PEGA 7 PLATFORM

Business Intelligence Exchange (BIX) 7.3.1

User Guide

USA

Phone: 617-374-9600

Fax: (617) 374-9620

www.pega.com

DOCUMENT: Business Intelligence Exchange (BIX) 7.3.1 User Guide

SOFTWARE VERSION: 7.3.1

PUBLISHED: Monday, October 16, 2017

Contents

Business Intelligence Exchange (BIX)

Business Intelligence Exchange (BIX) is an optional add-on product consisting of a ruleset and a stand-alone Java program that can be run from a command line. BIX provides the extract functions of an ETL (extract, transform, and load) utility by using the *Pega-BIX* ruleset, which supports the *Rule-Admin-Extract* rule type. Data is extracted and exported in a format suitable for use in popular business intelligence applications, such as data warehouses. High-performance, multi-threaded extraction operations operate independently of the Pega Platform, and can occur even when the Pega Platform is not running.

You can also run BIX extracts manually in the Designer Studio. Data can be output as XML or Comma Separated Value (CSV) formats, or to a database.

BIX Version

Before using BIX, ensure that the BIX version you are using matches the installed Pega Platform version. Versions might not match after you update or upgrade the Pega Platform because updates and upgrades do not automatically update add-on products such as BIX. Refer to the *BIX User Guide* for instructions on upgrading BIX.

BIX Architecture

When BIX runs as a standalone Java program outside of the Pega Platform, the server that BIX runs on must be able to access the Pega Platform database(s) from which data will be extracted. The following diagram shows a typical deployment architecture for BIX.

An extract, also referred to as an extraction or extraction process, is a process that runs one or more Extract rules. If it runs more than one Extract rule, they are run serially.

Each Extract rule extracts data from a single class. All of the Extract rules for an extract must extract data from the same database.

An extract can be run:

- within the Pega Platform using an agent that calls the pxExtractDataWithArgs activity for one or more Extract rules.
- as a command-line process. An extract run as a command-line process can be run from any server that can access the source database and the destination database or file system.
- A single Extract rule can be run manually from the Extract rule form, however, this is typically only done in development environments with small volumes of data to test these rules.

BIX Installation and Upgrades

Platform requirements

When BIX runs as a separate process outside of the Pega Platform, there are no platform restrictions.

- When extracting to a database, the destination database must be supported by the Pega Platform version being run.
- A JVM is required that is running the same Java version as the one supported by the Pega Platform being used.
- If you are using ANT, it must be version 1.5 or higher. Version 1.7.1 or higher is recommended.
- The BIX version must be the same as the Pega Platform version.

Installing BIX

The procedure for installing BIX into an instance of the Pega Platform differs depending on which version you are running. Use the following steps to install BIX 7.2 with the Pega Platform.

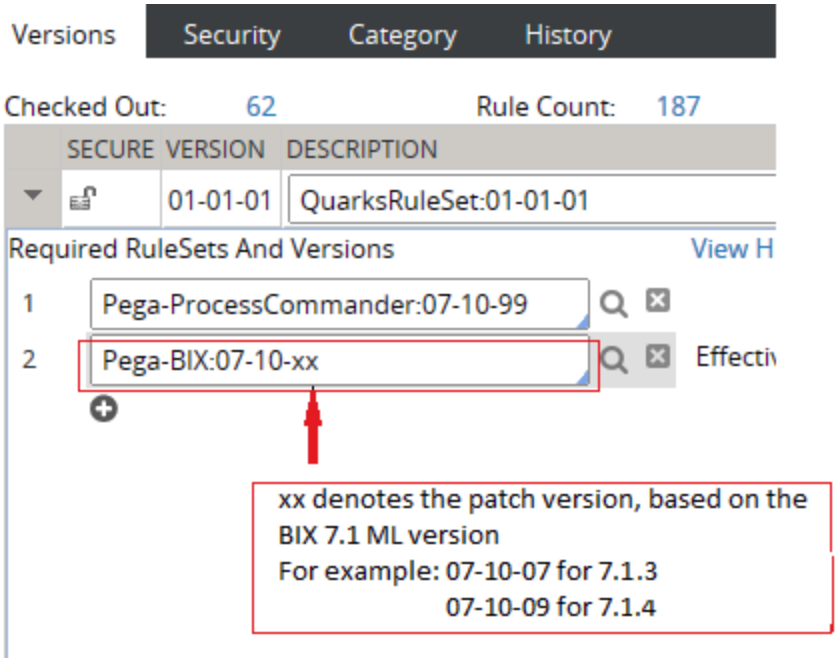
1. Unzip the BIXDistributionImage.zip file to a directory on your local computer.
2. Log in to your instance of the Pega Platform using an operator ID with Administrator privileges.
3. In Designer Studio, select **Designer Studio > Application > Distribution > Import**.
4. Click **Choose File**.
5. Browse to the directory that you populated in step 1, select the **Additional Products > BIX > Rulesets** directory.
6. Select the appropriate ZIP file and click **Open**. For BIX 7.2, the file is pxBIX_07.10.01.zip.
7. Click **Next**. The File information dialog box is displayed.
8. Click **Next**.
9. Select how to apply the schema changes. The options are:
    - Automatic
    - Manual
10. Click **Next**.
11. The import status screen is displayed. When the import has finished, click **Done**.
12. Locate your Application rule (select it from your Access Group display or from the list under **Application Definition > Application** in the Records Explorer). Add Pega-BIX: 07-10 to the list of Application rulesets.
13. Save the ruleset.

14. On your application rule, locate and open your base Application ruleset (or select it from the list under **SysAdmin > RuleSet** in the Records Explorer). Add Pega-BIX: 07-10 to the list of required rulesets.
15. Save your changes and log out. When you log in again, BIX displays as the Extract selection under **SysAdmin** in the Rules Explorer.

Upgrading BIX

Upgrading BIX requires installing one ruleset. The new version of BIX can be installed on top of your existing version.

1. Unzip the BIXDistributionImage.zip file to a directory on your local computer.
2. Log in to your instance of the Pega Platform using an operator ID with Administrator privileges.
3. In Designer Studio, select **Designer Studio > Application > Distribution > Import**.
4. Browse to the directory that you populated in step 1, select the **Additional Products > BIX > Rulesets** directory, and then select the appropriate ZIP file. For BIX 7.1, the file is pxBIX_07.10.01.zip.
5. Click **Upload file**.
6. When the .zip file has been uploaded to the server, the screen displays **Import Options**. Select the check boxes to:
    - Compile Libraries
    - Update Only If Newer
    - Overwrite Existing Rules
7. Click **Import**.
8. When the import process is complete, add the BIX ruleset to your base ruleset version in the Application rulesets list as Pega-BIX:07-10-xx, where "xx" is the patch version that matches the version of Pega Platform that you are running.



9. Save the ruleset.
10. Locate your Application rule (select it from your Access Group display or from the list under **Application Definition > Application** in the Records Explorer). Add Pega-BIX: 07-10-01 to the list of Application rulesets.
11. Save your changes and log out. When you log in again, BIX displays as the Extract selection under **SysAdmin** in the Rules Explorer.

Post Upgrade

After upgrading BIX, you may need to:

- Reconfigure your stand-along command-line environment:
  - Download the BIX distribution archive and expand it into a directory.
  - Configure a JVM environment with access to the Pega Platform jar files and the appropriate database driver files
  - Configure the prconfig.xml, prbootstrap.properties, and prlog4j2.xml files provided in the configuration directory of the BIX distribution to specify the source and target databases and split schema setup. See Configuring the extraction environment.
- Perform additional steps if you were using the Extract rule xml as input for extracting the rules:
  - Regenerate the pegarules.keyring file, if required.
  - Copy the Extract rule xml to the appropriate path.
  - Validate the ANT build file with the sample build file provided in the distribution package.
- Update custom work tables if they do not have the pxCommitDateTime and pxSaveDateTime columns and are using pxUpdateDateTime for incremental extracts. Incremental extracts are done

using pxCommitDateTime. See BIX Performance for additional information.

- Update your work tables to include the new columns.
- Populate the new columns by running the resave utility.

Apply DDL scripts manually to create the pr_extract_time and pr_sys_queues_bix tables if the are not present in the PegaRULES schema.

About Extract rules

Extracts are specified at the class level, not at the application level. For example, if you have three case types in your application and would like to extract data for each, you must define an extract rule for each of the three case types.

If you are extracting from a class group, the extract will include all classes in the class group. If the class is not a class group, the extract will only include instances of that class.

You can specify the properties in each class of your application that you want to extract. You can extract all properties, including Value Lists, Value Groups, and multiple levels of embedded Pages, Page Lists, and Page Groups. The extract for a class can include any or all properties inherited by that class from other classes. You can also filter the data to be extracted, extracting only the data that matches the conditions you set.

The data contained in these properties can be extracted to an XML file, to a Comma Separated Values (CSV) file, or to one or more tables in a relational database.

A single extract process cannot concurrently extract from multiple databases or different database instances on the same machine. Each extract process can have only a single data source. A single extract process may execute multiple Extract rules within the same Pega Platform database instance. If it does so, the rules are executed serially. A single extract process can, however, extract to different target or destination databases or files. You can, however, run different extract processes, extracting from multiple Pega Platform databases, concurrently.

Extraction frequency

BIX does not extract data in real-time as the Pega Platform database is updated. Because it runs as a stand-alone Java program outside of the Pega Platform or using a Pega Platform agent, it can be scheduled to run as frequently as desired.

Schema

The database schema can be generated based on your specifications in the extract definition. You can specify the table and column names and define the data type and precision. BIX provides and option that automatically generates a Data Definition Language (DDL) file for the database schema it expect to write data to during an extraction. This file can be used by a Database Analyst (DBA) to create the appropriate tables in the destination database. Any updates to the extract definition that result in the addition of columns or tables need to be added manually by a DBA.

When generating to XML, the XML element names match the Pega Platform names and cannot be modified.

Extracting data from BIX

To extract data from within Pega Platform:

1. Create a new Extract rule on the **New** tab.
2. Specify the properties to be extracted on the **Definition** tab.
3. Establish any filters for the data on the **Filter Criteria** tab.
4. Specify the destination for the Extract rule on the **File Specification** tab.
5. Run the Extract rule. For details on this step, see Running the command-line extract process, Running an Extract rule manually in Designer Studio, and Running an Extract rule using a Pega Platform agent.
6. Run the Extract rule. For details on this step, see Running an Extract rule manually in Designer Studio, and Running an Extract rule using a Pega Platform agent.

The output data includes a time stamp and a batch identifier. These help when you want to compare information from different extracts or isolate information from a specific extract.

– Completing the Create, Save As, or Specialization form

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing rules by creating a copy in a specific ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Based on your use case, you use the Create, Save As, or Specialization form to create the record. The number of fields and available options varies by record type. Start by familiarizing yourself with the generic layout of these forms and their common fields using the following Developer Help topics using the following Developer Help topics:

- Creating a rule or data instance
- Copying a rule or data instance
- Creating a specialized or circumstance rule
- Creating a rule or data instance
- Copying a rule or data instance
- Creating a specialized or circumstance rule
- Creating a rule or data instance
- Copying a rule or data instance
- Creating a specialized or circumstance rule

This information identifies the key parts and options that apply to the record type that you are creating.

Create an Extract rule by selecting **Create > SysAdmin > Extract**.

Key parts

Extract rules have two key parts.

Field	Description
Applies to	Enter the name of the concrete class that corresponds to the properties being extracted.
Purpose	Enter a name for this Extract rule. Begin the name with a letter, and use only letters, numbers, and hyphens.

In addition to these fields, enter a short description of the Extract rule, the class name of the properties that you are extracting, the name of the Extract rule (in the **Purpose** field), and the appropriate ruleset and version. Click **Create** to create the Extract rule.

Rule resolution

When searching for rules of this type, the system:

- Filters candidate rules based on a requestor's ruleset list of rulesets and versions
- Searches through ancestor classes in the class hierarchy for candidates when no matching rule is found in the starting class

Time-qualified and circumstance-qualified rule resolution features are not available for this rule type.

– Completing the Definition tab

Use the **Definition** tab to specify properties to extract from any class of an application. You can extract values for properties of many modes, including Single Value, Value List, Value Group; and multiple levels of embedded Pages, Page Lists, and Page Groups. BIX cannot export properties of the Java Object or Java Property modes.

BIX can extract values to an XML file or a Comma Separated Values (CSV) file, or directly into a relational database as database schema.

The output data includes a time stamp and a batch identifier. This information helps when comparing different extracts, or when isolating information from a specific extract.

Selecting an output format

Select an output format for the extract:

- **XML** — Supports two levels of extraction:
  - Select the **Get All Properties** check box to extract all properties of a specified class into XML documents.
  - Clear the **Get All Properties** check box then select the properties to extract in the tree list of properties that displays.

- **Comma Separated Values (CSV)** — Export instances of classes as comma-separated values in a text file.
    - Typically produces multiple CSV files, depending on the complexity of properties selected.
    - You must explicitly select the set of properties to export from the tree display.
    - CSV files generated by BIX conform to the RFC 4180 standard, which results in field values containing line breaks (CRLF), double quotes, and commas being enclosed in double quotes.
- **Database Schema** — Export data from the PegaRULES database directly into another relational database.
    - Identify this database on the File Specification tab using a Database Name data instance (*Data-Admin-DB-Name* class).
    - The table name that you specify must comply with the naming conventions of the target database and cannot include spaces.

  This option is not available when running an Extract rule in the Pega Cloud.

Working with the property tree

The Extract rule separates available properties into nodes that can expand and collapse. The root node is the Applies To class of the Extract rule.

Selecting properties to extract

When using XML output, to extract all the properties in the class, select the **Get All Properties** check box.

For other output formats, click a node to display a gear icon to the right of the node name. Click the gear icon to display the **Select Properties** form, and then select the properties that you want to extract.

Filtering the properties list

To filter the properties list, enter a text string in the **Property Name Contains** field and click **Filter**.

To show all properties again, clear the field and click **Filter** again.

Configuring properties

Click any property to display a gear icon and a trash-can icon to the right of the property. Click the gear icon to display a **Property Configurations** form. In this form:

- When using CSV output, you can map the property to a name in the CSV file.
- When exporting for import into a database, set the attributes for the property: map to a name in the database schema, provide the data type and the field length. The name needs to comply with the naming conventions of the target database and cannot include any spaces.

Embedded properties

How embedded properties are handled depends on which output format is used:

- When extracting to an XML file, all data is extracted to a single file. In the XML file, embedded properties are represented as embedded tags within the XML.
- When extracting to a Comma Separated Value (CSV) file, BIX automatically normalizes the data, creating a file to hold the top-level properties of a class, and separate files (with appropriate foreign keys pointing to the entries in the main file) to hold data for each embedded page list or page group.
- When extracting to a database, BIX automatically normalizes the data, storing the top-level properties of a class in one table and data embedded page lists and page groups in separate tables.

– Completing the Filter Criteria tab

Complete the **Filter Criteria** tab to extract values:

- That have changed since the last time the Extract rule was run
- That satisfy any filter conditions you specify involving properties of the rules's primary class

The following table details how to use the fields listed in the Criteria array.

Field	Description
Use Last Updated Time as Start	For classes that include the property *pxCommitDateTime*, select this check box to extract all data that was created or updated since the last time the Extract rule was run. This field is not relevant until the Extract rule has run at least once. If you select the incremental extraction option, a filter condition is automatically included during an extract that restricts extraction to class instances where the *pxCommitDateTime* property value is greater than or equal to the last date and time when the extract ran. *pxCommitDateTime* is automatically set for each class instance when it is changed in the database by the Pega Platform database regardless of how the data was created or changed (interactively by an end user, programmatically by an activity, through an import, or by any other means). This condition is added to any others you specify on this tab. Sometimes the *pxCommitDateTime* property may need to be added as a database column to the class table. For example, it is not added automatically on upgrades. An extract using the incremental extraction property will fail if this column does not exist in the class table. Also note that when incremental extraction is performed, class instances with NULL values of the *pxCommitDateTime* property will be skipped and when using the -c command line option to extract from child classes, those child classes whose class tables do not include the *pxCommitDateTime* property will be skipped. Do not select this check box if you plan to run the Extract rule using the command-line and intend to use the -d, -D, -u, and -U parameters. This is because these parameters do not override the check box setting , which can result in unanticipated results from a command-line extract.
Skip standard filters	Specifies whether to add the default filter condition on *pzInskey /pxObjClass* columns in the where clause of the source query during the extraction process. • When selected, the source query will not have the default filter condition on *pzInskey /pxObjClass* columns (even when the -c command line option is enabled) in the where clause during the extraction process. • When cleared, the source query will have the default filter condition on *pzInskey /pxObjClass* columns in the where clause during the extraction process. This is the default.
Skip standard order by clause	Specifies whether to add default sorting on the *pzInsKey* column to the source query during the extraction process. • When selected, then order by clause based on *pzInskey* column will not be added to the source query during the extraction process.

	• When cleared, then order by clause based on *pzInskey* column will be added to the source query during the extraction process. This is the default. When cleared and the -n command-line option is used, the order by clause based on the *pzInsKey* column is not added to the source query during the extraction process to provide backward compatibility. Prior to the introduction of this field, if the -n command-line option was specified, the order by clause based on *pzInsKey* column was not added to the source query during the extraction process.
Logic	Based on the **Label** values in the Criteria array, enter a logical expression that defines how the system combines the criteria into an overall Boolean value at runtime. The expression can include parentheses and the operators "AND" and "OR". For example, if the table contains four rows labeled A, B, C, and D, you can enter logic such as: (A OR B OR C) AND D. Notation for this field is similar to the logic statement in When condition rules. If you create a logic statement, it must include all labeled rows. If you do not include a logic statement, the system selects data instances for which all logic rows return the value True. You must include at least one logical expression in the Criteria array.
Label	Optional. An entry in this field is used in the **Logic** field. The entry must: • Be unique • Start with a letter • Have only alphanumeric characters • Have no spaces If you enter multiple criteria, enter a letter or letters to uniquely identify the row. For instance, if you have three rows, their labels can be A, B, and C.
Field	Enter a property reference to provide the filter values. You can only use properties exposed as individual columns. Start the property reference with a period. Click **Edit** to the right of the field to create a new property. You can specify single value properties that are produced by an SQL function. To use the result of an SQL function query, click the **Calculation Builder** icon to the right of the field. In the Calculation Builder, select an SQL function and specify the parameters. Do not use properties of type TextEncrypted. Unexposed property references, linked property references and non- scalar property references such as embedded properties, page lists, page groups, value lists, and value groups are not supported.
Operator	Select a comparison condition, such as **Is Equal** or **Starts With**. At runtime, BIX converts the operator comparison condition to a standard SQL comparison statement. Non-standard operators, such as the Oracle CONTAINS feature, are not available.
Value	Enter or select one of the following comparison values: • A literal constant, formatted in accordance with the property type. See Developer help topic Constants in expressions Developer help topic Constants in expressionsfor formatting help. Place a backslash character in front of any double-quote character that displays within the constant (\"). • A fully qualified property reference to a single value property that is present on the clipboard at run time. The type of this property must match the type of the property in the **Column** field. • A reference to a parameter that is defined in the report's rule, in the format param.name.

	• The name of another single value property that is exposed as a column. This allows you to compare the values of two properties in the same database row if their types are identical or allow comparison. • A calculation. To compare it with the result of a calculation, click the **Calculation Builder** icon to the right of the field. In the Calculation Builder, select an SQL function and specify the parameters. • A symbolic date, such as Yesterday or Current Year for a Date or DateTime value. When the list value rule executes, this symbolic reference is converted to an actual date or date range based on the time zone of the user and the condition value. For example, if you select Last Year and the condition value is IS EQUAL TO, the result is a date range in the previous calendar year between January 1 and December 31 inclusive. • For a DateTime property, you can specify the symbolic value **Last Extraction Time**. At runtime, this value is replaced with the date and time when the extract rule last ran. You can define a filter using this value as an alternative to checking the **Use Last Updated Time as Start** option described above, to perform incremental extraction based on any *DateTime* property you choose. If the **Condition** field is set to IS NULL or IS NOT NULL, leave the **Value** field blank. Unexposed property references, linked property references and non- scalar property references such as embedded properties, page lists, page groups, value lists, and value groups are not supported.
Use null if empty	This option is important only when there is no value in the **Value** field at run time. By default, when the **Value** field is empty at run time, the criteria defined by this row are ignored, and processing is the same as if this row is not present. For example, assume the criteria are that the customer's last name starts with a given letter, and the value set in the **Value** field is Param.Letter. If this extract runs at a time when Param.Letter has the value "C", the report will only contain instances in which the Lastname property value starts with the letter C. However, if the same extract rule runs again when the Param.Letter parameter has no value, this criteria is dropped, so the report will contain instances where the Lastname property value has any value or no value. When the default behavior is not desirable, select the **Use null if empty** check box to force the condition value to become IS NULL when the **Value** value is blank at run time. In the above example, the criteria is transformed to "Lastname is null" which is different than having no restriction on Lastname.
Ignore case	Select this option if you want run time comparisons of the **Field** value and the **Value** value to be case-insensitive. For example, "Smith" matches "SMITH" and also matches "sMiTH". If selected, at run time "a" matches "A" and is less than "B". When you do not select this option, comparison of the **Field** value and the **Value** value occurs without case conversion. In this case, "a" is greater than both "A" and "B". In most cases, do not select this option. Select this box only when necessary to obtain the rows of the report: • If your source database for the extract is hosted by Microsoft SQL Server, comparisons are always case-insensitive. Selecting this option does not affect the report contents, but might increase processing time. • If your source database for the extract is hosted by Oracle, IBM DB2 or most database vendor products, selecting this option might affect which rows appear in the report when values occur in mixed case. Conversion to uppercase can significantly slow down database processing. • Case conversion is meaningful only for properties of type Text, Identifier, or Password. Regardless of database software, case conversion is not needed for numbers, dates, or text that contains only uppercase or lowercase characters.

– Completing the File Specification tab

Complete the **File Specification** tab to control where extract output is written. The fields that display depend on the **Output Format** that you select in the **Definition** tab.

Use the -P command-line option to override the output directories specified on this tab, instead directing all files to a specified location.

Output Format	Fields

XML	If the Dynamic System Setting that you created to enable or disable the generation of an XSD file is set to false, the XML Schema director and XML schema File Name field are not available.
	• **XML schema directory** — Identify a directory to receive the XML schema file.
	• **XML schema File Name** — Enter a name for the XML schema file.
	• **XML file output directory** — Identify a directory to receive the XML output files.
	• **XML output file name** — Enter a name for the XML output files. The system automatically adds the.xml and .xsd extensions to the file name. You can use these wild card strings in the output file name to reference their corresponding substitution strings:
	  • %i - (RunIdentifier) The unique run identifier that is written in the extract file.
	  • The identifier can be passed in as a command-line parameter to the command-line extract process using the -k option.
	  • If no identifier is passed in, the system generates an identifier consisting of the *pzInsKey* of the Extract rule followed by a unique sequence number that increments every time that the extract runs.
	  • This identifier is unique across all nodes.
	  • %s - The automatically generated sequence number.
	  • %c - (ClassName) The Applies To class of the Extract rule.
	  • %r - The name of the Extract rule.
	  • %d - (ExtractRunDate) The date of the extract run.
	  • %t - (ExtractRunDateTime) The date and time of the extract run.
	These substitutions are also automatically applied to the BIX schema file for XML (XSD file), which is automatically generated for each run of an Extract rule using XML output for its data.

CSV	**CSV Output Directory** — Identify a directory to receive the CSV files.
Database	Select the name of the database (an instance of *Data-Admin-DB-Name*) to receive the extracted data. Because the Pega RULES database is the source of the data, do not select PegaRULES as the target in this field. Before extracting data to the target database, you must first create a schema to match the extracted data. You can use BIX to generate an SQL file that creates the appropriate schema. For more information, see Running a BIX extract from the command line. This options is not available when running an Extract rule in the Pega Cloud.

After you complete this tab, save the rule form, then click **Test Connectivity** to verify that you have a valid path to the database or directories specified in this tab. Pega Platform reports the results of the test in a pop-up window.

Create a manifest

Select the **Include manifest** check box to create a manifest, or summary statement, of the records extracted. You can create the manifest in the file format of your choice, independent of the output format that you selected for the actual extract.

Depending on your choice, additional fields display:

Output Format	Fields

XML	• **Manifest XML file output directory** — Identify a directory to receive the XML output files.
	• **Manifest XML output file name** — Enter a name for the XML output files. The system automatically adds the.xml and .xsd extensions to the file name. You can use these wild card strings in the output file name to reference their corresponding substitution strings:
	  • %i - (RunIdentifier) The unique run identifier that is written in the extract file.
	  • The identifier can be passed in as a command-line parameter to the command-line extract process using the -k option.
	  • If no identifier is passed in, the system generates an identifier consisting of the *pzInsKey* of the Extract rule followed by a unique sequence number that increments every time the extract runs.
	  • This identifier is unique across all nodes.
	  • %s - The automatically generated sequence number.
	  • %c - (ClassName) The Applies To class of the Extract rule.
	  • %r - The name of the Extract rule.
	  • %d - (ExtractRunDate) The date of the extract run.
	  • %t - (ExtractRunDateTime) The date and time of the extract run.
	These substitutions are also automatically applied to the BIX schema file for XML (XSD file) that is automatically generated for each run of an Extract rule that outputs data in XML output.

Comma Separated File (CSV)	**Manifest CSV file output directory** — Identify a directory to receive the CSV files.
Database Schema	**Manifest output database name** — Specify the name of the database where the manifest is to be written.

If you want to output the manifest to a database, you must prepare the database by adding two tables, **pr_extract_summary** and **pr_extract_details**. When you run the extract for the first time (or after editing the rule):

1. In the form where you enter run parameters, select the **Generate manifest schema DDL** check box to create the schema DDL that is used to create the tables.
2. In the next field, provide the location on your computer to create the schema.

3. Provide the created schema to your database administrator, who can then update the database so it can receive the manifest.
4. The manifest DDL scripts are generated with the default schema name prefixed to the table name.

Click **Test Connectivity** to verify that you have a valid path to the database or directory that you specified. Pega Platform reports the results of the test in a pop-up window.

In the **Include check total property** field, you can select an extract property to use as a "checksum" for the extract.

Contents of the manifest

The manifest that you generate includes a summary of the extract run, including:

- The unique identifier for the run, consisting of:
  - *pxExtractIdentifier*, which is the *pzInsKey* of the extract or a command-line parameter passed in using the -k option, with the sequence number appended.
  - *pxExtractDateTime*, which holds the date and time of the start of the run.
- The application that the extract is associated with (if it is associated with a single application).
- The number of class instances that the extract retrieved from the class table during the run.
- The DateTime for the start and end of the run and the elapsed time of the run in minutes, seconds, and milliseconds.
- The number of retrieved class instances that failed (for which the run inserted no records into the destination file or database table).
- The total of all values of the extracted numeric property, if the extract specifies a numeric property in the **Include check total property** field.

If you save the manifest to CSV or to a database, the manifest also includes for each destination CSV file or table:

- The unique identifier for the run.
- The name of the destination table.
- The number of INSERTs committed to the table.

The extract writes the manifest records as a single commit at the end of the extract. If there is an error during this commit, the extract writes the manifest information to the log file.

Notes about output

- When scalar properties in an embedded value list, page list, value group, or page group have been selected for extraction to a CSV file or to a database, BIX generates a separate CSV extract file or a separate database table for the embedded properties. BIX promotes scalar properties of embedded pages (not page lists or page groups) to the parent node.

- When you extract data for insertion into a database, text values longer than the defined length of the destination database column are automatically truncated. The "+" character is appended to the end of the extracted text value to indicate that it has been truncated.
- The entry in the PegaBIX log file for each run contains the BIX parameters used for that run, except for any password provided. For more information, see Command line BIX extracts.
- When extracting Boolean properties to a database, top-level properties are extracted as 0 and 1, and value list and value group properties are extracted as true and false.

– Understanding the Execution History tab

The **Execution History** tab lets you view the results of previous runs of the extraction. Double-click any row to view more details.

This read-only information is drawn from the **pr_extract_time** table in the PegaRULES database. The time values are in terms of the time zone where the computer server is located.

Field	Description
Create Date Time	The time and date that the extract session started.
Class and Purpose	The class of the properties being extracted and the name of the Extract rule.
System Name	The name of the Pega Platform system that ran the extraction.
% Completed	How much of the extraction has been completed.
Status	Whether the extract succeeded or failed.

Rule history

From the **History** tab of a rule you can track the changes that have been made to the rule over time. By viewing the history, you can see detailed information about each change to the rule, such as when it was changed and by whom. You can also add a memo to history records for future reference, compare historical versions of the rule, and restore a previous version of a rule.

For the rule history, the system converts dates and times from the internal UTC format to your time zone, which might be different from the time zone of the person who updated the rule.

The system saves rule history details as instances of the History-Rule class. By default, instances of the History-Rule class correspond to rows in the pr4_history_rule database table.

If your application implements the optional security audit feature, the History details section can identify which values were added, updated, or removed from a rule or data object.

If the rule was imported from another Pega Platform system by using the Import Archive tool, the History details section reflects only the changes that were made after the rule was imported to this Pega Platform system.

More about Extract rules

Testing

After you save an Extract rule, you can test the extraction.

1. Click **Run**.
2. Enter a unique identifier and the maximum number of records to extract.
3. Click **Perform Extract**.
4. Review the resulting data and log files.

Scheduling Extract rule runs

For more efficient operation, you should schedule run of an Extract rule using a Pega Platform agent or from a Windows or UNIX/Linux command-line. See Running the command-line extract process and Running an Extract rule using a Pega Platform agent.

For more efficient operation, you should schedule run of an Extract rule using a Pega Platform agent.

Exception handling

Executing BIX may result in errors or exceptions being reported. These exceptions may relate to syntax errors (in command-line execution), database constraint violations (when exporting to a database), or similar issues. See BIX logging and error handling.

Rule operation

Extract rules can operate on the BLOB column (Storage StreamStorage Stream) of tables in the PegaRULES database or they can be used to extract data from tables with no BLOB column. However, when a class table includes a BLOB (pzPVStream column), extraction always is performed from that BLOB value, and not from other database columns.

If the extract rule class is a class group, the query on the source database gets all WorkObject instances of all classes within the class group. If the extract rule class is not a class group, the query on the source database only gets instances of the particular class.

Forward and backward chaining during BIX extract

In the prconfig file, a setting governs whether forward and backward chaining are enabled during a BIX extract. You should disable forward and backward chaining for better throughput when running BIX.

Configuring the extraction environment

Before you run a BIX Extract from the command line, several configuration files must be updated to specify the source and target databases, configure logging, configure the database holding the engine code,and set up split schema. Additionally, you can create a pegarules.keyring file to support encryption of the database password that must be supplied with the database connection settings in prconfig.xml.

Complete the following tasks to set up a BIX Extract process to run from the command line.

- Configuring the source database
- Optional: Configuring the target database connection settings
- Optional: Configuring optional prconfig.xml settings
- Configuring BIX logging
- Specifying the database holding the engine code
- Setting up split schema
- Enabling password encryption
- Preparing a target database

Configuring the BIX source database

Edit the prconfig.xml file provided in the configuration directory of your BIX distribution, adding database connection settings that specify the source Pega Platform database where data is extracted.

The prconfig.xml file contains sample database connection settings for a SQL database that you can modify for your Pega Platform database.

The following steps outline the procedure for specifying typical database connection settings. Consult your DBA to resolve requirements specific to your installation. More information is available in PDN article: *How to configure non-J2EE database connections in the prconfig.xml file*.

1. In the database/drivers element, change the value to the appropriate driver for your database. For example:

Database driver	*Value*
Oracle 9i/10g	oracle.jdbc.OracleDriver
IBM DB/2 Type 4	com.ibm.db2.jcc.DB2Driver
SQL Server 2005	com.microsoft.sqlserver.jdbc.SQLServerDriver
SQL Server 2000	com.microsoft.jdbc.sqlserver.SQLServerDriver

2. In the database/databases/PegaRULES/url element, set the value to the connection URL for your database. Following are examples of the appropriate format for some supported databases:

Database driver	*Value*

Oracle	jdbc:oracle:thin:@//serverName:port/service-name-or-SID
DB/2 (Universal Driver)	jdbc:db2://serverName:port/dbName
SQL Server 2005	jdbc:sqlserver://your_sql_server_host:1433; SendStringParametersAsUnicode=false; SelectMethod=cursor; ProgramName=${NodeName}:${SystemName}.${ConnectionID}

3. Set the value of database/databases/PegaRULES/userName and database/databases/PegaRULES/password to the user name and password of a user for the Pega Platform database.
4. Set the value of database/databases/PegaDATA/userName and database/databases/PegaDATA/password to the user name and password of a user for the Pega Platform database.

Configuring the BIX target database connection settings

If you want to use the Database Schema output format, add a second set of database connection settings that specify the target database where the extracted data is written to the prconfig.xml file provided in the configuration directory of your BIX distribution.

The BIX Database Schema output format writes the extracted data directly to a target database. To support this output, you must provide a second set of database connection settings specifying the target database where the data is to be loaded.

1. In the BIX prconfig.xml file, copy and paste the database connection settings that you created for the source database.
2. In each element name, change PegaRULES to the name of the database that you specified as the Output Database Name in the Extract rule. Refer to the following examples:

Original path	*Updated path*
database/databases/PegaRULES/url	database/databases/TargetDB/url
database/databases/PegaRULES/userName	database/databases/TargetDB/userName
database/databases/PegaRULES/password	database/databases/TargetDB/password

3. Repeat the procedure used for the source database connection to set the target database connection settings to the appropriate values.

- In addition to specifying the database connection settings, Oracle and UDB databases require some additional configuration. Consult your DBA to resolve requirements specific to your installation. More information is available in PDN article *How to configure non-J2EE database connections in the prconfig.xml file*.

Configuring optional prconfig.xml settings

You can add optional parameters to the prconfig.xml file that filter the data by any property, including date ranges, enable or disable chaining, and specify whether records added to the database while the extract is running are included in the extract.

Add the following settings to the end of the prconfig.xml file:

- **Optimize XML output** – Set to true by default to facilitate extracting a large number of class instances to XML: <env name= "compatibility/BIXUseOptimizedClipboardXML" value= "true" />
- **Disable forward chaining** – Enable or disable forward chaining calculations of properties when loading the clipboard for BIX extract: <env name= "compatibility/BIXdisableForwardChaining" value= "true">

  The default setting is TRUE.
- **Disable backward chaining** – Enable or disable backward chaining of properties when loading the clipboard for BIX extracts: <env name="compatibility/BIXdisableBackwardChaining" value = "true" />

  The default setting is TRUE.
- **Disable declarative processing** – Enable or disable declarative processing: <env name="declarative/enabled" value = "false" />

  Only use this setting when you are running BIX from the command line. If you use this setting on a node to improve performance, you cannot open or create rules on that node.
- **Include class instances added during a batch run** – Enable or disable whether class instances added to the Pega Platform database after a batch run begins execution are included in the extract: <env name="bix/useHistoryClasses" value="true" />

  bix/useHistoryClasses only works when using the **Use Last Updated Time as Start** filter option. If command line parameters such as –u, -U, -d, and -D are passed while running the BIX script, bix/useHistoryClasses will not have any effect even if it is set to true.

Configuring BIX logging

prlog4j2.xml is a log4j configuration file that enables two log files for the extraction process, PegaBIX and PegaBIX-ALERT. By default, the log files are written to the directory where the extraction process is started. Ensure that you set logging levels appropriately, because turning up logging levels to DEBUG or ALERT can lead to performance issues, including slower extracts or full log files.

Modify the FileNamePattern for the PegaBIX and PegaBIX-ALERT logs to change the location to which they are output.

Specifying the database with the engine code

Pega Platform stores engine code in the database as a CodeSet. Edit the prbootstrap.properties file provided in the configuration directory of your BIX distribution to add database connection settings that specify the Pega Platform database holding the engine code.

1. Specify the database connection settings using the naming convention: <unique-identifier>,<dbtype>.<property>

   where:

   - <unique-identifier> is a distinct value that together with <dbtype> identifies a specific data source.
   - <dbtype> refers to a set of common properties associated with a given database type. There are four values that may be used:
     - oracle
     - db2
     - mssql
     - derby
   - <property> is one of the key values needed to access the data source.

   For example, to connect to an Oracle database:

   ```
   com.pega.pegarules.bootstrap.allclasses.dbcpsource=example.oracle
   example.oracle.url=jdbc:oracle:thin:@//localhost:1521/codebase
   example.oracle.username=user
   example.oracle.password=pass
   oracle.jdbc.class=oracle.jdbc.OracleDriver
   ```

2. Specify any additional properties needed to connect to the database using the connectionProperties entry and provide a semicolon-separated list of values. For example:

   example.oracle.connectionProperties=oracle.jdbc.V8Compatible=true

- Set the system property com.pega.pegarules.bootstrap.ignorejndi to TRUE when using Pega-managed connections. This tells Pega Platform to ignore the JNDI data source information in the file and to use the Pega-managed connection settings instead.

Setting up split schema

Both the prconfig.xml and prbootstrap.properties files must be updated when using split schema.

1. Add the following environment variables to the prconfig.xml file:

   env name="database/databases/PegaRULES/defaultSchema" value=" <rulesSchema_Name> "/>

   env name="database/databases/PegaDATA/defaultSchema" value=" <dataSchema_Name> " />

2. For a split schema setup, add the following properties to the prbootstrap.properties file:

   com.pega.pegarules.bootstrap.allclasses.schema=<rulesSchema_Name>

   com.pega.pegarules.bootstrap.datatables.schema=<dataSchema_Name>

Enabling password encryption

BIX lets you pass in a Pega Platform username and password to rule-resolve the Rule-Admin-Extract rule to run. If you want to enable security for the database username and password, you can implement JCE "keyring" encryption by creating a pegarules.keyring file. You can use keyring encryption to encrypt the username and password.

1. Run the KeyringImpl Java class, which accepts three parameters and generates the keyring file. The parameters are:
   - The path where the pegarules.keyring file is generated (including the filename)
   - The path to the prconfig.xml file
   - The path to the BIX distribution directory
2. To encrypt BIX passwords in the same file, pass an additional argument named "bix". Running the KeyringImpl Java file with this additional parameter prompts the user for the BIX username and password. The file encrypts the information that you enter.

- The runPega batch script assumes that the prconfig.xml provided as an argument to the Keyringimpl class is the same as the prconfig.xml file in the prweb/web-inf/classes folder. If the prconfig.xml file is in a different location, use the Java command directly as instructed in the PDN article *Creating a custom cipher in Pega Platform*.

Preparing a target database

Before extracting BIX data to the target database, you must create a schema to match the extracted data. You can use BIX to generate an SQL file that creates the appropriate schema.

1. Call the ExtractImpl class with the parameter -X <Path><FileName> to generate a DDL for the target database.
2. Run the SQL script against the target database using your database tools.

To use the -X parameter, the output format of the Rule-Admin-Extract rule that you specify must be Database schema. The user must also have Write access to the directory. If the output format is XML or CSV, an error message displays:

BIX cannot generate a database schema when the selected output type is set to CSV or XML.

Optional command-line BIX parameters

When running BIX from the command line or from a Pega Platform agent, you can input various parameters to filter what is extracted from your application.

Parameter	Description
-a <username>	Rule resolve Rule-Admin-Extract instances based on the user access group. This option is not available when running an Extract rule with pxExtractDataWithArgs.
-b <number of updates before a batch is issued>	Specify the batch size (the number of extracted class instances) for commits to a relational database. This option only applies to databases with JDBC drivers that support batch updates. This option is not available when running an Extract rule in the Pega Cloud.
-B	Specify how to format extracted Boolean values. The options are: • 01 • YN • YesNo • yesno • TF • TrueFalse • truefalse • YESNO • TRUEFALSE
-c	Include children classes. Use this parameter to extract the data for each sibling class of the class (class group/non-class group) for which the extract is defined.
-d <start date>	Only extract instances with a pxCreateDateTime equal to or greater than <start date>. <start date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS". Clear the **Use Last Updated Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter, because it does not override the check box setting.
-D <end date>	Only extract instances with a pxCreateDateTime equal to or less than <end date>. <end date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS". Clear the **Use Last Updated Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter, because it does not override the check box setting.
-f	Force the extract to stop at the first error that it encounters. This option only works with a batch size greater than 1. See BIX logging and error handling for information on conditions where encountering an error does not force the extract to stop. This option is not available when running an Extract rule in the Pega Cloud.
-g <start date>	Only extract instances with a pxCommitDateTime equal to or greater than <start date>. <start date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS" and relative to the local time zone.

	Clear the **Use Last Update Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter, because it does not override the check box setting.
-G <end date>	Only extract instances with a pxCommitDateTime equal to or less than <end date>. <end date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS" and relative to the local time zone. Clear the **Use Last Update Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter, because it does not override the check box setting.
-i <InstanceName>	Specify the pxInsName of the Rule-Admin-Extract or a comma-delimited list of Rule-Admin-Extract instances. This option is not available when running an Extract rule with pxExtractDataWithArgs.
-I <path and filename>	Use an Extract rule specified in an external file in XML format. See Using a stand-alone command-line BIX extract process. If an extract has any filter that uses unexposed property references that are passed in by the external file, then extraction fails with the following exception: "Please enter or select a property that is exposed as a column in the database."
-J	Specify the time zone that you want to convert DateTime properties to when extracting to XML or CSV output. You can specify the time zone in the following formats: • Time zone with an ID: For example, to specify America/Los_Angeles. Specify this time zone as "-J""America/Los_Angeles" • Three letter timezone: For example, to specify India Standard Time, use "-J" "IST" • UTC+/-dd:dd: Specify UTC plus or minus hours and minutes. For example, "-J" "UTC+5:30" If not specified, the default is GMT.
-k <Identifier>	Add a unique identifier for this run. Use alphanumeric characters or underscores only with no spaces.
-l <delimiter>	Specify a user-defined character as the delimiter separating extracted data. The following characters are not available for use as delimiters, as they signify other things to BIX: • Double-quote (") • Asterisk (*) • Hyphen (-) BIX also does not support multi-line delimiters. If the delimiter length is greater than 1, then BIX reports an error.
-n	Retrieves class instances that satisfy the filter conditions with an Extract rule from the Pega Platform production database without sorting them by how recent they are. This improves performance, but makes it impossible to perform a partial re-extraction following any errors.
-o	Overrides the default setting that combines filter conditions specified on pxCreateDateTime and pxUpdateDateTime using the AND Boolean operator and instead combines them using the OR Boolean operator.
-p <password>	Rule resolve Rule-Admin-Extract instances based on the user access group. This option is not available when running an Extract rule with pxExtractDataWithArgs.
-P	Overrides the output directories specified on the File Specification tab in the Extract rule form. When specified, all output files are directed to this location. This option is not available when running an Extract rule with pxExtractDataWithArgs.
-s <number>	No longer supported.
-t <threads>	No longer supported.
-T <tenantID>	Required option for using BIX in a multitenant environment. This option is not available when running an Extract rule with pxExtractDataWithArgs.

-u <start date>	Only extract instances with a pxUpdateDateTime equal to or greater than <start date>. <start date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS". Clear the **Use Last Updated Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter because it does not override the check box setting.
-U <end date>	Only extract instances with a pxUpdateDateTime equal to or less than <end date>. <end date> must be in the format "20110823T164017.000", following the Java pattern "yyyyMMddTHHmmss.SSS". Clear the **Use Last Updated Time as Start** check box on the **Filter Criteria** tab of the Extract form when you use this parameter because it does not override the check box setting.
-x	Include header and footer information for XML or CSV output.
-X <Path><FileName>	Generate only a set schema file for the export. No data is exported. To use the -X parameter, the output format of the Rule-Admin-Extract rule that you specify must be **Database schema**. If the output format is XML or CSV, a PRRuntimeException is generated. This option is not available when running an Extract rule in the Pega Cloud.
-z <start key>	Only extract instances with a pzInsKey equal to or greater than <start key>. The comparison is character by character with the pzInsKey and is not an integer comparison with the work object's ID.
-Z <end key>	Only extract instances with a pzInsKey equal to or less than <end key>. The comparison is character by character with the pzInsKey and is not an integer comparison with the work object's ID.

The pzInsKey values of class instances whose extraction resulted in errors are logged automatically. You do not need to set a flag to enable this behavior.

- When running a BIX extract from command-line on UNIX using the -u, -U, -d, or -D parameters, you need to set the timezone using a JVM argument, such as: -Duser.timezone="America/New York". For more information, see PDN article *Formatting time in BIX*.

Running an Extract rule using a Pega Platform agent

To schedule an extract from within the Designer Studio, modify the RunExtract agent that is in the Pega-BIX ruleset to periodically invoke the pxExtractDataWithArgs activity to run a BIX Extract rule. This activity takes as arguments the class and name of the Extract rule to run and, as a single string, the list of BIX command-line parameters desired for each run. The agent must only be enabled on the dedicated node for BIX processing to help avoid any negative performance impacts to production while the Extract rule is running.

The agent/threadpoolsize setting in the prconfig.xml file controls the number of threads available in the agent pool, and consequently, the number of agents that can be run concurrently. If you schedule more agents to run concurrently than the value of the agent/threadpoolsize setting, you might get a stale thread error. Increase the value of the agent/threadpoolsize setting. The default value of the agent/threadpoolsize setting is 5 and the maximum value is 10.

1. On the **Schedule tab** of the Agent rule, enter Rule-Admin-Extract in the **Agent Name** field.
2. In the **Mode** field for the agent, select **Advanced** unless you are creating queue items that a standard mode agent requires.

   Agents in standard mode that do not have queue items created for them will appear to have run, but the activity is not executed.
3. Enter pxExtractDataWithArgs in the **Activity name** field.
4. Click **Parameters** to enter command line parameters in the following format when you are using one or more parameters.

   If you are using an agent that calls pxExtractDataWithArgs, place quotes around the values. The extract fails if you do not use quotes. For example:

   z "TEST-TESTAPP-WORK G-12" -Z "TEST-TESTAPP-WORK G-12"

   If you are using an agent that calls a wrapper activity that calls pxExtractDataWithArgs, place quotes around the values and around the entire parameter string. Use the backslash to escape the quotes around the values. For example:

   "z \"TEST-TESTAPP-WORK G-12" -Z \"TEST-TESTAPP-WORK G-12\""

   The -i, -a, -p, -T, and -P command-line parameters are not supported for pxExtractDataWithArgs. The -X, -b, and -f command-line parameters cannot be used if pxExtractDataWithArgs is used on a Pega Cloud instance.
5. On the **Security tab**, configure the access group to be the application access group.
6. Select **Bypass activity authentication**.
7. Create an Agent schedule and map the agent to the BIX node type.

See Developer Help topics Optional command-line BIX parameters and Agents rules for more information.

See Optional command-line BIX parameters and Developer Help topic Agents rules for more information.

Running an Extract rule manually in Designer Studio

You can run a BIX extract manually from within a Pega Platform application.

1. From the Records Explorer, select **SysAdmin > Extract** and select the Extract rule that you want to use.
2. On the rule form, click **Run**.
    - Extracting to .csv and .xml output.
        1. Provide the **Number of Records to Update at a Time**. To run the extract in non-batch mode for a single record at a time, set the number of records to update at a time to 1.
        2. Provide a **Unique Identifier** for the extract.
        3. Click **Perform Extract** to begin extracting data from your application. A progress bar informs you when the extract is complete.
    - Configuring schema for extracting to a database before you run an extraction for the first time.

      Before extracting data to a target database, create a schema to match the extracted data.

        1. Provide the **Number of Records to Update at a Time**. To run the extract in non-batch mode, a single record at a time, set the number of records to update at a time to 1.
        2. Provide a **Unique Identifier** for the extract.
        3. Click **Generate Schema DDL**.
        4. Provide a location and a name to use when saving the DDL.
        5. Click **Perform Extract** to generate the DDL and save it in the specified location.
        6. Run the generated DDL manually in the required database to create the table.

           If the names of any selected properties correspond to reserved keywords in the destination database, manually update the DDL file to either change the property names or surround them with quotation marks. If you do not update the DDL, a database error will occur.
    - Extracting to a database after schema has been created.
        1. Provide the **Number of Records to Update at a Time**. To run the extract in non-batch mode, a single record at a time, set the number of records to update at a time to 1.
        2. Provide a **Unique Identifier** for the extract.
        3. Clear the **Generate Schema DDL** check box.

4. Click **Perform Extract** to begin extracting data from your application into the created table. A progress bar informs you when the extract is complete.

After the extract is complete, you can access the output data at the location (or locations, for XML output) that you specified in the File Specifications tab of the rule.

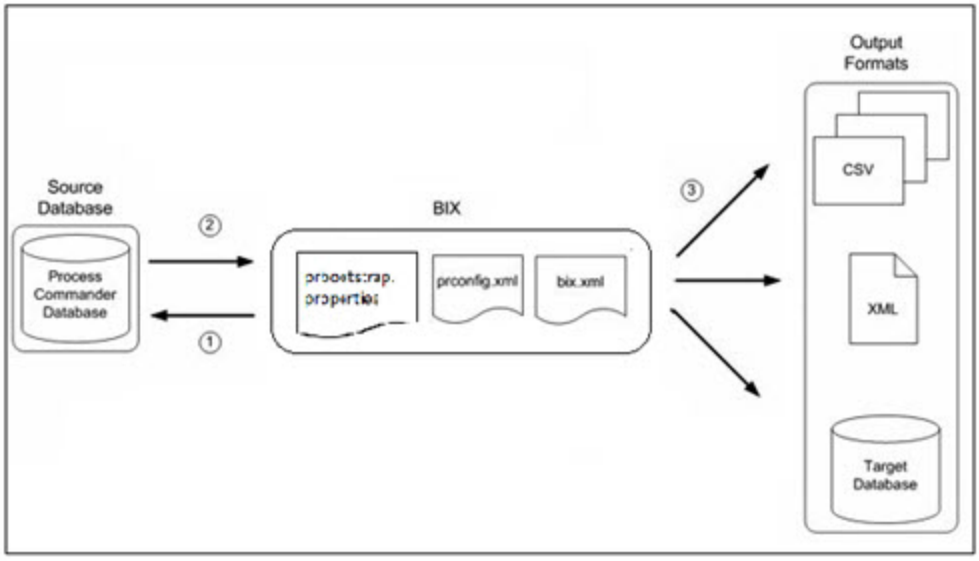
Command-line BIX extractions

In addition to running a BIX extract from within the Designer Studio, you can directly call the ExtractImpl Java class in the Pega Platform application libraries to define and run an extract process. Specify the data to be extracted and the output format by referencing an Extract rule that you have created in the Pega Platform. ExtractImpl can either access the Extract rule in the Pega Platform database or read an XML file containing the XML representation of the rule. BIX in the command-line executes in its own Java Virtual Machine (JVM) instance.

During command-line BIX extraction, a stand-alone instance of the PegaEngine is started, therefore the database user running the command-line BIX extraction must have all of the privileges needed to start the engine.

You can add other optional parameters that filter the data by any property, including date ranges, and specify other processing options. For more information, see Setting optional command-line BIX parameters.

Running an Extract rule from the command line works in the following way:



1. BIX accesses the Pega Platform (source) database using the information defined in the BIX prconfig.xml file.
2. BIX prbootstrap.properties starts up the PegaEngine with the Pega Platform (source) database details provided and gets the ExtractImpl.class code.
3. Using the information configured in the bix.xml file and specified in the Extract rule, BIX extracts the specified data from the Pega Platform (source database).
4. The extracted data is output in the output format specified in the Extract rule.

- The output data includes a time stamp and a batch identifier. These help when you want to compare information from different extracts or isolate information from a specific extract.

Running the command-line extraction process

Once you have configured your environment, you can run the command-line extract process.

From the command line or in a script, make a Java call to com.pega.pegarules.data.internal.access.ExtractImpl, supplying arguments to specify:

- JVM memory options – Ensure that your JVM is allocated with enough memory for the operation by setting:
  - The initial Java heap size to 512m (-Xms512m)
  - The maximum heap size to at least 768m (-Xmx768m)
- System properties - Provide the location of the Pega Platform configuration files prconfig.xml, prlog4j2.xml, prbootstrap.properties, and if used, pegarules.keyring. For example:
  - -Dpegarules.config=config/prconfig.xml
  - -Dpegarules.logging.configuration=config/prlog4j2.xml
  - -Dcom.pega.pegarules.bootstrap.properties.url=config/prbootstrap.properties
  - -Dcom.pega.pegarules.bootstrap.ignorejndi=true
  - -Dpegarules.keyring=config/pegarules.keyring
- An Extract rule definition - Provide either:
  - The -i switch and the name (pxInsName) of the Extract rule in the source Pega Platform database.
  - The -I switch and the path and filename of the XML file that contains the XML representation of the Extract rule.
- Optional parameters - Specify ranges of the data to be extracted and other processing configuration. For example, you can:
  - Filter the data by key or date range
  - Specify the number of statements to issue before committing to the target database, or number of records to process before a batch commit,
  - Provide a Pega Platform username and password to allow rule resolution of the Extract rule in the Pega Platform database.

To use the last option listed under "Optional parameters," call the main program com.pega.pegarules.pub.PegaRULES, and provide the username and password to ExtractImpl as the first argument. The resulting statement might look like this:

```
java -Dpegarules.config="./config/prconfig.xml" -

Dpegarules.logging.configuration="./config/prlog4j2.xml" -
Dcom.pega.pegarules.bootstrap.properties.url=”./config/prbootstrap.properties” -
Dcom.pega.pegarules.bootstrap.ignorejndi=true
```

com.pega.pegarules.pub.PegaRULES com.pega.pegarules.data.internal.access.ExtractImpl -a $USERNAME$ -p $PASSWD$ -i

PegaSample!SampleBIX_CSV

You can add a unique identifier for the current extract process.

See Setting optional command-line BIX parameters for details on all available parameters.

The engine Java classes invoked by BIX are stored in the database and not in the file system. The ExtractImpl class cannot be directly run using the Java interpreter. Instead, run the PegaRULES class and pass the ExtractImpl class as an argument. The PegaRULES class is part of the prbootstrap.jar library.

Example extract on a DB2 system

To run the extract "PegaSample!SampleBIX" on a DB2 system:

java –Xms512m –Xmx768m –classpath".;lib\prbootstrap.jar;lib\prdbcp.jar;lib\db2jcc_v95.jar;lib\jsr94-1.0.jar;%CLASSPATH%" -Dcom.pega.pegarules.bootstrap.properties.url=config\prbootstrap.properties -Dpegarules.config=config\prconfig.xml -Dpegarules.logging.configuration=config\prlog4j2.xml -Dcom.pega.pegarules.bootstrap.ignorejndi=true com.pega.pegarules.pub.PegaRULES com.pega.pegarules.data.internal.access.ExtractImpl -i PegaSample!SampleBIX

In this instance, you provide the appropriate path and JDBC driver name.

Stand-alone command-line BIX extractions

In certain circumstances, you may want to establish BIX as a stand-alone command-line process, rather than running it from within a Pega Platform system. For example, you may want to avoid disrupting a production instance of Pega Platform, or you may want to use the same instance of BIX to extract data from multiple Pega Platform instances.

Establishing a stand-alone command-line BIX extraction process

To establish a stand-alone command-line BIX extract process:

1. Download the BIX distribution archive to your system and expand the archive into a directory. The config directory in this distribution contains the configuration files that you set up to support the extraction process and the lib directory contains the Pega Platform library files that you run against. The distribution also contains a sample Ant build file to run an extract process.
2. Configure a JVM environment with access to the Pega Platform JAR files and the appropriate database driver files. You must have a Java Runtime Environment (JRE) distribution in place in the command-line system that allows you to make Java calls. Use JVM version 1.5, 1.6 or 1.7.

   Ensure that your classpath:

   - Contains the JAR files in the lib directory of your BIX distribution on the command-line system. This directory contains the Pega Platform libraries.
   - Includes the location of the Java Database Connectivity (JDBC) driver JAR files for your database. For more information on the appropriate driver for your database, see Configuring database support on your application server in the *Installation Guide*. See the *Platform Support Guide* for additional information.
3. Configure the prconfig.xml, prbootstrap.properties, and prlog4j2.xml files to specify the source and target databases, BIX logging, and other properties to complete the command-line environment. See Setting up command-line extracts.

- The engine classes are stored in the database and not in the file system. The ExtractImpl class cannot be directly run using the Java interpreter. Instead, run the PegaRULES class and pass the ExtractImpl class as an argument.

Example shell script file

The following is a sample ksh script calling the ExtractImpl class with an XML representation of the Extract rule. This sample assumes that the script is being run from the BIX distribution directory and that the CLASSPATH environment variable is set to include the JAR files in the BIX distribution's lib directory.

Notes:

1. JVM arguments to set the JVM memory parameters.
2. JVM arguments to set system properties specifying the location of the prconfig.xml and prlog4j2.xml files.
3. JVM arguments to set system properties specifying the location of prbootstrap.properties.
4. JVM arguments to set the classpath to include the jar files in the BIX distribution's lib directory and to specify the location of the JDBC driver jar file needed to run the Extract rule.
5. ExtractImpl parameters: -I switch with the name of the XML file containing the XML representation or the Extract rule.

Sample ant build files

The following sample Ant build files execute a call to the ExtractImpl class referencing an Extract rule in the Pega Platform database. The BIX distribution also contains a sample BAT script that calls a similar file.

See the readme.txt file in the BIX distribution directory for details on all of the files included in the distribution directory.

```
<?xml version="1.0" encoding="UTF-8"?>

<project name  = "Sample" default = "Bix" >

<target name="Bix">
        <property name="ARGLINE" value="PegaSample!SampleBIX"/>

        <echo message="Performing Export"/>

        <condition property="javaVersion" value="5">
            <matches string="${java.version}" pattern="1.4.*" />
        </condition>

        <condition property="javaVersion" value="5">
            <matches string="${java.version}" pattern="1.5.*" />
        </condition>

        <condition property="javaVersion" value="6">
            <matches string="${java.version}" pattern="1.6.*" />
        </condition>

        <echo message="using java version ${javaVersion}"/>

        <java classname="com.pega.pegarules.pub.PegaRULES" failonerror="true" fork="true">
                <classpath>
                        <fileset dir="lib/">
                            <include name="*.jar"/>
                        </fileset>
                        <pathelement path="."/>
                        <pathelement path="${java.class.path}"/>
                </classpath>
                <jvmarg value="-Xms128m"/>
                <jvmarg value="-Xmx512m"/>
                <jvmarg value="-Dcom.pega.pegarules.bootstrap.properties.url=config/prbootstrap.properties"/>
                <jvmarg value="-Dpegarules.config=config/prconfig.xml"/>
                <jvmarg value="-Dcom.pega.pegarules.bootstrap.ignorejndi=true"/>
                <jvmarg value="-Dpegarules.logging.configuration=config/prlogging.xml"/>
            <arg value="com.pega.pegarules.data.internal.access.ExtractImpl" />
            <arg line="-i"/>
            <arg value="${ARGLINE}" />
        </java>
</target>
</project>
```

Notes:

1. Set the ARGLINE Ant property to the pxInsName of the Extract rule you wish to execute.
2. Java call of the ExtractImpl class.
3. JVM arguments to set the JVM memory parameters.
4. JVM arguments to set system properties specifying the location of the prconfig.xml and prlog4j2.xml files.
5. Set the classpath to include the JAR files in the BIX distribution's lib directory.
6. ExtractImpl parameters: -i switch with the pxInsName of the Extract rule as set in the ARGLINE property.

Creating an XML representation of an Extract rule

When you run BIX from the command line, BIX can execute an Extract rule that is stored as an external file in XML format. This lets customers dynamically create or modify Extract rules outside of the Pega Platform for use during batch extract runs.

To create an XML version of an Extract rule:

1. From the Records Explorer, select **SysAdmin > Extract** and select the Extract rule to run.
2. On the rule form, click **Actions > View XML**. An XML version of the Extract rule displays.
3. Save the displayed content as an XML file.

Unique run identifier

When using BIX, you can provide a unique run identifier for each BIX extraction run to a database or a CSV file, marking additions to both parent and child tables in the database, and parent and child nodes in the CSV file. This enables you to:

- Easily identify all inserted records for a particular BIX extract run, both in the main table for a class and in subsidiary tables created for page lists, page groups, and value lists in the class.
- Delete records from a failed BIX run so that an extract can be re-run without producing incorrect or redundant data.

Two fields have been added to the main table for each extracted class in the database, **pxExtractDateTime.TIMESTAMP** and **pxRunIdentifier VARCHAR(255)**.

The **pxExtractDateTime** field has also been added to all subsidiary tables for the class in the database.

The value of the **pxExtractDateTime** field matches the value of pyLastUpdateTime in the table **pr_extract_time** for the run. This allows you to perform a join action and obtain all of the records inserted for a BIX run.

You can also include a run identifier through the –k command-line option.

BIX can still run in backward compatibility mode for tables without new fields. If the new fields exist in the database, they are filled in by BIX; otherwise, BIX runs in backward compatibility mode.

BIX logging

The PegaBIX log file contains information about BIX extractions that can be useful to determine what caused a problem during a data extraction. BIX extracts and writes out a batch of records, one batch at a time. These events, including any error messages, are recorded in the PegaBIX log file.

- When BIX is run from within Designer Studio, any errors are displayed on-screen and are also logged.
- When BIX is run from the command line in batch execution mode, you can set the extract to stop when it encounters its first error or to continue processing, and log all errors for later consideration.

This functionality is available when you purchase and install the BIX application.

BIX error handling

Errors normally cause the BIX application to terminate, however, the application may continue to run following certain exceptions:

- An invalid class definition for a single class described in a set of XML files
- An invalid property definition
- An invalid property transformation to the specified data type

BatchUpdateException error

Committing an extract batch of class instances to a relational database may be interrupted because of issues with the data, syntax errors, a database constraint violation, or some other issue raised by the database.

Different databases respond differently to batch update exceptions. Generally, the database throws a BatchUpdateException error, which BIX records in the PegaBIX log file.

When an error occurs, BIX behaves differently depending on the database platform that it is working with: Oracle, MS-SQL, DB2 LUW, or DB2 for z/OS servers.

Troubleshooting BatchUpdateException for databases other than Oracle

The server continues processing the operations in a batch even if one of the operations caused a database error. The database returns an exception that contains the exact information of the record in error. This information can be used to pinpoint the failure to the pzInskey of the record involved.

In this scenario, BIX supports two modes of execution:

- Continue processing records even when there are failures (the default mode)
- Stop processing when the first error is encountered

In default mode, all successful inserts are committed and the pzInsKeys of the failing entries are listed in the log file.

To rectify the work object and rerun the extract, specify the same value for pzInskey in the –z and –Z options.

When the –f option is in effect, the extraction process terminates after the first error is encountered. All successful batch inserts up to that point are committed. The batch that caused the failure is rolled back completely.

Troubleshooting BatchUPdateException for Oracle

Oracle stops processing the remaining operations in the batch if it encounters an error. The exception returned by the database does not point to the exact operation that caused the exception. For that reason, all of the pzInsKeys of the records in the batch are included in the BIX error log. The records from the batch that were successfully committed are not indicated because the Oracle driver does not provide that information.

In this scenario, BIX supports two modes of execution:

- Continue processing the records even when there are failures (the default mode)
- Fail on first error

In default mode, all successful inserts up to the record that failed are committed and the entries in the batch that followed the error entry are not processed. All of the pzInsKeys of that batch are written out to the log file.

To rectify the error and rerun BIX for the remaining records in the batch, specify the pzInsKey range for the remaining records using the –z and –Z options.

- When the –f option is in effect, the extraction process terminates after the first error is encountered. All of the successful batch inserts up to that point are committed. The batch that caused the failure is rolled back completely.

BIX in the Pega Cloud

If you are licensed to run BIX, a separate node is set up for running batch processes such as BIX, when your Pega Cloud instance is provisioned. This node does not participate in the load-balancing of end-user sessions, which minimizes the effect of BIX processing on production performance.

BIX in the Pega Cloud is not supported in a multitenant environment.

There are several key differences between a customer running BIX onsite versus running BIX in a Pega Cloud instance:

- Output format options for extracted data are limited to either XML or CSV files, referred to as extract files. Writing extracted data directly to a database is not supported in the Pega Cloud. The **File Specification** tab for the Extract rule does not display because the extract files are not directly accessible and are instead automatically transported to your FTP site following extraction.
- Direct access to the Pega Cloud server file system is not allowed, and BIX cannot be run from an external command-line process.
- You can either obtain the extract files from the Pega SFTP server or the extract files can be sent to your SFTP site. All extract files from a BIX run are compressed and delivered as a single .zip file whose name consists of classname, extractname, and sequence number.

Running BIX in the Pega Cloud

You can either obtain BIX extract files from the Pega SFTP server or have the extract files sent to your SFTP server.

BIX in the Cloud is not supported in a multitenant environment.

To obtain BIX extract files from the Pega SFTP server:

1. Define a BIX Extract rule.
2. Schedule the Extract rule as described in Running a BIX Extract rule manually in Designer Studio.
3. Obtain the files from the Pega SFTP server.

To have the BIX extract files sent to your SFTP server:

1. Define SFTP related data instances.
2. Define a BIX Extract rule.
3. Schedule the Extract rule as described in Running a BIX Extract rule manually in Designer Studio.

Defining SFTP-related data instances

If you are run BIX in the Pega Cloud, you can either obtain BIX extract files from the Pega SFTP server or have the extract files sent to your SFTP server. If you want BIX extract files to be sent you your SFTP server, you must create an SFTP-related data instance.

1. Set up an SFTP site in your domain (if one does not already exist). Define a relative path on the destination SFTP site where the files will be sent by configuring the BIXFTPRelativeRemotePath Dynamic System Setting.
2. Within Designer Studio, configure the SFTP Server data instance BIXFTPServer on the **Environment** tab to point to your SFTP site.
3. Configure the Authentication Profile data instance BIXFTPAuthProfile with the login credentials of the SFTP Server, and save the instance.
4. Click **Test Connectivity** in BIXFTPServer to verify the configuration and save this instance. See Developer Help topic About FTP Server data instances. See Developer Help topic About FTP Server data instances.
5. Configure the Service Package data instance BIXFTPServicePackage with the operator's access group and save the instance.
6. Open the File Listener BIXFTPListener, clear the **Block Startup** check box, and then save the instance.
7. Open Listener Management in the System Management application and make sure that the BIXFTPListener file has been started.
8. Optional. Manually run an Extract rule to test the BIX extraction process and verify successful receipt of the extract file at your SFTP site. When running a manual test, define filter conditions in the Extract rule to ensure that only a small number of class instances are extracted.

BIX performance

Understanding the performance trade-offs of the BIX options will help you to maximize the throughput of data extracts using BIX. In addition, you may need to perform some tuning in the Pega Platform database.

Performance trade-offs

The three most important factors that determine the throughput of the BIX extraction process, (that is, how many instances can be extracted from a class per hour), are:

- **The number of instances extracted.** Use the options on the **Filter Criteria** tab of the Extract rule form to limit extraction to just those instances needed. In particular, use the incremental extraction option (the **Use Last Updated Time as Start** option on the Filter Criteria tab) to limit the instances extracted to those that have been created or changed since the last time the Extract rule was run:
    - If you select the incremental extraction option, a filter condition is automatically included during an extract that restricts extraction to class instances where the pxCommitDateTime property value is greater than or equal to the last date and time when the extract ran. pxCommitDateTime is automatically set for each class instance when it is changed in the database by the Pega Platform database regardless of how the data was created or changed (interactively by an end user, programmatically by an activity, through an import, or by any other means).

      The pxCommitDateTime property may sometimes need to be added as a database column to the class table. For example, it is not added automatically on upgrades. An extract using the incremental extraction property will fail if this column does not exist in the class table.

    - As an alternative to using the incremental extraction option, you can manually specify a filter condition that compares the symbolic value Last Extraction Time to any optimized DateTime property in the class. At runtime, this value is replaced with the date and time when the extract rule last ran. If you do so, this database column should be indexed for better performance.

    - Also note that when incremental extraction is performed, class instances with NULL values of the pxCommitDateTime property will be skipped and when using the -c command line option to extract from child classes, those child classes whose class tables do not include the pxCommitDateTime property will be skipped.

    - When it is necessary to perform extraction within a small time window, for example, extract all instances within 15 minutes to accommodate scheduled downstream processes, you may want to run multiple extracts from the same class in parallel. You can do this by using different filter criteria on each run to select distinct sets of class instances. For example, use filter criteria that select data for different product lines, geographies, etc.
- **The output format selected**. XML is much faster than CSV output or writing directly to a destination database, since data in embedded page lists and page groups does not need to be normalized as it is written.

- **The number of properties extracted, and how deeply embedded those properties are**. The more deeply nested the properties are in the class data model, the slower performance will be.
- For more details on these last two factors, see details of the BIX performance benchmark.

BIX performance recommendations

Optional performance settings

Set the following options to maximize BIX performance. These options are described in Configuring optional prconfig.xml settings:

<env name= “compatibility/BIXUseOptimizedClipboardXML” value= “true” />

<env name= “compatibility/BIXdisableForwardChaining” value= “true”>

<env name=”compatibility/BIXdisableBackwardChaining” value = “true” />

<env name="declarative/enabled" value= "false" />

BIX retrieval query optimization

When a BIX extract rule is run, it generates an SQL query to retrieve the instances of a class from which it will extract data. The filter conditions in this query are based on the selections made on the **Filter Criteria** tab of the rule form. To optimize the performance of this SQL query for large class tables, consider adding database indexes for the columns referenced in the filter criteria. If you are using the incremental extraction option, be sure that the pxCommitDateTime property is indexed.

Single class data extraction using multiple parallel batches

When you are extracting data from one class that has a large number of class instances and a small time window in which to run the batch, you can extract data using multiple parallel batches. To do this, run your Extract rule multiple times, starting at the same time, filtering each run to extract a different set of records. For example, you could filter on region, product line, etc.

BIX high-throughput data downloads in the Pega Cloud

High-throughput data downloads increase the speed of downloads up to 25 GBs of data per hour, improving extract performance. This option only applies to BIX extracts in the Pega Cloud from a BLOB-less class table in a Postgres database for CSV output. It is automatically used for these types of extracts.

There is, however, a tradeoff in functionality. When the high-throughput data download option is used, the following functionality is not available:

- -c command line option to include children classes
- -x command line option to include header and footer information in the output

- checksum in the manifest summary

If you need to use this functionality, you can disable high-throughput data downloads and revert to the older implementation by setting the bix/useHighThroughputDownloadForCSV setting in the prconfig.xml file to false.

BIX performance benchmark

The following performance metrics were captured by running the BIX from a Windows Server command line against a series of Process Commander extract rules (Rule-Admin-Extract) of varying complexities.

Test setup and benchmark results

Four different extract rules were used. The Simple extract rule has twelve properties of mixed types (Text, Decimal, Double, Date, DateTime, and others).

Complex Level 1 extract rules have the same properties as Simple with the addition of the Page List and Page Group properties. Each embedded page has twelve properties.

Complex Level 2 has the same layout as Complex Level 1 plus one additional layer of Page Lists and Page Groups.

Complex Level 4 has the same layout as Complex Level 2 plus two layers of Page Lists and Page Groups.

Each extract rule was executed three times, each with 100000, 200000, 300000, 400000 and 500000 records (work objects). The chart below presents a summary of the extract rules.

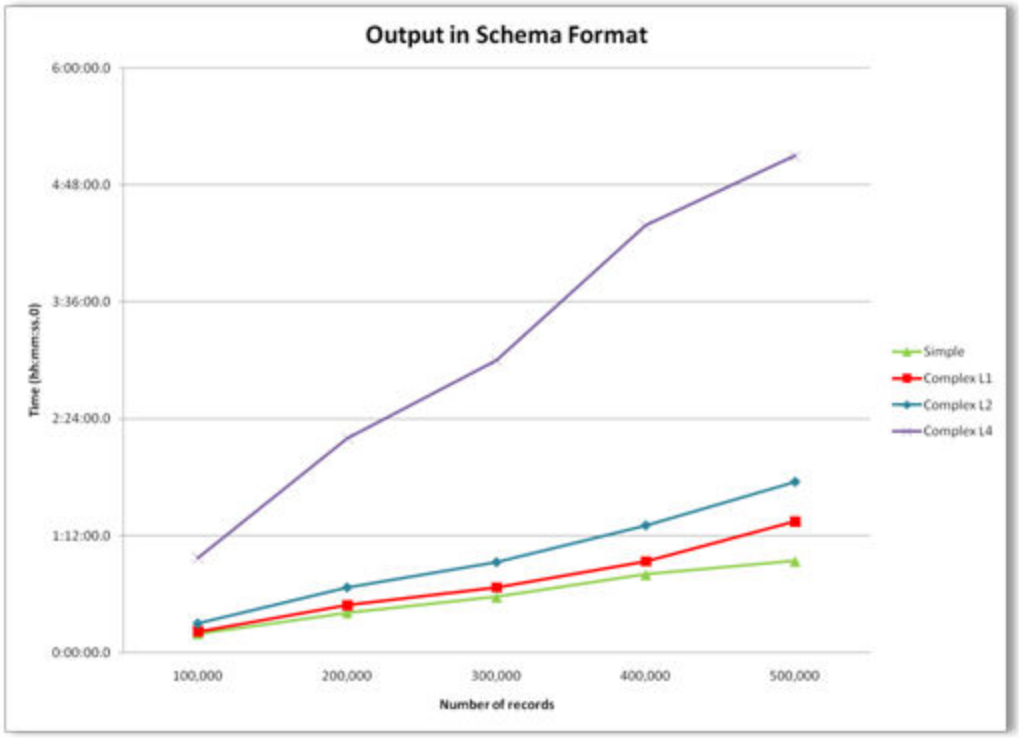
Data structure								
Name	Property	Page	Page list	Page group	Level 1 embedded page	Level 2 embedded page	Level 3 embedded page	Level 4 embedded page
Simple	X	X	X	X	X	X	X	X
Complex level 1	X	X	X	X	X	X	X	X
Complex level 2	X	X	X	X	X	X	X	X
Complex level 4	X	X	X	X	X	X	X	X

Data Structure	
CPU	Intel Xeon 2.5GHZ
Memory	15 GB RAM
Hard Drive Disk (HDD)	1.6 TB

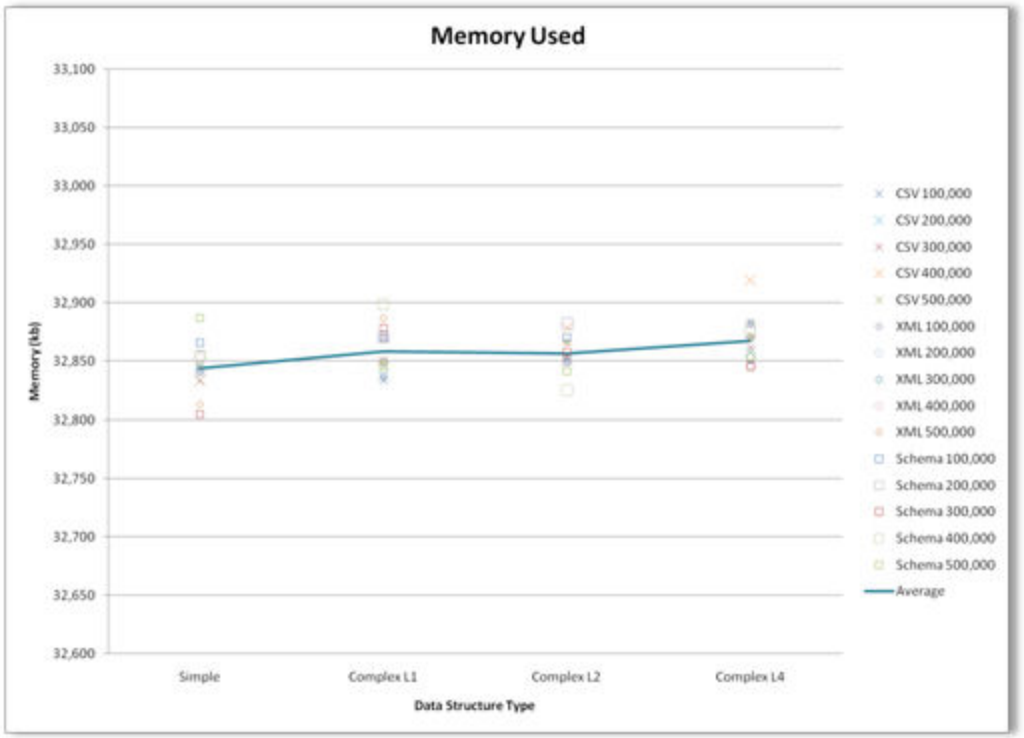
System Software	
Operating System	Microsoft Windows Server 2003 Service Pack 2 (Enterprise Edition)
Database	Oracle 10g
ANT Version	Apache-ant 1.7.1
JRE	JRE 1.6.0_07

Benchmark results

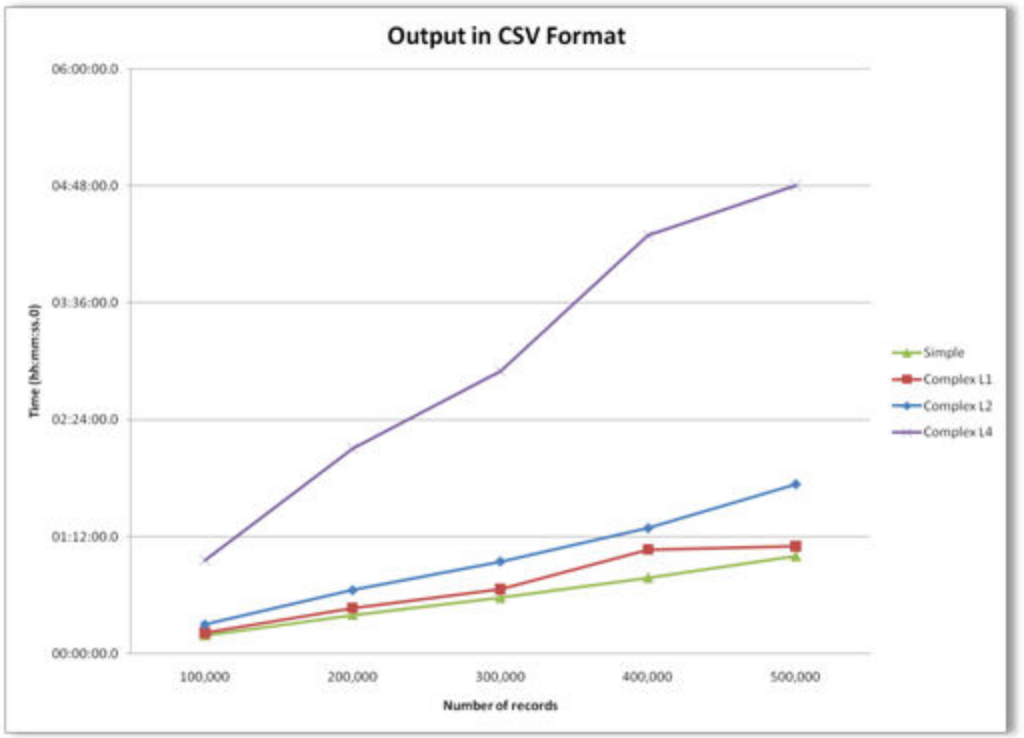
The following graph shows the performance of a Comma Separated Values (CSV) extract for each set of records. For example, 500,000 Complex Level 4 records were extracted into a CVS file in 4 hours and 48 minutes, or about 29 records per second.



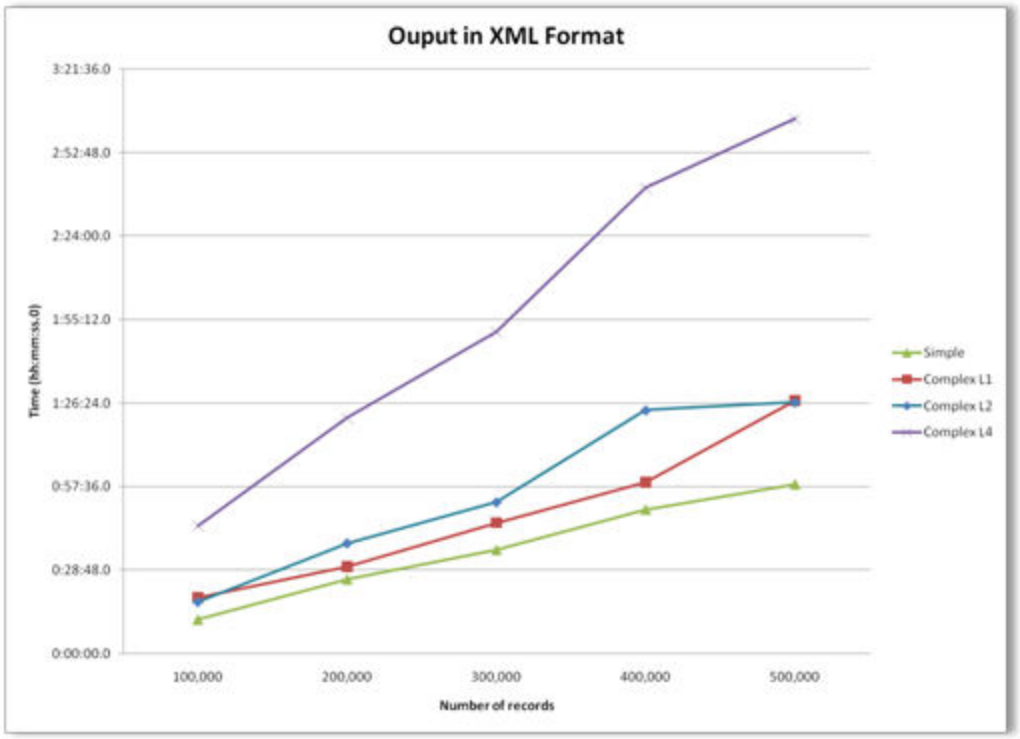
The following graph shows the performance of an XML extract for each set of records. For example, 500,000 Complex Level 4 records were extracted into an XML file in 3 hours, or about 46 records per second.



The following graph shows the performance of a Database schema extract for each set of records. For example, 500,000 Complex Level 4 records were extracted into a database schema file in 5 hours, or about 28 records per second.



The following graph shows the comparison between each type of extract for a Complex Level 4 data structure.



The following graph shows the amount of memory that was used during the extract.