`

The LDAP data adapter takes configurations specified in the configuration file as input and produces an XML output with a predefined structure. In the configuration file, the LDAP data adapter configuration has three major configuration nodes: <adParameters>, <hierarchyConfig> and <classConfig>. Each configuration node can have one or many sub nodes which are described below.

LDAP data adapter output XML

The LDAP data adapter produces an XML output like the one below.

Note: Output XML is highly dependent on configuration, but produces an output with the following general structure.

```
<IntegrationData from="AdDataAdapter">
  <hynode type="TREE" name="ABC Inc.">
    <attributes>
      <lineage />
    </attributes>
    <hynode type="TREE" name="Custom Support">
      <attributes>
        <lineage>/</lineage>
      </attributes>
      <hynode type="USER" name="CSR A">
        <attributes>
          <lineage>/Custom Support/</lineage>
          <sn>A</sn>
          <givenname>CSR</givenname>
          <samaccountname>csr_a</samaccountname>
          <displayname>CSR A Chief</displayname>
          <objectclass>
            <value>user</value>
            <value>organizationalPerson</value>
            <value>person</value>
            <value>top</value>
          </objectclass>
          <userAccountControl>66050</userAccountControl>
        </attributes>
      </hynode>
      <hynode type="STATICLIST" name="All Users">
       <attributes>
    <lineage>/AtHoc Users/Security Groups/</lineage>
    <name>AtHoc Users</name>
    <groupType>-2147483646</groupType>
    <mail>AtHocUsers@athoc.com</mail>
    <description />
    <cn>AtHoc Users</cn>
        </attributes>
        <members>
    <member type="USER">
      <samaccountname>jsmith</samaccountname>
    </member>
    <member type="USER">
      <samaccountname>ksmith</samaccountname>
    </member>
    <member type="STATICLIST">
      <cn>Engineering Users</cn>
    </member>
    <member type="STATICLIST">
      <cn>Sales and Marketing Users</cn>
    </member>
       </members>
      </hynode>
    </hynode>
  </hynode>
</IntegrationData>
```

LDAP data adapter node descriptions

<IntegrationData>—Top level node which contains all sub nodes.

- **<hynode>**—This node represents one of the following entities: TREE, STATICLIST, or USER. TREE nodes are hierarchy or group nodes. The STATICLIST node is for static distribution lists. The USER node is for a user. <hynode> can have other <hynode> nodes as children.
  - **<attributes>**—Each <hynode> node contains an <attributes> node. Attributes that were specified in the configuration file for each class type are found under the <attributes> node. The nodes under the <attributes> node are different based on configuration and type of hynode. See LDAP class-based configuration for instructions on how to specify attributes.
  - **<members>**—The <members> node is usually present for <hynode type="STATICLIST">. When present, this node has one or many <member> sub nodes which can have type="USER" or type="STATICLIST" to denote if the members of this list are users or other nested lists.

LDAP connection parameters

<adParameters>—The <adParameters> node is used to configure connection to the LDAP directory and has the following structure:

```
<adParameters>
     <server>LDAP_server[:port]</server>
     <username>username</username>
     <!--<password isEncrypted="true">CipherText</password>-->
     //After first run below will change to this
     <password>ClearText</password> //First run

     <authentication>
```

```
        Anonymous | Delegation | Encryption | FastBind |
        None | ReadonlyServer | Sealing | Secure |
        SecureSocketsLayer | ServerBind | Signing
      </authentication>

      <search>
        <pagingMode>none | paging</pagingMode>
      </search>

      <rootNodeDistinguishedName>
         LDAP Distinguished Name
      </rootNodeDistinguishedName>
   </adParameters>
```

LDAP connection parameter node descriptions

- **<server>**—An optional IP address or the LDAP server name. This field can be blank. If blank, the synchronization module searchs the default domain controller. For an SSL-secured LDAP, you must specify the fully-qualified domain name of the LDAP server.

  Specifying the port is optional. By default, the LDAP service port is 389 and the SSL-secured LDAP service port is 636.
- **<username>**—The username of the account that accesses the LDAP server. If the server is not specified, the username is ignored. For an SSL-secured LDAP server, you must specify a fully-qualified user DN (FQDN). For example, CN=User Common Name,OU=Admin Accounts,DC=SomeDomain,DC=com.
- **<password>**—The password of the user account which accesses the LDAP server. If the server is not specified, the password is ignored.
- **<isEncrypted>**—If the password is encrypted, its value is "true", otherwise it is "false". The default is “false”.
  - If the <isEncrypted> attribute is missing, the application assumes that the provided password is in clear text. The application encrypts the password and adds the <isEncrypted> attribute with the value "true".
  - If the <isEncrypted> attribute is set to true, the application assumes that the password is already encrypted and does nothing.
  - If the <isEncrypted> attribute is set to “false”, the application assumes that the password is in clear text and encrypts it. The application encrypts the password and sets the <isEncrypted> attribute to "true" and replaces clear text with cipher text.
- **<authentication>**—(Optional) A combination of authentication types used to access the LDAP server. It can be a combination of the following types (case-insensitive):
  - Anonymous
  - Delegation
  - Encryption
  - FastBind
  - None
  - ReadonlyServer
  - Sealing
  - Secure
  - SecureSocketsLayer
  - ServerBind
  - Signing

The authentication type defaults to “Secure”. When using a combination of multiple types, use a pipe (|) to separate them. For example, “Secure | FastBind”.

Common use case

- For standard Microsoft Active Directory operations, it is not necessary to specify the authentication type.
- For SSL-secured LDAP, use “SecureSocketsLayer”.

- **<search>**—(Optional) Specifies LDAP query-related parameters. Currently, there is only one parameter.
- **<pagingMode>**—Specifies how to use paging control in an LDAP search. Select from the following modes (case-insensitive):
  - **None**—Does not use pagination. This mode is usually for a Sun Directory, which does not support a paged result control. This mode depends on the size limit configuration of the LDAP server. If this limit is less than the number of items in one single OU, the LDAP module obtains items up to the server size limit.
  - **Paging**—This mode is for all LDAP servers which support a paged result search control (OID: 1.2.840.113556.1.4.319). This is the default and preferred mode and should be used for Microsoft Active Directory.
- **<rootNodeDistinguishedName>**—The distinguished name of the LDAP entry that corresponds to the hierarchy root node. If it is missing or blank, the sync module obtains the hierarchy from the root of the LDAP directory. For information about how to create the distinguished name, see the following URL: http://msdn2.microsoft.com/en-us/library/aa366101.aspx.

LDAP class-based configuration—<classConfig>

Important: The attribute used to map to the login_id field must have memberExport=“true”. The User Sync Client uses the login_id as the primary key to map users and distribution lists. You must have memberExport=“true” for the LDAP attribute that you intend to use as the login_id.

- **<classConfig>**—This node is used for mapping LDAP node types and BlackBerry AtHoc hierarchy node types. The LDAP data adapter produces one of the three node types: USER, TREE, and STATICLIST.

To view the sample output generated by LDAP data adapter see **samples/ldap/sample-output-LDAP-data-adapter.xml** in the installation folder of the User Sync Client.

The following example shows the structure of the <classConfig> node:

```
<!-- CLASS CONFIG

To map the LDAP object of the given object class to a BlackBerry AtHoc entity,
 user, or tree; to define which LDAP attributes of a given object class are
 returned.

 Attributes
 objectClass   : LDAP object class
 type          : BlackBerry AtHoc entity type; could be "USER" or "TREE"
 nameAttribute : Specify the value of the LDAP attribute to
                 be used as the name in resulting hierarchy.
-->

<classConfig>

  <class objectClass="user" type="USER">
    <attributes>
      <!--login_id-->
      <attribute memberExport="true">samaccountname</attribute>
      <!--last name-->
      <attribute>sn</attribute>
      <!--first name-->
      <attribute>givenname</attribute>
      <!--display name-->
      <attribute>displayname</attribute>
      <attribute>userAccountControl</attribute>
      <attribute multiValued="true">objectclass</attribute>
```

```
     </attributes>
   </class>
   <class objectClass="group" type="STATICLIST">
     <attributes>
       <attribute memberExport="true">cn</attribute>
       <attribute>name</attribute>
       <attribute>description</attribute>
     </attributes>
    </class>

   <class objectClass="*" type="TREE" />

</classConfig>
```

- **<class>**—The <class> sub node under <classConfig> is used to map a specific objectClass in LDAP to one of the USER, STATICLIST, or TREE nodes in the output produced by the LDAP data adapter. To map a specific objectClass, set the value of the objectClass attribute to the object classes defined in LDAP. Specify the LDAP data adapter type in the type attribute. For example, <class objectClass=”user” type=”USER”> maps users in LDAP to the USER node in the LDAP data adapter output.

  The <class> node has following sub node:

  - **<attributes>**—This sub node is used to encapsulate the attributes that you want to fetch for that object class from LDAP. You can have multiple <attribute> sub nodes under the <attributes> node. Each <attribute> tag specifies one LDAP attribute to fetch. The optional “multiValued” attribute specifies returning multiple values for this LDAP attribute. The optional “memberExport” attribute indicates that the current attribute should be exported as a static list member’s attribute.

The configuration code shown above directs the synchronization module to map LDAP users to user nodes and map all other LDAP entries to tree nodes.

This section also defines the custom LDAP attributes that need to be fetched and placed in the output XML of the LDAP data adapter.

LDAP Hierarchy-based configuration—<hierarchyConfig>

The <hierarchyConfig> node contains search filter definitions to create search filters for LDAP searches and transform definitions that direct the LDAP data adapter to transform the resulting hierarchy tree. The following XML segment shows the example hierarchy configuration:

```
<!-- Hierarchy-based configuration example -->
   <hierarchyConfig>
      <filters>
        <filter type="xxx" inheritable="true">…</filter>
        <filter type="xxx" inheritable="false">…</filter>
        <filter type="xxx" inheritable="true">…</filter>
      </filters>
      <transforms>
        <transform type="xxx" inheritable="true">…</transform>
        <transform type="xxx" inheritable="false">…</transform>
      </transforms>
      <node name="IT Department">
        <filters>
          <filter type="xxx" inheritable="false">…</filter>
        </filters>
        <transforms>
          <transform type="xxx" inheritable="true">…</transform>
        </transforms>
        <node name="Web Team">
          <filters>
            <filter type="xxx" inheritable="false">…</filter>
          </filters>
          <transforms>
            <transform type="xxx" inheritable="true">…</transform>
          </transforms>
        </node>
      </node>
    </hierarchyConfig>
```

<hierarchyConfig>—The <hierarchyConfig> node corresponds to the hierarchy root entry that is specified in the <adParameters> section. Each <node name="xxx"> corresponds to one of the LDAP root entry's descendants whose common name is “xxx”. The hierarchy relationship between these nodes is defined by their relative position in the XML. The "name" attribute of each node should be the same as the common name of the corresponding LDAP entry. Tree branches can be missing, but if any child node appears in this hierarchy configuration, the tree path to the root node should be complete and the same as it is in the LDAP directory.

Each node can have its own filter and transform definitions which are placed in the tags <filters> and <transforms>. Do not copy the entire LDAP tree into this configuration section and define specific filters and transforms for all nodes. This configuration section is inheritance-based. By setting the appropriate attribute value, any filter or transform defined in any node can be inherited by its child nodes, and they take effect if they are applicable for a specific child node.

You should define non-inheritable, node-specific <filters> and <transforms> but define common <filters> and <transforms> in a high-level node and apply them throughout its sub-tree.

Search filter definitions

The start tag of the filter definition is <filter>. Filters are used directly to create an LDAP search filter which searches for direct child nodes of the current node. Filters exclude unnecessary LDAP entries such as printers, computers, and contacts. There are three types of filters: attribute, date, and LDAP.

Filter examples

```
<filters>

  <!-- Filter 1 -->
  <filter type="attribute" target="objectclass" inheritable="true">
    <include>
      <value>user</value>
      <value>organizationalUnit</value>
      <value>container</value>
      <value>group</value>
    </include>
    <exclude>
      <value>computer</value>
    </exclude>
  </filter>
  <!-- Filter 2 -->
  <filter type="attribute" target="name" inheritable="true">
    <include>
      <value>*</value>
    </include>
    <exclude>
      <value>Computers</value>
     </exclude>
  </filter>
```

```
   <!-- Filter 3 -->
   <filter type="attribute" target="name" inheritable="false">
      <exclude>
         <value>DB Servers</value>
         <value>Web Servers</value>
      </exclude>
   </filter>


   <!-- Filter 4 -->
   <filter type="date" target="whenchanged" inheritable="true">
      <applicableClasses>
         <class>user</class>
      </applicableClasses>
      <from>7/1/2017</from>
      <to>07/18/2017</to>
      <within>1</within>
   </filter>

   <!-- Filter 5 -->
   <filter type="LDAP" appliedClass="group" inheritable="true">
     <!-- Include exchange enabled group only -->
     <![CDATA[
     (&
     (mail=*)
     (!msexchhidefromaddresslists=TRUE)
     )
     ]]>  </filter>

 </filters>
```

Each filter has two mandatory XML attributes: “type” and “inheritable,” and one optional attribute.

- **<type>**—Identifies the type of the filter. The possible values are: attribute, date, and LDAP.
- **<inheritable>**—Determines if the filter is inherited by child nodes. The value can be “true” or “false.” If a filter is set to inheritable=”true”, it will be run on the rootDistinguishedNode and all sub nodes. If it is set to “false”, it will only be run on the rootDistinguishedNode.
- **<appliedClass>**—Specifies the LDAP object class on which to apply the current filter. If no class is specified, the filter applies to all object classes. For example, to run a filter on user objects and not have the filter apply to group objects, set appliedClass=”user”.

Attribute filter

The applicable targets of the attribute filter are attributes of LDAP entries. Every LDAP entry (including user, organizational unit, container, group, and contact) has a set of attributes which store information for the entry. The attribute sets can vary among different LDAP entries.

The attribute filter has an XML attribute named “target” which identifies the attribute of the LDAP entry which the filters are applied to. The value of “target” must be the exact name of one of the LDAP entry’s attributes. If the name is different, the filter does not take effect.

Identify the set of LDAP attributes using the in-package utility application named AdTools.exe. For the synchronization process, the most useful LDAP attributes are “name” and “objectclass”.

The XML definition of the attribute filter may have two subsections, <include> and <exclude>, which define several <value> nodes by themselves.

The <include> section identifies the child LDAP nodes whose target attribute (with defined values) are included in search results. Note that the value "*" is a wildcard which means any value. A missing or empty <include> section includes all values by default. Empty values such as <value></value> are ignored.

The <exclude> section has the reverse effect of the <include> section. The <exclude> section identifies the child LDAP nodes whose “target” attribute (with defined values) are excluded. The value "*" means has no meaning in this context. However, if the value "*" is defined or inherited in the <include> section, defining "*" in the <exclude> section triggers the removal of the value "*" from both the <include> and <exclude> sections. A missing or empty <exclude> section, or no value defined, excludes nothing. Empty values are ignored.

During sync processing for one node, before applying “attribute” filters, all applicable “attribute” filters (either defined in this node or inherited from parents) with the same targeted LDAP attribute are merged to eliminate redundant and conflicting value definitions.

The following are example XML segments:

- **Filter 1** includes the LDAP entry whose “objectclass” (a type of LDAP entry) values are: “user”, “container”, “organizationalUnit” and “group”. Currently, only the tree node (organizationalUnit, container), and user and group (static list) are synchronized, so all LDAP entries of other types are excluded in the resulting hierarchy tree. Filter 1 is inheritable, so it applies to all child nodes.

```
<!-- Filter 1 -->
       <filter type="attribute" target="objectclass" inheritable="true">
         <include>
           <value>user</value>
           <value>organizationalUnit</value>
           <value>container</value>
           <value>group</value>
         </include>
         <exclude>
            <value>computer</value>
         </exclude>
       </filter>
```

- **Filter 2** includes an LDAP entry whose value of “name” (common name) can be any value except “Computers”. This filter is inheritable, so all LDAP entries with the name “Computers” are excluded. This filter may be useful if many nodes have a subentry whose name is “Computers”.

```
<!-- Filter 2 -->
    <filter type="attribute" target="name" inheritable="true">
      <include>
        <value>*</value>
      </include>
      <exclude>
       <value>Computers</value>
      </exclude>
    </filter>
```

- **Filter 3** is not inheritable; it excludes a child entry with the name “DB Servers” or “Web Servers” under the current LDAP node.

```
<!-- Filter 3 -->
   <filter type="attribute" target="name" inheritable="false">
      <exclude>
         <value>DB Servers</value>
         <value>Web Servers</value>
      </exclude>
   </filter>
```

- **Filter 4** is a date filter. This filter can apply to any <date> type LDAP attribute. The XML definition of the date filter can have two or three subsections: <applicableClasses>, <from> and <to>, or <within>.

  <applicableClasses>—Defines the LDAP object classes to which the date filter is applied.

  <from> and **<to>**—Define the start date and end date of the filter.

  If <from> and <to> do not exist, define a <within> tag and specify a number of days. This creates a date range from the specified number of previous days until today.

```
<!-- Filter 4 -->
   <filter type="date" target="whenchanged" inheritable="true">
      <applicableClasses>
         <class>user</class>
      </applicableClasses>
      <from>7/1/2007</from>
      <to>07/18/2007</to>
      <within>1</within>
   </filter>
```

- **Filter 5** is an LDAP filter. This type of filter enables defining a pure LDAP filter string and using it during the search process. It provides the most flexible and powerful way to define filters. LDAP filters can achieve the goals of all other filters, but this requires understanding LDAP. To learn how to create an LDAP query string, go to the following URL:http://msdn2.microsoft.com/en-gb/library/ms675768.aspx.

```
<!-- Filter 5 -->
   <filter type="LDAP" inheritable="true">
     <!-- Exclude All Disabled Users -->
     <![CDATA[
       (!(samaccounttype=268435456))
     ]]>
   </filter>
```

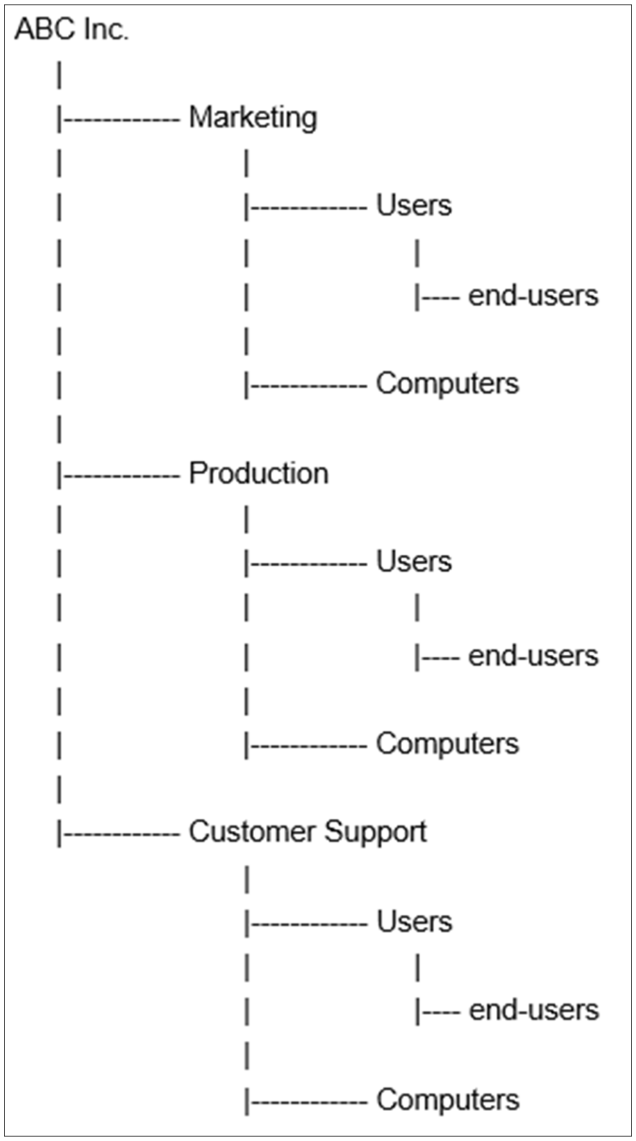
Transform definitions

The start tag of the transform definition is <transform>. Transforms are applied to the hierarchy tree resulting from the fetching and filtering process. Their purpose is to alter the tree structure to make it compatible with the BlackBerry AtHoc system. There are two types of transforms: “move” and "delete”. “Move” changes the parent node of specified node. “Delete” removes specified sub-trees.

```
<!—- Transform Examples -->
<transforms>
   <!—- Transform 1 -->
  <transform type="move" inheritable="true">
    <target>Users/*</target>
    <destination>..</destination>
  </transform>
  <!—- Transform 2 -->
  <transform type="delete" inheritable="true">
    <target>Users</target>
  </transform>
</transforms>
```

<transform>—Has the “inheritable” attribute (as does <filter>). Using this attribute enables the sync module to transform much of the common structure of sub-trees. Defining only one <transform> and making it non-inheritable will transform the sub-tree of a specified node.

Many LDAP administrators are likely to create common organizational structures in all LDAP branches. The following is a sample organizational structure:

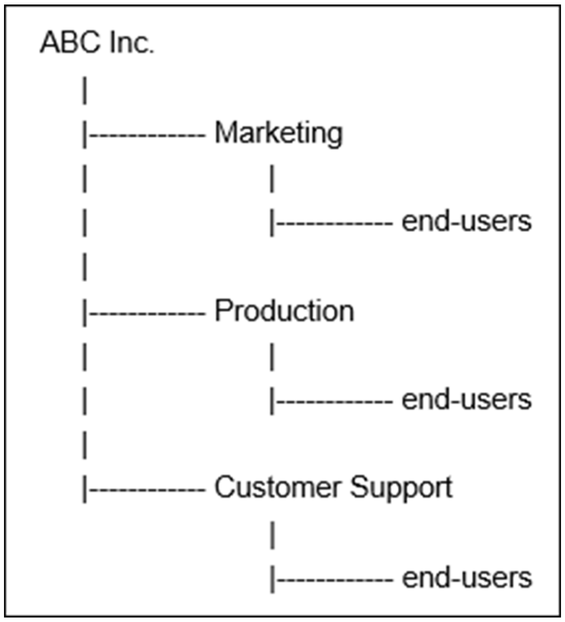


```
ABC Inc.
    |
    |------------ Marketing
    |                  |
    |                  |------------ Users
    |                  |                  |
    |                  |                  |---- end-users
    |                  |
    |                  |------------ Computers
    |
    |------------ Production
    |                  |
    |                  |------------ Users
    |                  |                  |
    |                  |                  |---- end-users
    |                  |
    |                  |------------ Computers
    |
    |------------ Customer Support
                       |
                       |------------ Users
                       |                  |
                       |                  |---- end-users
                       |
                       |------------ Computers
```

The ABC Inc. has three departments: Marketing, Production, and Customer Support. In each department, there are two sub-nodes which contain all computers and users, respectively. For synchronization purposes, place all users directly under the department nodes to which they belong. Use a filter to exclude the unnecessary “Computers” node. Use transforms to make the end user nodes compatible with the BlackBerry AtHoc system.

Transform 1 in the example above moves all end user nodes under the “Users” nodes to the appropriate department nodes.

Transform 2 cuts off empty “Users” nodes. After filtering and transforming, the resulting tree is shown below:



In the <transform> definition, the <target> tag defines the nodes to which this transform are applied. The inner text of the target element is a path-like string. It is a relative path starting from the current node. If no target is found, the transform does not take effect.In the <transform> definition, the <target> tag defines the nodes to which this transform are applied. The inner text of the target element is a path-like string. It is a relative path starting from the current node. If no target is found, the transform does not take effect.

Assume Transform 1 and 2 are defined in the "ABC Inc" node. When the process reaches the department nodes, Marketing, Production, and Customer Support, the inherited Transform 1 finds the target node "Users." It then moves all child nodes of "Users" to the appropriate nodes defined by the <destination> element. Assume Transform 1 and 2 are defined in the "ABC Inc" node. When the process reaches the department nodes, Marketing, Production, and Customer Support, the inherited Transform 1 finds the target node "Users." It then moves all child nodes of "Users" to the appropriate nodes defined by the <destination> element.

The <destination> element contains a path-like string, but it is a relative path starting from the parent nodes of the targeted nodes. In the transform's path definition, "*" means everything, including all nodes. ".." means go up one level. Any other string is considered a regular expression that matches node names. The delete <transform> has only a target element and it removes targeted nodes from the hierarchy tree.The <destination> element contains a path-like string, but it is a relative path starting from the parent nodes of the targeted nodes. In the transform's path definition, "*" means everything, including all nodes. ".." means go up one level. Any other string is considered a regular expression that matches node names. The delete <transform> has only a target element and it removes targeted nodes from the hierarchy tree.

Transforms take effect in order of declaration, from top to bottom and from parent to child. You must define them in a specific order to obtain the expected result. In the above example, if Transform 2 is before Transform 1, it will remove all "Users" nodes with all their end user nodes before Transform 1 takes effect.Transforms take effect in order of declaration, from top to bottom and from parent to child. You must define them in a specific order to obtain the expected result. In the above example, if Transform 2 is before Transform 1, it will remove all "Users" nodes with all their end user nodes before Transform 1 takes effect.

Tip: A regular expression in a <transform> definition should always start with ^ and end with $. Do not use the "/" character in the expression.

CSV data adapter configuration

The CSV data adapter takes configurations as an input and produces an output XML file that can be consumed by the data processor. The sample directory contains a sample CSV data adapter output XML file.

CSV data adapter sample output XML

The CSV data adapter generates output similar to the following example. Each output node is described after the example.

Note: The output is dependent on the configuration of the CSV data adapter. The following example output shows only the general structure.

```
<IntegrationData from="CsvDataAdapter">
<users>
  <user>
    <Username>jdoe</Username>
    <Firstname>John,Firstname</Firstname>
    <Lastname>Doe'Lastname</Lastname>
    <OrgH>/USA/</OrgH>
    <Date>01/11/2019</Date>
    <DateTime>01/11/2018 11:11:59</DateTime>
    <AllUsers>Yes</AllUsers>
  </user>
  <user>
    <Username>Janedoe</Username>
    <Firstname>Jane</Firstname>
    <Lastname>Doe</Lastname>
    <OrgH>/Germany/</OrgH>
    <Date>01/11/2019</Date>
    <DateTime>01/11/2018 11:11:59</DateTime>
    <AllUsers>Yes</AllUsers>
  </user>
</users>
</IntegrationData>
```

<IntegrationData>—This is a root node in the output XML. It has following sub nodes:

- **<users>**—This node represents all users that were read from CSV files. There is a separate <user> node in the <users> node for each user.
  - **<user>**—This node encloses other attributes for users. It can have one or many nodes as defined in the CSV data adapter configuration file. Each child node is one of the columns defined in the <csvColumns> node in the CSV data adapter configuration file.

CSV data adapter configuration details

The structure of CSV data adapter configuration is defined below.

```
<dataAdapter assembly="AtHoc.DataIntegration.dll"
 class="AtHoc.DataIntegration.Adapter.CSV.CsvDataAdapter" enable="true">

        <!-- CSV configuration goes here -->
        <skipFirstLines>1</skipFirstLines>

        <delimiter>,</delimiter>

        <readFolder>C:/tmp/</readFolder>
        <!-- Enable below line and specify path if processed files need to be
 moved to a different location: Optional -->

        <!--<processedFolder>C:/tmp/processedFolder/</processedFolder>-->
       <!-- Column names in the order they are presented in the CSV -->
        <csvColumns>
          <csvColumn>Username</csvColumn>
          <csvColumn>Firstname</csvColumn>
        <csvColumn>Lastname</csvColumn>
          <csvColumn>OrgH</csvColumn>
          <csvColumn>Date</csvColumn>
          <csvColumn>DateTime</csvColumn>
        <csvColumn>AllUsers</csvColumn>
        </csvColumns>
      </dataAdapter>
```

- **<skipFirstLines>**—This node specifies the number of lines to exclude from the CSV file while processing data. Set this value to 1 if the input CSV file contains headers. If the headers are spread over multiple lines, set this value to the number of lines that are used for headers.
- **<delimiter>**—This node specifies the delimiter used in the CSV file. The default value is comma (,). However, files with other delimiters are also supported. For example, the pipe (|) symbol.
- **<readFolder>**—Absolute path of the folder where the CSV files are located. The CSV data adapter can process all files present in a folder. You should keep only CSV files in this folder.
- **<processedFolder>**—(Optional) If specified, once the files are processed by the Data Adapter, they are moved to this location. The value must be an absolute path.
- **<csvColumns>**—The ordered list of user-defined names for columns to use as node names in the resulting XML file. This node has the following sub node:
  - **<csvColumn>**—The node name in the resulting XML file. You should keep this name simple and without spaces or special characters. Consider using a name that is a valid name for XML nodes. You will need this node name when writing the XSL for the Data Processor.

Data processor configuration

```
<xslt href="sample.xsl" />
```

The configuration section for the XSLT data processor has only one tag, <xslt>, to provide the information about the XSLT source. Place the file name in the href attribute to provide an external XSLT file. Alternatively, specify XSL code in the <xslt> tag.

The XSLT provided transforms XML from the data adapter into the XML used by the data synchronizer. All data mapping rules are implemented in the XSLT data processor. See the `Sample.xsl` file provided in the sample folder for each data source.

LDAP data processor

The LDAP data processor configuration uses an XSL file to transform the output from the LDAP data adapter to the output expected by the data synchronizer. The example output of the data adapter, sample XSL file and sample output from LDAP data processor are provided in the sample folder. You must write an XSL file which transforms the output from the LDAP data adapter to XML that can be consumed by the data processor.

Sample LDAP data processor output

The following is sample output produced by the LDAP data processor:

```
<!-- lastName is optional -->
<lastName>A</lastName>
<!-- enabled is optional -->
<enabled>N</enabled>
<!-— customFields, optional -->
<customFields>
<field commonName="PIN">1234</field>
<field commonName="NUMBER1_CN">94010</field>
</customFields>
<!-- Device address, optional -->
<devices>
<device commonName="workEmail">mail_test@athoc.com</device>
<device commonName="workPhone">650-111-0000</device>
</devices>
</user>
</users>
<staticlists>
    <list seqId="1" commonName="StaticDl5">
      <name>StaticDl5</name>
      <description></description>
      <members>
        <user login_id="csr_a" />
      </members>
    </list>
  </staticlists>
</IntegrationData>
```

LDAP data processor node descriptions

<IntegrationData>—This is a root node of the output file. The <integrationData> node has the following sub nodes:

- **<hierarchy>**—This node contains hierarchy node details. It has a <hynode> sub node for each level of hierarchy.
- **<users>**—This node contains user definitions. Under this node is a <user> node which represents one user that needs to be synchronized.
  - **<user>**—This node represents each node that will be synchronized. The following sub nodes can be present under the <user> node:
    - **<login_id>**—The XSLT must produce this mandatory node. The value of this node is a username of the user to be matched against the username in the BlackBerry AtHoc system.
    - **<lineage>**—(Optional) This node represents a membership to an organizational hierarchy. This node must be present when assigning a user to an organization hierarchy. The value of this node is the path of the organization hierarchy.
    - **<displayName>**—(Optional) When provided, the value is the display name for a user.
    - **<firstName>**—(Optional) When provided, the value is the first name of the user.
    - **<lastName>**—(Optional) When provided, the value is the last name of the user.
    - **<enabled>**—(Optional) Values are “Y” or “N”. “Y” means the user should be created as an enabled user. “N” means the user should be created as a disabled user.
    - **<customFields>**—(Optional) The sub nodes of this node represent each attribute to synchronize. The sub node should be in the following format: `<field commonName=”common name of attribute”>Value of attribute</field>`

    - **<devices>**—(Optional) The sub nodes of this node represent each device to synchronize. The sub node should be in the following format:`<device commonName="common name of device">value of device</device>`
- **<staticlists>**—This node contains definitions for static distribution lists. It has the following sub nodes:
  - **<list>**—This node represents one static list. The commonName attribute of this node is the common name of the static distribution list in the BlackBerry AtHoc system. The <list> node has following sub nodes:
    - **<name>**—Name of static distribution list.
    - **<description>**—Description of static distribution list.
    - **<members>**—The node that represents members. Members can be users or other distribution lists.

CSV data processor

The CSV data processor configuration takes an XSL file to transform the output from the CSV data adapter to output expected by the data synchronizer. Example outputs of the data adapter, sample XSL file, and sample output from the CSV data processor are provided in the sample folder. You must write an XSL file which transforms the output from the LDAP data adapter to XML that can be consumed by the data processor.

The following example shows the structure of a sample output that must be generated by the data processor XSL transformation.

Sample CSV data processor output

```
<IntegrationData from="XsltDataProcessor">
  <users>
    <user seqId="1">
      <login_id>jdoe</login_id>
      <mapping_id>jdoe</mapping_id>
      <firstName>John,Firstname</firstName>
      <lastName>Doe'Lastname</lastName>
      <lineage>/India/</lineage>
      <Date>01/11/2019</Date>
      <DateTime>01/11/2018 11:11:59</DateTime>
      <customFields>
       <field commonName=”CheckboxAttr”>Yes</field>
      </customFields>
      <devices>
        <device commonName="WORKEMAIL"></device>
      </devices>
      <lists>
        <list commonName="test">Yes</list>
      </lists>
    </user>
    <user seqId="2">
      <login_id> </login_id>
      <mapping_id> </mapping_id>
      <firstName>John,Firstname</firstName>
      <lastName>Doe'Lastname</lastName>
      <lineage>/India/</lineage>
      <Date>01/11/2019</Date>
      <DateTime>01/11/2018 11:11:59</DateTime>
      <customFields>
       <field commonName=”CheckboxAttr”>Yes</field>
      </customFields>
      <devices>
        <device commonName="WORKEMAIL"></device>
      </devices>
      <lists>
        <list commonName="test">Yes</list>
      </lists>
    </user>
  </users>
</IntegrationData>
```

<IntegrationData>—This is a root node of the output file. It has the following sub nodes:

- **<users>**—This node contains definition for user. Under this node is a <user> node which represents one user that needs to be synchronized.
  - **<user>**—This node represents each node that will be synchronized. The following sub nodes can be present under the <user> node:
    - **<login_id>**—The XSLT must produce this mandatory node. The value of this node is a username of a user that will be matched against a username in the BlackBerry AtHoc system.
    - **<lineage>**—(Optional) This node represents a membership to an organizational hierarchy. This node must be present when assigning a user to an organization hierarchy. The value of this node is the path of the organization hierarchy.
    - **<displayName>**—(Optional) When provided, the value is the display name of the user.
    - **<firstName>**—(Optional) When provided, the value is the first name of the user.
    - **<lastName>**—(Optional) When provided, the value is the last name of the user.
    - **<enabled>**—(Optional) Values are “Y” or “N”. “Y” means the user should be created as an enabled user. “N” means the user should be created as a disabled user.
    - **<customFields>**—(Optional) The sub nodes of this node represent each attribute to synchronize. The sub node should be in the following format: `<field commonName=”common name of attribute”>Value of attribute</field>`
    - **<devices>**—(Optional) The sub nodes of this node represent each device to synchronize. The sub node should be in the following format” `<device commonName="common name of device">value of device</device>`
    - **<lists>**—This node can contain one or more <list> sub nodes that represents a distribution list.
      - **<list>**—Represents one list. CommonName is the commonName of a static distribution list in the BlackBerry AtHoc system. The value is either “Yes” or “No”. “Yes” means the user should be added as a member of this list. “No” means the user should be removed.

To learn more about XSLT, see the following URL:

#http://www.topxml.com/xsl/tutorials/intro/default.asp

Tip: You can use XSLT to create customized lineage from LDAP attributes. In addition to mapping LDAP attributes to BlackBerry AtHoc custom fields, you can translate LDAP attribute values into other values recognized by the BlackBerry AtHoc software.

Data synchronizer configuration

The User Sync Client uses web API, which is protected using the OAuth2 protocol. Before making configuration changes to the sdkParameters section, you must provision the BlackBerry Web API client as described in the Provision the BlackBerry AtHoc API client section. After provisioning the API client, the sdkParameters section looks like the example in the sdkParameters section.

sdkParameters section

This section describes parameters that are required to interface with the BlackBerry AtHoc API. Although this section does not use the SDK, it is named sdkParameters for legacy reasons.

```
<sdkParameters>
   <orgCode><!CDATA[orgCode]]></orgCode>
   <url>https://AtHocServer</url>
   <username>username</username>
   <password>password</password>
   <syncSource>AD</syncSource>
   <clientId>you get this after provisioning</clientId>
   <clientSecret isEncrypted=”false”>You get this after provisioning</
clientSecret>
  <!-- Configuration for client certificate: Optional-->
   <clientCertificate>
       <!-- The subject of the certificate: Optional-->
       <subject></subject>
       <!-- The store name where the cert resides, either:
 ROOT,TRUSTEDPEOPLE,TRUSTEDPUBLISHER,AUTHROOT,CERTIFICATEAUTHORITY,MY-->
       <storeName></storeName>
       <!-- The cert store location, either: LOCALMACHINE or CURRENTUSER-->
       <storeLocation></storeLocation>
   </clientCertificate>
</sdkParameters>
```

Field descriptions

- **<orgCode>**—The organization code that you set up in the BlackBerry AtHoc management system in the General Settings.
- **<url>**—The URL of the BlackBerry AtHoc server.
- **<username>**—The BlackBerry AtHoc username used to send web API requests.
- **<isEncrypted>**—If the password is encrypted, its value is "true", otherwise it is "false". The default is “false”.
- **<password>**—The BlackBerry AtHoc password used to send web API requests.
- **<syncSource>**—Required for static list synchronization. The synchronization source must be predefined in the BlackBerry AtHoc system.
- **<clientId>**—Unique identifier for your LDAPSync client after the provisioning step.
- **<clientSecret>**—OAuth2 client secret that is returned after you provision the LDAPSync client. Do not share this secret.
- **<clientCertificate>**—(Optional) This node is required if you want to append certificates in data sync requests. This node has following sub nodes:
  - **<subject>**—(Optional) Subject of the specific certificate that you want to use.
  - **<storeName>**—(Optional) The name of the certificate store.

Behavior

- If the <isEncrypted> attribute is missing, the application assumes that the provided password is in clear text. The application encrypts the password and adds the "isEncrypted" attribute with the value "true".
- If the <isEncrypted> attribute is set to “true”, the application assumes that the password is already encrypted and does nothing.

- If the <isEncrypted> attribute is set to “false”, the application assumes that the password is in clear text and encrypts it. The application encrypts the password and then sets the <isEncrypted> attribute to "true" and replaces clear text with cipher text.
- If the <clientCertificate> block is missing, the application does not append any certificate to the data synchronization request.
- If the <subject> block is missing or empty, the application appends all certificates to the data synchronization request.
- If the <storeName> block is missing or empty, the application uses the default store ("CERTIFICATEAUTHORITY") to look for certificates.
- If the <storeLocation> block is missing or empty, the application uses the default location ("LOCALMACHINE") to look for certificates.

Sync operation configuration

This section configures the data synchronizer.

```
<syncOperations>
  <hierarchy><commonNameOfHierarchyToSync>
   Userbase_Hierarchy_Common_Name</commonNameOfHierarchyToSync>
  </hierarchy>
  <user>
<syncExistingUserOnly>true</syncExistingUserOnly>
<usersPerPackage>5000</usersPerPackage> <!—set it to 5000->
  </user>
  <staticlist>
<newListDefaultFolder>/folder_of_distribution_list_hierarchy/</
newListDefaultFolder>
  </staticlist>
<allowPartialUpdate>TRUE</allowPartialUpdate>
</syncOperations>
```

In this section, there are two possible subsections, <hierarchy> and <user>, that provide control information for hierarchy synchronization and user synchronization operations.

- **<hierarchy>**—The **<hierarchy>** section has only one sub node, <commonNameOfHierarchyToSync>. This sub-node defines the common name of the target hierarchy in the current provider. This common name must be specified in the BlackBerry AtHoc system before synchronization.

  To find the common name, go to **Settings** > **User Attributes** and find the Path type attribute in the list. The default attribute is called "Organization Hierarchy" and has a common name of "ORGANIZATION HIERARCHY".
- **<user>**—The **<user>** section has the following two sub nodes:
  - **<usersPerPackage>**—Because the number of users to be synchronized can be very large, the synchronization module synchronizes users into multiple packages of a web API request. This parameter controls the number of users placed in one package, which depends on the API server’s capacity. If this tag is missing, the number of users defaults to 5000.
  - **<syncExistingUserOnly>**—Indicates if the Web API should synchronize existing BlackBerry AtHoc users only. If this tag is missing, it defaults to “false”, meaning that all users are synchronized.
- **<staticList>**—The **<staticList>** section has the following sub node:
  - **<newListDefaultFolder>**—(Optional) Specifies the hierarchy folder to insert newly created lists into. Defaults to “root”. This is not applicable for CSV.
- **<allowPartialUpdate>**—(Optional) Indicates if the user sync can be performed for partial updates.

Tip: If you do not want to synchronize the hierarchy, remove the <hierarchy> section.

Tip: To test the configuration, execute only one or two of the three functional parts of the integrator. Remove the configuration section of the unwanted part.

Synchronize LDAP groups

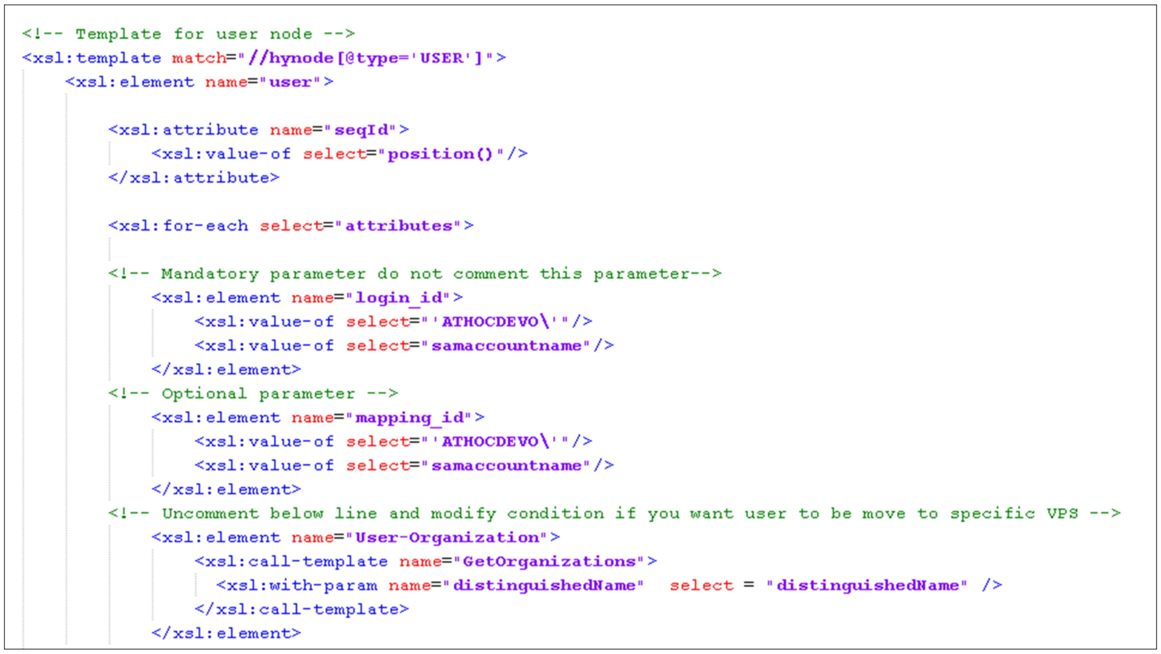
To synchronize LDAP groups, complete the following tasks:

1. Include the group objectClass in the LDAP search filter. Refer to the sample configuration file for details. To synchronize only Microsoft Exchange-enabled groups (Outlook groups), add a special filter which can be found in the sample configuration file.
2. Define the class configuration in the classConfig section. A sample configuration can be found in the sample configuration file.
3. Modify the XSLT to transform group data into the group sync data format. See the `Sample.xsl` sample XSLT file for details.
4. Specify the synchronization source under the sdkParameters node. Include the staticList synchronization section and specify the appropriate values.

How to synchronize users for enterprise user moves

To support enterprise user move with the User Sync Client, a new "User-Organization" attribute was added. You must enter the name of the organization that you want to add the user to in the "User-Organization" attribute. The enterprise user move feature is available on enterprise organizations with sub organizations. You can define how you want to do the mapping. You can write a template in an XSL which can have different xsl: if, or xsl:choose conditions. Before using the Enterprise User Move feature, the enforce uniqueness feature must be turned on for the enterprise in the BlackBerry AtHoc Management System.

The following is a sample XSL file:



```
<!-- Template for user node -->
<xsl:template match="//hynode[@type='USER']">
    <xsl:element name="user">

        <xsl:attribute name="seqId">
            <xsl:value-of select="position()"/>
        </xsl:attribute>

        <xsl:for-each select="attributes">

        <!-- Mandatory parameter do not comment this parameter-->
            <xsl:element name="login_id">
                <xsl:value-of select="'ATHOCDEVO\'"/>
                <xsl:value-of select="samaccountname"/>
            </xsl:element>
        <!-- Optional parameter -->
            <xsl:element name="mapping_id">
                <xsl:value-of select="'ATHOCDEVO\'"/>
                <xsl:value-of select="samaccountname"/>
            </xsl:element>
        <!-- Uncomment below line and modify condition if you want user to be move to specific VPS -->
            <xsl:element name="User-Organization">
                <xsl:call-template name="GetOrganizations">
                  <xsl:with-param name="distinguishedName"  select = "distinguishedName" />
                </xsl:call-template>
            </xsl:element>
```

Appendix A: Adtools.exe

`AdTools.exe` is a utility Windows application used to create, delete, and display LDAP hierarchies for the purposes of analyzing and testing the LDAP structure. It provides the functionalities described in the following sections through its main menu:

Set LDAP Info

This menu command opens a window that enables users to modify LDAP information including the server, username, password, and the distinguished name of the root node which is used for data integration into BlackBerry AtHoc.

This information is automatically stored in the XML file, `AdInfo.xml`, which can be found in the same folder as the AdTools.exe file. All values in this XML file are the default values the next time that the AdTools application starts.

Display LDAP Entry Details

This menu command displays all attributes and values for all LDAP entries in the LDAP tree whose root entry is specified in the LDAP information file (`AdInfo.xml`).

Some attributes of the LDAP entries may have multiple values, such as “objectclass” and “memberof”. These values display in sequential order.

Display Class Type and Count

This menu command displays the class hierarchies and counts of their object. This is useful when you need to identify the most specific class for an LDAP entry.

Test Regular Expression

Use this command to test the regular expression pattern to be used in the transform configuration. To test the pattern, input a pattern and string to match.

Appendix B: Migrate from LDAP Sync Client version 1.2.7 to BlackBerry AtHoc User Sync Client

This appendix explains how to migrate the LDAPSync Module configuration files from LDAP Sync Client version 1.2.7 to the BlackBerry AtHoc User Sync Client version 1.0.0.

Prerequisites

Before you migrate your LDAP Client Sync version, complete the following tasks

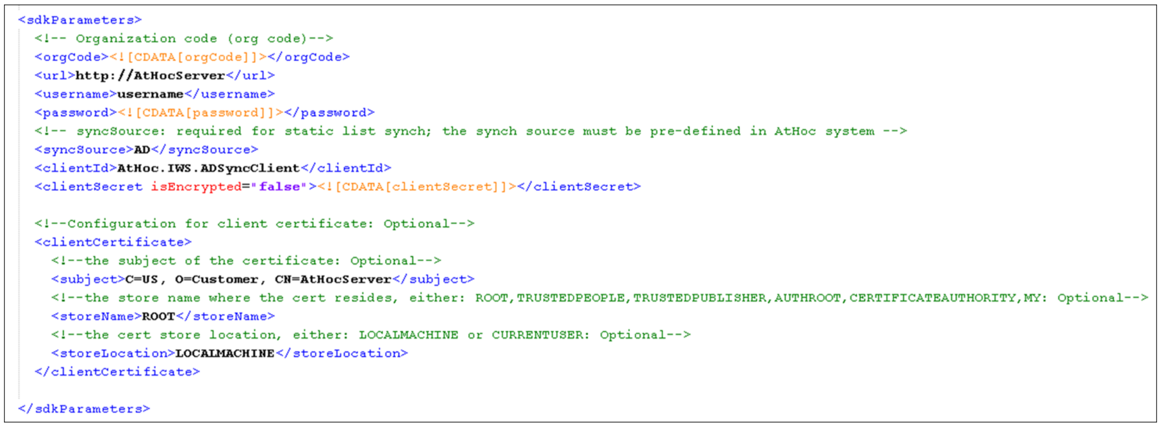
- Provision the BlackBerry AtHoc API client
- Set up an organization code in the BlackBerry AtHoc system

Set authentication parameters

To set authentication parameters in the in AtHocDataIntegration configuration file, make the following changes under the **sdkParameters** node:

1. Remove the **providerId** node.
2. Add **orgCode** and set it to the organization code that you created in Set up an organization code in the BlackBerry AtHoc system.
3. Change the **Url** from https://AtHocServer/sdk/listener/listen.asp to https://AtHocServer.
4. Add the **clientId** and **clientSecret** that you noted in the Provision the BlackBerry AtHoc API client section.
5. Provide the **username** and **password** of the user who has the SDK role.

Example



```
<sdkParameters>
  <!-- Organization code (org code)-->
  <orgCode><![CDATA[orgCode]]></orgCode>
  <url>http://AtHocServer</url>
  <username>username</username>
  <password><![CDATA[password]]></password>
  <!-- syncSource: required for static list synch; the synch source must be pre-defined in AtHoc system -->
  <syncSource>AD</syncSource>
  <clientId>AtHoc.IWS.ADSyncClient</clientId>
  <clientSecret isEncrypted="false"><![CDATA[clientSecret]]></clientSecret>

  <!--Configuration for client certificate: Optional-->
  <clientCertificate>
    <!--the subject of the certificate: Optional-->
    <subject>C=US, O=Customer, CN=AtHocServer</subject>
    <!--the store name where the cert resides, either: ROOT,TRUSTEDPEOPLE,TRUSTEDPUBLISHER,AUTHROOT,CERTIFICATEAUTHORITY,MY: Optional-->
    <storeName>ROOT</storeName>
    <!--the cert store location, either: LOCALMACHINE or CURRENTUSER: Optional-->
    <storeLocation>LOCALMACHINE</storeLocation>
  </clientCertificate>

</sdkParameters>
```

Remove deprecated configuration

Under syncOperations/staticlist, remove the operator node. This node is no longer supported and must be removed.

Migrate from mid to login_id

Terms

- mid
  - The mid was the primary key when synchronizing in previous versions.
  - Display name in BlackBerry AtHoc: Mapping ID
  - Common Name in IWS: mapping_id
- login_id
  - The login_id is the Common Name for Username in BlackBerry AtHoc. It is the primary key when synchronizing against BlackBerry AtHoc with the User Sync Client
  - Display name in BlackBerry AtHoc: Username
  - Common Name in BlackBerry AtHoc: login_id

What is changing?

In LDAP Sync Client 1.2.7 and lower, all user synchronization happened through older SDK code. The SDK payload accepted mid in the input payload field to look up and synchronize users. In the User Sync Client, the API requires login_id as the primary key.

Behavior prior to User Sync Client:

- For the user payload, if a user did not exist with the provided mid, it would create a new user and set the mapping_id and the login_id to mid if the login_id was not separately provided in the payload. If mid was present in the payload, and if the user existed with the same mapping_id, it would then update that record for the user.
- For the distribution list synchronization payload, all mappings were managed by looking up members by their mid.

Behavior in User Sync Client:

- All user lookup and synchronization happens using the login_id (Username).
- All distribution list-to-user mappings are based on login_id.
- The “mid” field is no longer supported.

Are you affected?

1. Do you set a value for login_id in the template?
   a. Search for references to login_id.
   b. If you find any references, you have explicitly set a value for login_id.
   c. You need to use this LDAP attribute as the primary key. Whatever you previously set to mid can be set to mapping_id.
   d. Ensure that any LDAP attributes that you are using for login_id have memberExport=“true” in the AtHocDataIntegration.config file. Your configuration may have the setting for this attribute that was used for mid. Not making this update in the AtHocDataIntegration.config file will lead to errors during distribution list member synchronization.
2. Does BlackBerry AtHoc have different values?
   a. Export all users to a CSV file. Include only the mid and username columns.
   b. Open the CSV file in Microsoft Excel.
   c. Add this formula to cell C1 and then fill down: **=IF(A1<>B1,"No match","")**
   d. Filter to show only users who have No Match.
   e. Filter to show only users who have a mapping_id. This is not required when creating users in the UI.
   f. Any users left may be impacted if synced from the User Sync Client.

If you answered “No” to 1 and 2 above, BlackBerry AtHoc automatically copies the mid to login_id. Switching them will not have an impact.

If you answered “Yes” to either 1 or two above, you will likely need to change the usernames of existing users. Contact BlackBerry AtHoc technical support at athocsupport@blackberry.com for assistance.

How to migrate

Complete the following tasks to migrate from mid to login_id.

Change the AtHocDataConfiguration.config file

The attribute that you use for login_id should have memberExport=“true” in the `AtHocDataConfiguration.config` file.

In the following example, “samaaccountname” is used as the mid prior to 2.0. In 2.0, “userPrincipalName” is used as the login_id, and the memberExport is set to true.

AtHocDataConfiguration.config in LDAP Sync version 1.2.7

```
<class objectClass="user" type="USER" nameAttribute="name">
            <!-- Attributes to be fetched from LDAP for user -->
            <attributes>
              <!-- memberExport is optional; true indicates this attribute will be
 exported for static list membership sync too -->
              <attribute memberExport="true">samaaccountname</attribute>
          <attribute>sn</attribute>
              <attribute>userPrincipalName</attribute>
              <attribute>displayname</attribute>
              <attribute>userAccountControl</attribute>
              <attribute>mail</attribute>
              <attribute>telephoneNumber</attribute>
              <attribute>distinguishedName</attribute>
              <!-- Sample definition to get multi-values of given LDAP attribute
 -->
              <!--<attribute multiValued="true">objectclass</attribute>-->
            </attributes>
          </class>
```

AtHocDataConfiguration.config after User Sync Client

```
<class objectClass="user" type="USER" nameAttribute="name">
            <!-- Attributes to be fetched from LDAP for user -->
            <attributes>
              <!-- memberExport is optional; true indicates this attribute will be
 exported for static list membership sync too -->
              <attribute>samaccountname</attribute>
              <attribute>sn</attribute>
             <attribute memberExport="true">userPrincipalName</attribute>
              <attribute>displayname</attribute>
              <attribute>userAccountControl</attribute>
              <attribute>mail</attribute>
              <attribute>telephoneNumber</attribute>
              <attribute>distinguishedName</attribute>
              <!-- Sample definition to get multi-values of given LDAP attribute
 -->
```

```
              <!--<attribute multiValued="true">objectclass</attribute>-->
            </attributes>
          </class>
```

Change the XSL template file

You must update the existing XSL template file to use login_id as the primary look up key for both the payload and distribution list to member mapping payload as shown in the following examples.

User element in LDAP Sync version 1.2.7

```
<xsl:template match="//hynode[@type='USER']">
        <xsl:element name="user">
            <xsl:attribute name="seqId">
                <xsl:value-of select="position()"/>
            </xsl:attribute>
            <xsl:for-each select="attributes">
                <xsl:element name="mid">
                    <xsl:value-of select="samaaccountname"/>
                </xsl:element>
                <xsl:element name="displayName">
                    <xsl:value-of select="displayname"/>
                </xsl:element>
                <xsl:element name="firstName">
                    <xsl:value-of select="givenname"/>
                </xsl:element>
                <xsl:element name="lastName">
                    <xsl:value-of select="sn"/>
                </xsl:element>
```

User element after User Sync Client

```

<xsl:template match="//hynode[@type='USER']">
        <xsl:element name="user">
            <xsl:attribute name="seqId">
                <xsl:value-of select="position()"/>
            </xsl:attribute>
            <xsl:for-each select="attributes">
                <xsl:element name="login_id">
                    <xsl:value-of select="samaaccountname"/>
                </xsl:element>
                <xsl:element name="displayName">
                    <xsl:value-of select="displayname"/>
                </xsl:element>
                <xsl:element name="firstName">
                    <xsl:value-of select="givenname"/>
                </xsl:element>
                <xsl:element name="lastName">
                    <xsl:value-of select="sn"/>
                </xsl:element>
```

Distribution list mapping in LDAP Sync v 1.2.7

```
<xsl:element name="members">
    <xsl:for-each select="members/member[@type='USER']">
```

```
        <xsl:element name="user">
            <xsl:attribute name="mid">
                <xsl:value-of select="samaccountname"/>
            </xsl:attribute>
        </xsl:element>
    </xsl:for-each>
```

Distribution list mapping after User Sync Client

```
<xsl:element name="members">
    <xsl:for-each select="members/member[@type='USER']">
        <xsl:element name="user">
            <xsl:attribute name="login_id">
                <xsl:value-of select="samaccountname"/>
            </xsl:attribute>
        </xsl:element>
    </xsl:for-each>
```

Considerations for using User Sync Client to perform LDAP synchronization for existing customers

1. For the User Sync Client to work properly with existing customers, their mapping_id and login_id (Username) must be identical. Or, the combination of mapping_id and login_id in BlackBerry AtHoc needs to be the same as a combination of mapping_id and login_id fields from LDAP as a prerequisite.
2. Change the XSL files as outlined above.
3. Make sure that memberExport is set to true in the AtHocDataConfiguration.config file for the LDAP attribute that you are using to map to login_id.
4. Discourage operators from changing the login_id (Username) for users. As a best practice, set the username to not be editable by end-users in the BlackBerry AtHoc management system under System Setup.

Appendix C: Differences between the CSV Importer Tool and the BlackBerry AtHoc User Sync Client

The BlackBerry AtHoc User Sync Client is the supported way to synchronize CSV files, replacing the CSV Importer Tool. If you were using the CSV Importer Tool before, the following are the major changes that you must be aware of:

- The BlackBerry AtHoc User Sync Client uses the user's **username** as the **primary key** to synchronize users as opposed to the CSV Importer tool which used the **mapping_id**. Ensure that the input CSV file has the username (common name: login_id) present.
- The major functions of the CSV Importer Tool are preserved. However, the configuration file is completely different. See Configure the data integration file and sub sections for the CSV data adapter, CSV data processor, and data synchronizer.
- The Log file format and Debug steps are different. See Execute the BlackBerry AtHoc User Sync Client.

BlackBerry AtHoc customer portal

BlackBerry AtHoc customers can obtain more information about BlackBerry AtHoc products or get answers to questions about their BlackBerry AtHoc systems through the Customer Portal:

https://support.athoc.com/customer-support-portal.html

The BlackBerry AtHoc Customer Portal also provides support via computer-based training, Operator checklists, best practice resources, reference manuals, and users guides.

Legal notices

Contact Information

BlackBerry AtHoc

311 Fairchild Drive

Mountain View, CA 94043

Tel: 1-650-685-3000

Email: athocsupport@blackberry.com

Web: http://www.athoc.com