public ETLStepAPIVersion getAPIVersion()	YellowfinAPIETLStepAPIVersionAPI
	`@Override`
	`public ETLStepAPIVersion getAPIVersion() {`
	`  return ETLStepAPIVersion.V1;`
	`}`

public Collection<ETLException> validate()	ETLExceptionETLExceptiongetInvalidConfigETLException()
	`@Override`
	`public Collection<ETLException> validate() {`
	`  List<ETLException> validationErrors = new ArrayList<>();`
	
	`  String exampleOption = this.getStepOption("APPEND_VALUE");`
	`  if (exampleOption == null) {`
	`  // Add a generic message "Step not Properly Configured"`
	`  validationErrors.add(this.getInvalidConfigETLException());`
	`  }`
	
	`  try {`
	`  Integer.parseInt(exampleOption);`
	`  } catch (NumberFormatException e) {`
	`  ETLException ve = new ETLException(ETLElement.STEP, getUuid(), null,`
	`"Option is not a number", e);`
	`  validationErrors.add(ve);`
	`  }`
	`  return validationErrors;`
	`}`

public Map<String, String> getValidatedStepOptions()	optionKeyoptionValueYellowfinthis.getStepOptions()
	`@Override`
	`public Map<String, String> getValidatedStepOptions() {`
	`    Map<String, String> stepOptions = this.getStepOptions();`
	`    String exampleOption = stepOptions.get("APPEND_VALUE");`
	`    if (exampleOption == null) {`
	`        // Remove option if the value is no longer set`
	`        stepOptions.remove("APPEND_VALUE");`
	`    } else {`
	`        try {`
	`            Integer.parseInt(exampleOption);`
	`        } catch (NumberFormatException e) {`
	`            // Remove option if the value is not an integer`
	`            stepOptions.remove("APPEND_VALUE");`
	`        }`
	`    }`
	`    // Return the map of valid options`
	`    return stepOptions;`
	`}`

<table>
<tr>
<td>public void setupGeneratedFields()</td>
<td>YellowfinETLStepMetadataFieldBean

<pre>
@Override
public void setupGeneratedFields() throws ETLException {
    if (getStepOption("NEW_FIELD") != null) {
        // The field was already set up.
        return;
    }

    ETLStepMetadataFieldBean newField = new ETLStepMetadataFieldBean();
    newField.setFieldName("Concatenated Field");
    newField.setFieldType(ETLDataType.TEXT.name());

    // The sort order is 0 based, so the new field will be at the end
    newField.setSortOrder(getDefaultMetadataFields().size());

    // Ensure that the new field is output from the step
    newField.setStepIncludeField(true);
    newField.setUserIncludeField(true);

    // This method assigns the field a new UUID and
    // adds a Step Option to help reference it elsewhere.
    this.addNewGeneratedField(newField, "NEW_FIELD");
}
</pre>

<pre>
this.addGeneratedField(newFieldBean, ETLFieldLinkType.DUPLICATE,
originalFieldUUID)
</pre>

<pre>
this.replaceDefaultField(fieldToReplace)
</pre>

linkFieldUUIDlinkType

<pre>
this.restoreReplacedField(replacementField)
</pre>
</td>
</tr>
<tr>
<td>public Integer getMinInputSteps()

public Integer getMaxInputSteps()

public Integer getMinOutputSteps()

public Integer getMaxOutputSteps()</td>
<td>Yellowfin/ETLStepCategory</td>
</tr>
<tr>
<td>YFLogger</td>
<td>YFLoggerYFLoggerlog4jclass

<pre>
private static final YFLogger log = YFLogger.getLogger(TestStep.class.
getName());
</pre>
</td>
</tr>
</table>

AbstractETLRowStep classprocessWireData()

processWireData()

processWireData()Wirewire**this.getWireForField(fieldUUID)**wirewirewire

- **List<ETLStepMetadataFieldBean>**wire
- booleanfalse
- **ETLExceptionInterruptedException**throwETLExceptionthrowYellowfin**this.throwUnhandledETLException(e)**)catchswallow interruptedExceptioncatch**java.lang.Exception**catchInterruptedExceptioncatchcatchthrow

```
} catch (InterruptedException e) {
    throw e;
} catch (Exception e) {
    log.error("Error: " + e, e);
    throwUnhandledETLException(e);
}
```

```
@Override
protected boolean processWireData(List<ETLStepMetadataFieldBean> fields)
                                  throws ETLException, InterruptedException {

    // The options should've been validated by the validate() method,
    // so no need for further checks here
    String appendFieldUUID = this.getStepOption("APPEND_FIELD");
    String newFieldUUID = this.getStepOption("NEW_FIELD");
    String appendValue = this.getStepOption("APPEND_VALUE");

    Wire<Object, String> appendFieldWire = this.getWireForField(appendFieldUUID);
    Wire<Object, String> newFieldWire = this.getWireForField(newFieldUUID);

    Object data = appendFieldWire.getValue();
    String newFieldData = null;
    if (data == null) {
        newFieldData = appendValue;
    } else {
        newFieldData = data.toString() + appendValue;
    }

    newFieldWire.send(newFieldData);

    return true;
}
```

processWireData()appendFieldUUIDnewFieldUUIDappendValueappnedFieldWirenewFieldWireprocessWoreData()

AbstractETLCachedSteprocessEndRows()SQL

processEndRow()

- processEndRows()wire

```
@Override
protected void processEndRows() throws ETLException, InterruptedException {

    // Get the first output flow;
    // Useful for most steps which have a single output
    String outFlow = getFirstOutputFlow();

    // The step outputs four Generated fields.
    List<String> orderedFieldUUIDs = new ArrayList<>(4);

    // The step's implementation of setupGeneratedFields()
    // should set these up. Their UUIDs would've been saved as step options.
    orderedFieldUUIDs.add(getStepOption("FIELD1_UUID"));
    orderedFieldUUIDs.add(getStepOption("FIELD2_UUID"));
    orderedFieldUUIDs.add(getStepOption("FIELD3_UUID"));
    orderedFieldUUIDs.add(getStepOption("FIELD4_UUID"));

    // Sample Data
    String[] field1_data = {"Adventure", "Relaxation", "Culture", "Family"};
    int[] field2_data = {30, 32, 11, 44};
    Date[] field3_data = {new Date(103882823L), new Date(10388283323L),
                          new Date(103883232823L), new Date(102323882823L)};
    Timestamp[] field4_data = {new Timestamp(103882823L), new Timestamp(10388283323L),
                               new Timestamp(103883232823L), new Timestamp(102323882823L)};

    // Generate as many rows as configured in Step Option ROW_COUNT
    int rowCount = 10;//Integer.parseInt(getStepOption("ROW_COUNT"));
    Random random = new Random();

    for (int i = 0 ; i < rowCount ; i++) {
        // Data is emitted in packets.
        // This implementation creates a new packet for every row.
        // Data packets can accumulate rows and emit, say, every 20 rows.
        ETLStepResult dataPacket = getFreshDataPacket(outFlow);

        Object[] row = new Object[4];
        row[0] = field1_data[random.nextInt(4)];
        row[1] = field2_data[random.nextInt(4)];
        row[2] = field3_data[random.nextInt(4)];
        row[3] = field4_data[random.nextInt(4)];

        // Send the row of data from Default Fields to Output Fields
        beginInternalTransmission(row, orderedFieldUUIDs);

        // Accumulate transmitted data in a data packet
        endInternalTransmission(dataPacket);

        // Emit the packet of data to the next step.
        // This may be done less frequently, after accumulating rows
        emitData(dataPacket);
    }
}
```

- processEndRows()wireUnion-Allproc

```
@Override
protected void processEndRows() throws ETLException, InterruptedException {
    // Get input flows which feed data to this step
    Set<String> inputFlowUuids = this.getInputFlowUuids();

    // Get the output flow as there can be only one
    String outFlowUuid = getFirstOutputFlow();

    // Get a data packet
    ETLStepResult dataPacket = getFreshDataPacket(outFlowUuid);

    for (String inputFlowUuid : inputFlowUuids) {
        // Get data of each input from its cache
        ETLDataCache inputData = getDataCache(inputFlowUuid);

        // Use this to get the Default Metadata Field corresponding to an Input Field.
        // Input Metadata Field is the field in the input step.
        // Cached data will be in the order of input fields.
        Map<String, String> inputToDefaultFieldMap = getInputToDefaultFieldMap();

        // The data will match the Input Metadata Fields
        List<ETLStepMetadataFieldBean> inputFieldList = inputData.getMetadataFields();
        List<String> unionResultFields = new ArrayList<String>();

        for(ETLStepMetadataFieldBean fieldBean : inputFieldList){
            // Get the Default Metadata Field for an Input Metadata Field
            String inputFieldUuid = fieldBean.getEtlStepMetadataFieldUUID();
            String defaultFieldUuid = inputToDefaultFieldMap.get(inputFieldUuid);

            // Get the Generated Default Metadata Field holding the result of the Union.
            // getUnionFieldForDefaultField() is a method defined in the Step.
            // The result of the union operation is sent to new generated fields.
            // It figures out how a Default Field is linked to the generated field.
            String unionResultField = getUnionFieldForDefaultField(defaultFieldUuid);

            // This holds the Union field corresponding to the Input Field.
            // Fields which are excluded from the union will have a null entry.
            unionResultFields.add(unionResultField);
        }

        // Iterate through the cached data
        Iterator<Object[]> it = inputData.iterator();

        while (it.hasNext()) {
            Object[] row = it.next();

            // Transmit data from:
            // input fields -> default -> generated default (union fields) -> output
            // Data excluded from the union will have a "null" field,
            // so nothing will be transmitted.
            this.beginInternalTransmission(row, unionResultFields);
            this.endInternalTransmission(dataPacket);
        }
    }

    // Accumulate all data before emitting to the next step
    emitData(dataPacket);
}
```