SHIPCADDIE	INSTRUCTIONS **ODBC INSTRUCTIONS**	Document: EX-01 Rev: G Issued By: David R.

PURPOSE

These instructions will explain how to create a configuration file which is used by the ShipCaddie-Direct software to communicate via an ODBC connection to your database server. This will allow the shipcaddie application to:

- Pull information describing a shipment from your DB into ShipCaddie so it can be processed and a label generated for the package.
- Push information about a completed shipment record back into your DB
- Process a batch of shipment records from your DB in sequence

OVERVIEW

After installing and configuring ShipCaddie-Direct, the user may pull order information directly from a DB table or view by following these steps:

- Complete the ODBC configuration file.
    - See instructions to do this below under the **Procedure** heading
- Log in to ShipCaddie and navigate to the Add Shipment page
    - This may be done by clicking “Ship Now” or by clicking Shipments, and then clicking the “Add a Shipment” + icon in the top right corner of the Ready to Ship table
- Input the Reference number corresponding to the DB entry reference field and press Enter. (This can also be done with a handheld scanner against an reference/invoice barcode.)
    - The Add Shipment page will be populated with the order information pulled from the your DB
- After you verify the information and printing the label, ShipCaddie-Direct will push an entry into the update table in your DB
    - This entry will contain the Reference number, Tracking Number and other information about the shipment

WHERE TO FIND THINGS

Most items that you need can be found in the ShipCaddie-Direct tray icon menu.

Configuration files:

- Right click on the ShipCaddie-Direct icon
  - Settings
    - Config Files
- This folder contains
  - debug.json (The debug configuration file)
  - odbc.json (the odbc configuration file if you have added it)

ODBC Documentation:

- Right click on the ShipCaddie-Direct icon
  - Settings
    - ODBC
      - Documentation
- This folder contains
  - This instruction file
  - A blank odbc configuration file
  - An example of a completed odbc configuration file with sample data

Log Files:

- Right click on the ShipCaddie-Direct icon
  - Settings
    - Logging
      - Log Files
- This folder contains
  - The log files saved if logging is enabled

Saved Labels:

- Right click on the ShipCaddie-Direct icon
  - Saved Labels
- This folder contains
  - any labels you have printed to “ShipCaddie-Direct to PDF”

SHIPCADDIE	INSTRUCTIONS **ODBC INSTRUCTIONS**	Document: EX-01 Rev: G Issued By: David R.

PROCEDURE

To configure ShipCaddie-Direct to communicate with a Database over an OBDC connection, the user must follow these steps:

- Download and install ShipCaddie-Direct
  - The Windows version can be found at https://app.shipcaddie.com/printersetup
  - ○ OSX and Linux versions are not currently available
- Determine which table to pull from (referred to as the "**Get Table**")
  - The user may provide an existing table or create a new table or view specifically for ODBC
  - Records will never be inserted into this table by ShipCaddie-Direct
  - Records in this table will never be modified by ShipCaddie-Direct
  - ShipCaddie-Direct will interpret every row in this table as a package
  - Multi-package shipments are indicated by multiple rows which use the same Reference number; ShipCaddie-Direct will interpret this as a single shipment with multiple packages
  - Inventory may be included as well; for detailed instructions on how to handle inventory, refer to the **Inventory** section below
- Create a table to push to (referred to as the "**Put Table**")
  - The user will usually create a new table or view to receive shipped order records, but may use an existing table
- Prepare a **Connection String**
  - https://www.connectionstrings.com/ is a useful reference
  - The easiest method is to use System DSN and prepare a connection string referencing the System DSN
    - See https://www.connectionstrings.com/dsn/
- Fill out the **odbc.json** configuration file to create the mapping
  - A blank template for odbc.json can be found by right-clicking the tray icon and clicking Settings > ODBC > Documentation
  - When the odbc.json configuration is complete, right click the tray icon again and click Settings > ODBC > Config
  - This will open the configuration folder. Place the odbc.json in this folder, then once again right click the tray icon and click Settings > Reconnect to load the odbc.json configuration
  - ShipCaddie-Direct will display a notification message if the odbc.json configuration works, or an error notification message if there was a problem
  - See the **Setting up the Configuration** section for details

SETTING UP THE CONFIGURATION

- The **connectionString** field is required, set it to the Connection String prepared above
    - The Connection String will never be transmitted over http/https, and will only be used to connect to the ODBC using the ODBC driver on the user's computer. It will never be stored on the ShipCaddie server or in the ShipCaddie database.
- The **getMap.TableOrViewName** is required, set it to the name of the Get Table prepared above
- The **putMap.TableOrViewName** is required, set it to the name of the Put Table prepared above
- The **batchMap.TableOrViewName** is optional. This table is used for batch processing. See the **Batch Processing** section for details.
- A detailed description of every mappable field for each of the three tables can be found at the end of this document, under the **getTable/putTable/batchTable Detailed Field Description** headings.
- The rest of the fields should be mapped to the column names in the user's database, as appropriate.
    - The field names are descriptive, with data type and limitations (such as String, and the maximum number of characters in the string)
    - Fields which contain the text "_Required" *must* be filled out, or *ShipCaddie-Direct will fail to pull the order information*
    - Fields which do not contain the text "_Required" are optional. If the user wishes to skip a field, the user should set it to an empty string ("")
- Inventory is managed by the **getMap.Package.Inventory** fields.
    - See the **Inventory** section for details

INVENTORY

There are two ways to deal with inventory.

1) Basic Inventory (limit 10 items)
   The getMap.Package.Inventory fields contain ten groups of item fields, differentiated by index (1 → 10)
    - Each group of fields contains a SKU, Quantity, Name, and Description
    - If entering an inventory item, the SKU and Quantity are required at minimum
    - The user is advised to include a Name or Description so that international shipments containing the inventory can fill out customs declaration details

2) By-Row Inventory (no item limit)

   The getMap.Package.Inventory.ByRow element contains a number of fields that do not have an index attached to them (i.e. “SKU_String_50” and “Value_Decimal”). These fields may be used to break up a package into multiple rows, with one row per inventory item in the package

   - When using this method of inventory management, a Package ID is required for each row
   - Inventory rows will join on the same Package ID
   - If no Package ID is provided, ShipCaddie-Direct will assume that all inventory items should be placed in a single package.

The two methods cannot be mixed; if any of the getMap.Package.Inventory.ByRow fields are configured, the By-Row inventory method will be used, and the Basic Inventory fields will be ignored.

ADVANCED CONFIGURATION

FIELD VALUE ALIAS MAPPING

It is sometimes impractical or impossible to change the source data within your database to conform to the structure that ShipCaddie requires. In cases such as these, the configuration has the capability to remap certain values as specified to solve this problem.

For example, in the Database both the Carrier and Service Level may be referenced using a unique ID. In this case, the ID would have to be converted to something like “USPS” and “Priority Mail” before ShipCaddie could build the shipment record properly.

A Field Value Alias Mapping solves this problem. With the proper configuration, whenever ShipCaddie-Direct encounters *X* value in *Y* field, it will transform the value to a specified *Z*.

To do this, take a mapping value in the odbc.json file, such as:

```
…
  “Shipment”: {
       …
       “CarrierNickname_String_50”: “Carrier”
       …
      }
…
```

and instead of setting it to a column name string (“Carrier”), set it to a json object with the following structure:

```
…
  “Shipment”: {
       …
       “CarrierNickname_String_50”: {
         “dbColumnName”: “Carrier”,
         “valueAlias”: {
            “DatabaseValue1”: “ShipCaddieValue1”,
            “DatabaseValue2”: “ShipCaddieValue2”,
             …
            “DatabaseValueN”: “ShipCaddieValueN”
         },
       …
      }
…
```

dbColumnName is the unique column name in the Database. This is used to reference the column in SQL queries for getting/updating.

valueAlias is a json object containing values that should be altered. This object contains A: B pairs, where the key (A) is the raw value that is in the Database, and the value (B) is the what (A) should be changed to so that ShipCaddie can parse it.

Example:

For this example, the Database has three different IDs for Carrier / Service level combinations

- 10 = USPS Priority Mail
- 11 = USPS Express Mail
- 20 = UPS Ground

The column which holds the above value is called “ShippingMethod.”

If the odbc.json contains the following entries:

```
…
  “Shipment”: {
      …
      “CarrierNickname_String_50”: {
        “dbColumnName”: “ShippingMethod”,
        “valueAlias”: {
           “10”: “USPS”,
           “11”: “USPS”,
           “20”: “UPS”
        },
       “CarrierServiceLevelNickname_String_50”: {
         “dbColumnName”: “ShippingMethod”,
         “valueAlias”: {
            “10”: “Priority Mail”,
            “11”: “Express Mail”,
            “20”: “Ground”
         },
      …
     }
…
“putMap”: {
  …
  “CarrierServiceLevelNickname_String_50”: {
    “dbColumnName”: “ShippingMethodRecord”,
    “valueAlias”: {
      “Priority Mail”: “10”,
      “Express Mail”: “11”,
      “Ground”: “20”
    }
  }
}
…
```

The result is that the ShippingMethod IDs are converted from IDs to valid ShipCaddie Carrier/Service Level names and back again. When ShipCaddie-Direct parses the IDs, they are converted to ShipCaddie Carrier/ServiceLevel names, and when ShipCaddie-Direct pushes an update to the putTable, these are converted back to IDs that the database recognizes.

Field Value Mapping - Set Value

There may be a field in ShipCaddie that is required, but that is not present in the database because the value never changes. For example, if there are never international shipments, the Ship To/From Country would always be "USA." It might be redundant to include this in the database, which would mean the fields are not available to map into ShipCaddie.

If the field is always set to the same value, you can indicate this in the odbc.json file by using the following format:

```
…
"Shipment":
  …
  "ShipToAddress": {
    …
    "Country_String_50_Required": {"setValue": "USA"},
    …
  }
  …
```

Batch Processing

The usual workflow of the ODBC feature is a one-by-one verify -> print process, which is great in many circumstances since the user may need to slightly alter the data to make it correct before shipping, or may need to add package dimensions or weight before the shipment can be processed.

However, in other cases it would be too slow. Imagine going through the verify -> print process for 1,500 shipments at once. That is not realistic. For these situations, you can instead organize your ODBC records into batches, and process the batches in bulk.

A few words to the wise before providing step-by-step Batch Processing instructions:

- The batch process is not going to wait for you to verify anything before printing. If your data is incomplete, or if there are typos, ShipCaddie-Direct is going to use default values to get things moving. This can cause undesired results, so make sure the data is good before

starting a batch process.
    - Example: Say you have two Shipping Sites in ShipCaddie. “Site A” and “Site B”. “Site A” is set as your default shipping site. If you don’t provide a ShippingSiteNickname in the getMapping, all of your shipments will default to “Site A.” If you do provide a field, and provide “Site A” or “Site B” values in that field, the appropriate Shipping Site will be selected. If the value in that column is “Sit eB” or some other typo, the shipment will default to “Site A” even though you wanted “Site B.” **Be Careful**!
- Scripts and Views will be your friend in setting up the batch process. You probably don’t want to have to go through thousands of orders one by one to verify all the details and insert them into a table. Using a view instead of a table would allow you to use SQL queries to construct the batch table for you. Alternatively, you can use a script to pull the data, validate it and organize it, and output a table with Batch IDs and Record Numbers to use in the batch process.
- Try testing in Certify Mode (Test Mode) before you start printing live labels. That will let you work out any issues before letting it loose on live data.

ODBC Batch Processing Instructions:

- Before initiating the process, you need to add a batchMap to your odbc.json configuration. This configuration can be found by from the system tray menu, Settings > Config Files.
- The batchMap will require a table or view name, and a mapping for Batch Id and Shipment Reference IDs.
- The table is constructed to have Batch ID + Reference ID pairs that define one or more batches. For example, if you have Batch Ids “A” and “B”, and Order IDs 1, 2, 3, 4, 5, 6, 7, 8, your batchTable might look something like:

BatchID	OrderID
A	1
A	2
A	3
B	4
B	5
B	6
B	7
B	8

- Next, navigate to the ODBC batch processing section of ShipCaddie. Click the Import icon on the left side of the screen (from any page), and then click the ODBC Batch Print button.

- ShipCaddie will first try to detect an ODBC batch print configuration, which just means it will check that you have ShipCaddie-Direct installed, running, and that you have a valid odbc.json file that contains a batchMap configuration.
- After that, it will grab a list of BatchIDs from your batch table. These are put into a dropdown so you can select your desired batch.
- Select your desired batch from the dropdown, and click Run.
- ShipCaddie will now pull one record from the batch at a time through ShipCaddie-Direct, printing each label automatically. A batch processing window will pop up in the top right corner of the app, and will show the progress as it churns through the batch.
    - Using the example table above, if you select “A” in the dropdown, orders 1, 2, and 3 will be processed. If you select “B” in the dropdown, orders 4, 5, 6, 7, and 8 will be processed.
- Any errors that occur will be listed in the batch processing window, and shipments which error out will appear in the Ready to Ship table (along with the error message that prevented them from printing).

1 : N Put Map Export

When inserting an update record into your database, it is sometimes necessary to take a single field from ShipCaddie and insert it into two fields in the DB. This may be because, for example, two different bits of software need the same data, but from different column names.

You can accomplish this by putting an *array of strings* in the column name, rather than a single string.

Example:

```
“putMap”:{
  …
  “ShipmentReferenceNumber_String_50_Required_Key” : [“RefSpot1”, “RefSpot2”],
  “ShipmentDateShipped_DateTime_YYYY-MM-DDTHH:MM:ss” : [“Date1”, “Date2”],
  “CarrierNickname_String_50”: {
    “dbColumnName”: [“CarrierSpot1”, “CarrierSpot2”, “CarrierSpot3”],
    “valueAlias”: { …  }
  }
  …
 }
```

Note from the example above, this works even if you’re simultaneously doing a Field Value Alias.

This will only work in the put Map; it will break columns in the get or batch

Maps, because while it does make sense to push info from one column to two (or more), it does not make sense to data pull from multiple columns into one.

TROUBLESHOOTING

If a user is having trouble with the ODBC integration, these steps should help identify the problem. In some cases the user may be able to correct the problem, otherwise this section will include instructions on how to get the necessary debug information to ShipCaddie support.

BEFORE ANY OTHER STEP

Try to update ShipCaddie-Direct and see if the problem persists.

- For Windows users, ShipCaddie-Direct is a "click-once" application, and it will attempt to update itself on start-up
- All the user must do to update is right click the tray icon, click "Close," and restart the app

When the OSX and Linux versions are released, they will both have the same automatic update behavior.

CONNECTION

When the app starts, it will attempt to set up a print server to communicate with the user's printer, and, if an odbc.json configuration is present, it will attempt to connect to the database using the connection string. In both cases, a tray notification should pop up indicating success or failure.

The tray notification described should appear when the app starts, and also when the user manually restarts the server by right clicking the tray icon, and then clicking Settings > Reconnect

If the tray notification does not appear (there is neither a success nor a failure message at all)

- This may be a bug in the tray and not an indication of a failure to connect to the database
- Attempt to pull an order from the ODBC connection by following the instructions above under the *Overview* heading
- If the order cannot be pulled, try troubleshooting the connection string as instructed below (*If the tray notification indicates that the ODBC connection failed*), and if that doesn't fix the problem, follow the

instructions for generating and submitting debug/error logs (*Enabling the Debug Log* and *Submitting a Support Ticket with Debug and Error Logs*)

If the tray notification indicates that the ODBC connection failed

- Make sure the connection string is correct
    - Try using an external application to connect with that string, to verify whether it works or not
    - Update the odbc.json with the working connection string
- Right click the Tray, select Settings > Reconnect
- If the new connection string works, a success message should pop up in a tray notification

Pulling Orders from the Database via ODBC

If the ODBC is connected, but it does not pull an order as expected

- Double check the mapping
    - If there are any typos, or any of the fields entered in the odbc.json do not correspond to fields in the Get Table, the query ShipCaddie-Direct makes to find the order will fail
- If there don't appear to be any typos or names that do not match column names in the Get Table, follow the instructions for generating and submitting debug/error logs (*Enabling the Debug Log* and *Submitting a Support Ticket with Debug and Error Logs*)

Pushing Update Records to the Database via ODBC

If the ODBC is connected, and it pulls an order, but it does not push the update record as expected

- Double check the mapping
    - If there are any typos, or any of the fields entered in the odbc.json do not correspond to fields in the PutTable, the query ShipCaddie-Direct makes to insert the record will fail
- Check the error log
    - Any errors that occur during the insert query will be logged in the error.log file, found in User Folder > ShipCaddie-Direct > logs
- Follow the instructions for generating and submitting debug/error logs (*Enabling the Debug Log* and *Submitting a Support Ticket with Debug and Error Logs*)

Enabling the Debug Log

Sometimes the troubleshooting steps may not be enough to resolve an error. In these cases, it may be useful to enable Debug Logging

- Right-click the tray icon
- Clear any existing logs (optional)
    - Settings > Logging > View Logs
    - Delete any logs you wish to clear
- Click Settings > Logging > Enable
- Repeat the action that triggered the problem
    - The Debug Log will have details leading up to and following the error
- Disable the Debug Log (optional)
    - The Debug Log can grow quite large if left on permanently
    - To disable it, click Settings > Logging > Disable
    - To delete the log, click Settings > View Logs, and delete the desired file(s)

Submitting a Support Ticket with Debug and Error Logs

If the user cannot resolve the error using the troubleshooting steps detailed above, it is necessary to submit a bug ticket so that ShipCaddie Support can resolve the issue.

- E-mail support@shipcaddie.com
    - Include a description of the error
    - Include as many details about the database configuration as possible -- what kind of database, what kind of server or network location, what driver, what os, etc.
        - This will be helpful so that ShipCaddie Support can try to reproduce the problem
    - Include the debug.log and error.log files
        - Right click the tray icon
        - Click Troubleshooting > Logs
        - Attach error.log (if it exists) and debug.log (if it exists) to the e-mail
            - If neither log exists, follow the steps in *Enabling the Debug Log* and repeat the action that triggered the error
            - If there is still no log file, *let ShipCaddie Support know* because knowing that the log did not capture

anything may help them figure out what is happening

GETTABLE DETAILED FIELD DESCRIPTIONS

This section gives detailed descriptions of every field that can be mapped in the getTable configuration.

- **TableOrViewName**: The table or view in the user's database that will be used when pulling a shipment record. String.
- **Shipment**: A group of fields related to ship from/to addresses and carrier/service level configurations.
    - **ReferenceNumber_String_50_Required_Key**: This is the key used to find the shipment record. All shipments must have a unique value in this field. String. Max Length 50.
    - **ShippingSiteNickname_String_50_Required**: The physical address from which the shipment originates. These can be configured in ShipCaddie. The value in this field should match the value of a Shipping Site in ShipCaddie. String. Max Length 50.
    - **CarrierNickname_String_50**: When a carrier is registered in ShipCaddie, the user selects a Nickname to reference the carrier. This is that Nickname. String. Max Length 50.
    - **CarrierServiceLevelNickname_String_50**: This is the service level to be used, and it should be a valid service level for the carrier selected above. This field should match what shows in ShipCaddie on the AddShipment screen exactly. String. Max Length 50.
    - **BlindShipFromAddress**: A group of fields related to BlindShip Address details. When Provided, a Blind Ship Address is used as the Return Address on the label, instead of the actual Shipping Site Address.
    - **ShipToAddress:** A group of fields related to the ShipToAddress details. A ship to address is required. The shipment record will not be pulled if the **_Required** fields are left blank.
    - All Address Groups use These Fields (Exceptions noted):
        - **AttentionOf_String_50**: The recipient's name. String. Max Length 50. Blind Ship Address Only.
        - **AttentionOf_String_50_Required**: The recipient's name. String. Max Length 50. ShipToAddress Only.
        - **Company_String_50**: The name of the company. At least one of AttentionOf and Company should be provided. String. Max Length 50.
        - **AddressLine1_String_100**: The first Address line. If there

is only one line, please put it here. String. Max Length 100.

- **AddressLine2_String_100**: The second Address line. Use if necessary. String. Max Length 100.
- **City_String_50**: The return address city. Blind Ship Only. String. Max Length 50.
- **City_String_50_Required:** The destination city. ShipToAddress Only. String. Max Length 50.
- **Province_String_50**: The state or province. May write the name, but the abbreviation is preferred. Not required for international shipments. String. Max Length 50.
- **PostalCode_String_25**: The postal code. Not required for international shipments. String. Max Length 25.
- **Country_String_50**: The return address country. Blind Ship Only. ISO Standard Country Abbreviation is preferred, but if you provide a country name ShipCaddie-Direct will attempt to convert the name to the code. String. Max Length 25.
- **Country_String_50_Required**: The destination country. ShipToAddress only. ISO Standard Country Abbreviation is preferred, but if you provide a country name ShipCaddie-Direct will attempt to convert the name to the code. String. Max Length 25.
- **Phone_String_30**: Phone number goes here. It is optional for DHL, USPS, or UPS, but is required for FedEx. String. Max Length 30.
- **Email_String_128**: E-mail address goes here. String. Max Length 128.
- **IsResidential_Boolean**: Indicates whether the address is residential. It is assumed to be commercial if this is false. Carriers will often have their own back-end checks for residential addresses, which would override the setting here. Boolean (*true* or *false*).

- **Accessorials**: A group of fields with (shipment level) accessory options. Some options come with additional surcharges (such as CoD options), and some options are simply extra requirements for special circumstances (such as Apo/Fpo/Dpo shipments). Not all options are supported by all carriers and service levels. Any values entered into accessorial fields which are not used by the selected carrier and service level are ignored. You can look at the ShipCaddie AddShipment screen to see what Accessorials are supported for what carrier/service level. *Not all Accessorial options in ShipCaddie are supported by ShipCaddie-Direct's ODBC*

function.

- **ApoFpoDpo**: A group of fields for extra details needed to ship to APO, FPO, and DPO addresses.
  - **ContentType_String_50**: The type of content in the package. The carrier will have specific codes to put in here. String. Max Length 50.
  - **LicenseNumber_String_50**: In some cases, a license number is required for customs. The carrier will have details for how/when to fill this out. String. Max Length 50.
  - **CertificateNumber_String_50**: In some cases, a certificate number is required for customs. The carrier will have details for how/when to fill this out. String. Max Length 50.
  - **InvoiceNumber_String_50**: In some cases, an invoice number is required for customs. The carrier will have details for how/when to fill this out. String. Max Length 50.
  - **InternalTransactionNumberITN_String_50**: In some cases, an ITN is required for customs. The carrier will have details for how/when to fill this out. String. Max Length 50.
- **CoD**: A group of fields for CoD options.
  - **Amount_Decimal**: The CoD Amount. Decimal. Assumed to be in USD.
  - **PaymentType_String_50**: The type of acceptable payment, such as “Cash” or “Guaranteed Funds.” The carrier will specify which options are available. String. Max Length 50.
- **InternationalSetup**: A group of fields for extra details needed to ship to international addresses.
  - **ContentType_String_25**: The type of content in the package. The carrier will have instructions on when this is necessary and what specific codes to put in here. String. Max Length 50.
  - **ContentDescription_String_50**: A description of the contents of the package. The carrier will have instructions on when this is necessary and what specific codes to put in here. String. Max Length 50.
  - **ContentNonDeliveryOption_String_25**: Instructions on how to handle undeliverable packages. The

carrier will have instructions on when this is necessary and what specific codes to put in here. String. Max Length 50.

- **InternalTransactionNumberITN_String_50**: In some cases, an ITN is required for customs. The carrier will have details for how/when to fill this out. String. Max Length 50.

- **BillingOptions**: A group of fields for extra billing details usually used for third-party billing.
  - **ShippingPaidBy_String_50**: A code indicating who will pay for the postage (a third party, the shipper, the recipient, etc). The carrier will have instructions on when this is necessary and what specific codes to put in here. String. Max Length 50.
  - **ShippingPaidByAccountNumber_String_50**: The payer's account number. String. Max Length 50.
  - **ShippingPaidByPostalCode_String_25**: The payer's postal code. String. Max Length 25.
  - **ShippingPaidByCountry_String_50**: The payer's country. String. Max Length 50.
  - **DutiesAndTaxesPaidBy_String_50**: A code indicating who will pay for the duties and taxes(a third party, the shipper, the recipient, etc). The carrier will have instructions on when this is necessary and what specific codes to put in here. String. Max Length 50.

- **Package**: A group of fields related to the weight, dimensions, and contents of packages.
  - **PackagingName_String_30**: ShipCaddie has a number of default packaging options that vary by carrier and service level. Additionally, users can create custom packaging, which records length/width/height and maximum allowed weight. The name of the packaging can be put here instead of filling out the DimensionX/Y/Z fields below, and they will be filled in automatically. This is the recommended approach to packaging when many of the packages have identical dimensions. String. Max Length 30.
  - **PackageId_String_50**: An identifier to separate package groups when using the ByRow inventory method. String. Max Length 50.
  - **DimensionX_Decimal_Length_Largest**: The package length goes here, and the length should be the largest dimension. Decimal.

- **DimensionY_Decimal_Width**: The package width goes here. Decimal.
- **DimensionZ_Decimal_Height_Smallest** The package height goes here, and the hight should be the smallest dimension. Decimal.
- **Weight_Decimal_Lbs**: The weight of the package in lbs. Decimal.
- **CustomReference1_String_50**, **CustomReference2_String_50**: These two fields are used for custom references. When possible, the references are printed on the label. String. Max Length 50.
- **Notes_String_250**: A place for shipment/order notes. These notes will be printed in places like the Packing List. String. Max Length 250.
- **Accessorials**: A group of fields with (package level) accessory options. Some options come with additional surcharges and some options are simply extra requirements for special circumstances. Not all options are supported by all carriers and service levels. Any values entered into accessorial fields which are not used by the selected carrier and service level are ignored. You can look at the ShipCaddie AddShipment screen to see what Accessorials are supported for what carrier/service level. *Not all Accessorial options in ShipCaddie are supported by ShipCaddie-Direct's ODBC function.*
  - **InsuranceValue_Decimal**: The amount of insurance to purchase for the package. Decimal.
- **Inventory**: A group of fields to handle inventory. The two methods of inventory management are described above under the heading **Inventory.** The basic inventory will have an **N** in the field name that should be replaced with a number between 1 and 10.
  - **ByRow**: A group of fields used for by-row inventory management.
    - **SKU_String_50_Required**: The SKU of the inventory item. This is required. String. Max Length 50.
    - **Qty_Integer_Required**: The quantity of the above SKU to include in the package. Integer.
    - **Name_String_50**: The name of the inventory item. String. Max Length 50.
    - **Description_String_250**: A description of the inventory item. String. Max Length 50.
    - **Weight_Decimal_Lbs_Required_for_Intl**: The weight of the inventory item. This is required to fill out International customs forms. Decimal, Lbs.
    - **DimensionX_Decimal_Length_Largest_Inches**: The

length of the item. Decimal, inches.

- **DimensionY_Decimal_Width_Inches**: The width of the item. Decimal, inches.
- **DimensionZ_Decimal_Height_Smallest_Inches**: The height of the item. Decimal, inches.
- **InsuranceValue_Decimal_Dollars**: The insurance value of the item. Decimal, USD.
- **OriginCountry_String_25**: The origin country code. String, 25.
- **CustomsDescription_String_250_Required_for_Intl** : The description of the item to be used in customs forms for international shipments. String. Max Length 250.
- **HarmonziedCode_String_25**: The harmonized code, to be used for international shipments. String. Max Length 25.
- **DeclaredValue_Decimal_Dollars**: The declared value of the item to put on customs forms. Decimal, USD.

- **ItemNSKU_String_50**: Basic. The SKU of the inventory item. This is required. String. Max Length 50.
- **ItemNQty_Integer**: Basic. The quantity of the above SKU to include in the package. Integer.
- **ItemNName_String_50**: Basic. The name of the inventory item. String. Max Length 50.
- **ItemNDescription_String_250**: Basic. A description of the inventory item. String. Max Length 50.
- **ItemNWeight_Decimal_Lbs**: Basic. The weight of the inventory item. This is required to fill out International customs forms. Decimal, Lbs.

putTable Detailed Field Descriptions

This section gives detailed descriptions of every field that can be mapped in the putTable configuration.

- **TableOrViewName**: The table or view in the user's database that will be used when inserting a shipment update record. String.
- **ShipmentReferenceNumber_String_50_Required_Key**: The same value as the getMap's Shipment.ReferenceNumber_String_50_Required_Key. This is required. String. Max Length 50.
- **ShipmentDateShipped_DateTime_YYYY-MM-DDTHH:MM:ss** : This is the timestamp of when the shipment label was printed. The format is part of

the field name here; YYYY-MM-DDTHH:MM:ss will produce something that looks like “2015-01-01TY12:00:00”. DateTime.

- **PackageTrackingNumber_String_Required_Key**: This is the tracking number for the package. The ShipmentReferenceNumber + Tracking Number form the key for the putTable, so this is a required field. String.
- **PackageFreightCharge_Decimal**: The total freight charge for the packge. Decimal, USD.
- **PackageWeight_Decimal_Lbs**: The weight of the package. Decimal, lbs.
- **PackageInventory_Skus_String_CSV**: The inventory skus that were on the package, in a CSV string. Example: “SKU1, SKU2, SKU3, SKU4”. String.
- **CarrierNickname_String_50**: The name of the carrier used to ship the package. String. Max Length 50.
- **CarrierServiceLevelNickname_String_50**: The name of the service level used to ship the package. String. Max Length 50.
- **ShipTo_AttentionOf_String_50**: The Recipient’s name. String. Max Length 50.
- **ShipTo_PostalCode_String_25**: The Recipient’s postal code. String. Max Length 25.
- **ShipTo_Province_String_25**: The Recipient’s province. String. Max Length 25.
- **ShipTo_Country_String_25**: The Recipient’s country code. String. Max Length 25.
- **ShipFrom_ShippingSiteNickname_String_50**: The shipping site from which the package was shipped. String. Max Length 50.
- **CustomReference1_String_50**: A short custom note that is printed on the label when possible.
- **CustomReference2_String_50**: A short custom note that is printed on the label when possible.
- **PackageNotes_String_250**: A longer note that is not printed on the label, but which may be printed on the packing list.

batchTable Detailed Field Descriptions

This section gives detailed descriptions of every field that can be mapped in the batchTable configuration.

- **TableOrViewName**: The table or view in the user’s database that will be used to find shipment records during the batch process. String.
- **BatchReferenceId_String_50_Required_Key**: The BatchReferenceId is the name of the batch. A batch contains one or more shipments that should be processed together. This is a required field. String. Max Length

50.

- **Shipment_ReferenceNumber_String_50_Required_Key**: The Shipment Reference Number is the same as the getMap's Shipment.ReferenceNumber_String_50_Required_Key. This is required. String. Max length 50.

See the **Batch Processing** heading above (under **Advanced Configuration**) for details on how the Batch Process works.