Built-in Functions to Responsys Personalization Language (RPL)

Introduction

The document provides guidelines for using Responsys Personalization Language (RPL) to recreate common personalization solutions created using built-in functions. This guide describes the most common usage cases. For a complete RPL reference, see the Responsys Personalization Language (RPL) User Guide and Language Reference.

NOTE: RPL is available only if Content Library and Message Designer for Email are enabled for the account.

About the Data Schema

To demonstrate how to use RPL, we created a sample data model. We assume that all tables in this data model are stored in the *!MasterData* folder.

CONTACTS_LIST

CONTACTS_LIST is the profile list that contains the standard fields and some custom fields. The table below shows only the fields used in this guide.

CUSTOMER_ID_	EMAIL_ADDRESS_	FIRST_NAME	LAST_NAME	AGE
1234	Test1@test.com	John		38
5678	Test2@test.com			14

CONTACTS_LIST is assigned to the campaign as a data source, with the following aliases:

Data Source Alias: CONTACTS_LIST

ALIAS	COLUMN
CUSTOMER_ID_	CUSTOMER_ID_
EMAIL_ADDRESS_	EMAIL_ADDRESS_
FIRST_NAME	FIRST_NAME
LAST_NAME	LAST_NAME

ALIAS	COLUMN
AGE	AGE

FAVORITE_COLORS

FAVORITE_COLORS is a supplemental table that has a one-to-many relationship with the profile list. The CUSTOMER_ID_ field is used as the match key. The table contains data about customers' favorite colors.

CUSTOMER_ID_	COLOR
1234	Green
1234	Blue
1234	Baby Blue
5678	Pink

FAVORITE_COLORS is assigned to the campaign as a data source, with the following aliases:

Data Source Alias: FAVORITE_COLORS

ALIAS	COLUMN
COLOR	COLOR
CUST_ID	CUSTOMER_ID_
FAV_COLOR_MOD_DATE	MODIFIED_DATE_

CART_ABANDONMENT

CART_ABANDONMENT is a supplemental table that has a one-to-many relationship with the profile list. The CUSTOMER_ID_ field is used as the match key. The table contains data about products that customers have abandoned.

CUSTOMER_ ID_	PRODUCT_ ID	PRODUCT_ NAME	PRODUCT_ PRICE	ABANDONED_ DATE
1234	1111	Dress	25.99	2014-01-05 00:00:00.0
5678	2222	Boots	249	2014-01-20 00:00:00.0
1234	3333	Shorts	15.99	2014-01-15 00:00:00.0
5678	4444	Jeans	69.99	2014-01-21 00:00:00.0
5678	5555	Cap	19.99	2014-01-25 00:00:00.0

CART_ABANDONMENT is assigned to the campaign as a data source, with the following aliases:

Data Source Alias: CART_ABANDONMENT

ALIAS	COLUMN
ABN_CUST_ID	CUSTOMER_ID_
ABN_DATE	ABANDONED_DATE
ABN_PRODUCT_ID	PRODUCT_ID
ABN_PRODUCT_NAME	PRODUCT_NAME
ABN_PRODUCT_PRICE	PRODUCT_PRICE

PRODUCTS

PRODUCTS is a supplemental table that has no relationship with the profile list. It contains detailed data for each product and is used as a reference for personalization in the email campaign.

PRODUCT_ID	PRODUCT_NAME	PRODUCT_COLOR	PRODUCT_PRICE	PRODUCT_SALE_PRICE	ON_SALE
1111	Dress	Baby Blue	34.99	25.99	1
2222	Boots	Brown	249		
3333	Shorts	Blue	15.99		
4444	Jeans	Blue	69.99		
5555	Cap	Green	19.99		
6666	Sun Hat	Red	15.99	19.99	1
7777	Blouse	Baby Blue	25	30	1
8888	Tank Top	White	12.99	15.99	1
9999	Tank Top	Green	12.99	9999	
11111	Dress	Green	19.99	25.99	1
22222	Cap	Baby Blue	10.99	16.99	1
33333	Pants	Green	24.99		

PRODUCTS is assigned to the campaign as a data source, with the following aliases:

Data Source Alias: PRODUCTS

ALIAS	COLUMN
ON_SALE	ON_SALE
PRODUCT_COLOR	PRODUCT_COLOR
PRODUCT_ID	PRODUCT_ID
PRODUCT_NAME	PRODUCT_NAME
PRODUCT_PRICE	PRODUCT_PRICE

ALIAS	COLUMN
PRODUCT_SALE_PRICE	PRODUCT_SALE_PRICE

Important Tips

When using RPL, you need to:

Declare all tables as data sources
All tables (i.e. PETs and supplemental tables) that will be used for personalization, regardless of whether they have a direct relationship with the profile list used in the campaign, must be assigned as data sources in Message Designer for Email.

Declare indirect tables as Lookup tables
When declaring a supplemental table as a data source that has an indirect relationship with the profile list, such as the PRODUCTS table in our example, you must select the **Data source is used only as a Lookup table** when adding it as a data source in Message Designer for Email.

Assign all Lookup fields as Lookup keys
After assigning a supplemental table as a data source, select **Lookup Key** for any field that will be used for lookup.

Use single quotes instead of doubles quotes
RPL uses quotes in many of its functions. In some cases, these quotes conflict with HTML tags. As a result, Message Designer for Email might trim off part of the function due to the multiple uses of double quotes. For example:

```
<a href=”${form(campaign.name, {“usedb”, true})}”>
```

Results in:

```
<a href=”${form(campaign.name, {“>
```

As a solution, you should use single quotes in RPL functions:

```
<a href=”${form(campaign.name, {‘usedb’, true})}”>
```

Sort data in desired order using SQL
RPL does not always sort and display data retrieved from a table in the same order as it appears in the table. To control the sort order, you need to build a SQL query on top of the table

and use the query to sort the data in the desired order. Then, assign that SQL to the campaign a data source.

RPL Usage

This section shows examples using RPL for common personalization solutions.

Basics: Setting a variable

The following example sets the value *Winter* for a variable called `SEASON`.

	Syntax	**Example**
BUILT-IN FUNCTION	`$SETVARS()$`	`$SETVARS(SEASON, Winter)$`
RPL	`<#assign variable=”value”>`	`<#assign SEASON=”Winter”>`

Basics: Concatenation

The following example **c**ombines the value of the AGE field from the CONTACTS_LIST with the phrase “ *years old today*”.

	Syntax	**Syntax**
BUILT-IN FUNCTION	`$CONCAT ()$`	`$CONCAT(LOOKUP(AGE), SPACE(), years old today)$`
RPL	`${value1 + value2}`	`${CONTACTS_LIST.AGE + “ years old today”}`

For the user with CUSTOMER_ID_=1234, the result is:

```
38 years old today
```

For the user with CUSTOMER_ID_=5678, the result is:

```
14 years old today
```

Basics: Embedding subdocuments

The following example embeds the subdocument *cms://contentlibrary/common/footer.html* into the main HTML document.

	Syntax	Example
BUILT-IN FUNCTION	`$DOCUMENT()$`	`$DOCUMENT(/contentlibrary/common/footer.html)$`
RPL	`<#include subdocument>`	`<#include “cms://contentlibrary/common/footer.html”>`

Basics: Extracting the email domain

The following example extracts the email domain from the EMAIL_ADDRESS_ field in the CONTACTS_LIST.

	Syntax	Example
BUILT-IN FUNCTION	`$EMAILDOMAIN()$`	`$EMAILDOMAIN(LOOKUP(EMAIL_ADDRESS_))$`
RPL	`${emaildomain()}`	`${emaildomain(CONTACTS_LIST .EMAIL_ADDRESS_)}`

Basics: Replacing all occurrences of a string in an expression

The following example replaces all occurrences of “*palm*” with “*redwood*” in the following expression: *This palm tree is the tallest palm tree in the city*.

	Syntax	**Example**
BUILT-IN FUNCTION	`$REPLACEALL()$`	`$REPLACEALL(This palm tree is the tallest palm tree in the city, palm, redwood)$`
RPL	`${“expression”?replace(“search-string”, “replace-string”)}`	`${“This palm tree is the tallest palm tree in the city”?replace(“palm”,” redwood”)}`

Result:

```
This redwood tree is the tallest redwood tree in the city
```

Basics: Removing grouping commas from integer and number fields

RPL automatically inserts a grouping comma into values of integer and number fields (i.e. the value 1111 is converted to 1,111). Although these fields have numerical values, the values are treated like strings. For example, the PRODUCT_ID field in the PRODUCTS supplemental table was specified as an integer for data storage efficiency, but its value is more like a string. Since PRODUCT_ID is specified as an integer, RPL automatically adds a grouping comma to the value. This might cause issues when you use the value for reference. For this reason, you should remove the grouping commas.

The following example removes the grouping comma from the PRODUCT_ID field in the PRODUCTS supplemental table.

	Syntax	**Example**
BUILT-IN FUNCTION	N/A	N/A
RPL	`${expression?c}`	`${PRODUCTS.PRODUCT_ID?c}`

Expressions: Greater than

The following example checks whether the value of the AGE field in CONTACTS_LIST is greater than 18. If yes, the result shows “1”; otherwise it shows “0”.

	Syntax	**Example**
BUILT-IN FUNCTION	`$GT()$`	`$GT(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE gt 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Greater than or equal to

The following example checks whether the value of the AGE field in CONTACTS_LIST is greater than or equal to 18. If yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$GE()$`	`$GE(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE gte 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Less than

The following example checks whether the value of the AGE field in CONTACTS_LIST is less than 18. If yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$LT()$`	`$LT(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE lt 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Less than or equal to

The following example checks whether the value of the AGE field in CONTACTS_LIST is less than or equal to 18. If yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$LE()$`	`$LE(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE eq 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Equal to

The following example checks whether the value of the AGE field in CONTACTS_LIST is equal to 18. If yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$EQ()$`	`$EQ(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE == 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Not equal to

The following example checks whether the value of the AGE field in CONTACTS_LIST is not equal to 18. If it is not, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$NE()$`	`$NE(LOOKUP(AGE), 18)$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE != 18>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: And

The following example checks whether the value of the AGE field in CONTACTS_LIST is not equal to 18 and the value of FIRST_NAME is null. If both are yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$AND(NE(), NOTHING())$`	`$AND(NE(LOOKUP(AGE), 18), NOTHING(LOOKUP(FIRST_NAME)))$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE != 18 && CONTACTS_LIST.FIRST_NAME=””>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Expressions: Or

The following example checks whether the value of the AGE field in CONTACTS_LIST is not equal to 18 or the value of FIRST_NAME is null. If either one is yes, the result shows “1”; otherwise it shows “0”.

	Syntax	Example
BUILT-IN FUNCTION	`$AND(NE(), NOTHING())$`	`$OR(NE(LOOKUP(AGE), 18), NOTHING(LOOKUP(FIRST_NAME)))$`

RPL	`<#if>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE != 18 || CONTACTS_LIST.FIRST_NAME=””>`
		`1`
		`<#else>`
		`0`
		`</#if>`

Campaign Details: Retrieving the Campaign ID

The following example retrieves the campaign ID.

	Syntax	Example
BUILT-IN FUNCTION	`$CAMPAIGNID ()$`	`$CAMPAIGNID()$`
RPL	`${campaign.id}`	`${campaign.id}`

Campaign Details: Retrieving the Marketing Program

The following example retrieves the campaign Marketing Program.

	Syntax	Example
BUILT-IN FUNCTION	`$CAMPAIGNMARKETINGPROGRAM()$`	`$CAMPAIGNMARKETINGPROGRAM()$`
RPL	`${campaign.marketingprogram}`	`${campaign.marketingprogram}`

Campaign Details: Retrieving the Marketing Strategy

The following example retrieves the campaign Marketing Strategy.

	Syntax	Example
BUILT-IN FUNCTION	`$CAMPAIGNMARKETINGSTRATEGY()$`	`$CAMPAIGNMARKETINGSTRATEGY ()$`
RPL	`${campaign.marketingstrategy}`	`${campaign.marketingstrategy}`

Campaign Details: Retrieving the campaign name

The following example retrieves the campaign name.

	Syntax	Example
BUILT-IN FUNCTION	`$CAMPAIGNNAME()$`	`$CAMPAIGNNAME ()$`
RPL	`${campaign.name}`	`${campaign.name}`

Conditions: Single condition, single rule

If the FIRST_NAME field in CONTACTS_LIST has a value, the following example shows that value; otherwise, it shows nothing.

	Syntax	Example

BUILT-IN FUNCTION	`$COND()$`
	`$LOOKUP()$`
	`$EMPTY()$`
	`$NOTHING()$`
		`$COND(EMPTY(LOOKUP(FIRST_NAME)), NOTHING(), LOOKUP(FIRST_NAME))$`

RPL	`<#if condition>`
	`</#if>`
		`<#if CONTACTS_LIST.FIRST_NAME !="">`
		`${ CONTACTS_LIST.FIRST_NAME }`
		`</#if>`

For the user with CUSTOMER_ID_=1234, the result is:

```
John
```

Conditions: Single condition, multiple rules

If the AGE field in CONTACTS_LIST has a value greater than or equal to 18, the following example shows "*Welcome to our site*." Otherwise, the example shows "*You are not allowed to proceed to the site*."

	Syntax	Example

BUILT-IN FUNCTION	`$COND()$`
	`$LOOKUP()$`
	`$GE()$`
	`$LT()$`
		`$COND(GE(LOOKUP(AGE), 18), Welcome to our site., You are not allowed to proceed to the site.))$`

RPL	`<#if condition>`
	`<#elseif condition>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE gte 18>`
		`Welcome to our site.`
		`<#else>`
		`You are not allowed to proceed to the site.`
		`</#if>`

For the user with CUSTOMER_ID_=1234, the result is:

```
Welcome to our site.
```

For the user with CUSTOMER_ID_=5678, the result is:

```
You are not allowed to proceed to the site.
```

Conditions: Multiple conditions, multiple rules

If the value of the AGE field in CONTACTS_LIST is greater than or equal to 18, the following example shows “*Welcome to our site.*”

If the value of AGE field in the CONTACTS_LIST is less than 18, the example shows “*You are not allowed to proceed to the site*.”

If the AGE field in CONTACTS_LIST has no value, the examples shows “*What is your age?*”

	Syntax	Example

BUILT-IN FUNCTION	`$COND()$`
	`$LOOKUP()$`
	`$GE()$`
	`$LT()$`
		`$COND(GE(LOOKUP(AGE), 18), Welcome to our site., COND(LT(LOOKUP(AGE), 18), You are not allowed to proceed to the site., What is your age?))$`

RPL	`<#if condition>`
	`<#elseif condition>`
	`<#else>`
	`</#if>`
		`<#if CONTACTS_LIST.AGE gte 18>`
		`Welcome to our site.`
		`<#elseif CONTACTS_LIST.AGE lt 18>`
		`You are not allowed to proceed to the site.`
		`<#else>`
		`What is your age?`
		`</#if>`

For the user with CUSTOMER_ID_=1234, the result is:

```
Welcome to our site.
```

For the user with CUSTOMER_ID_=5678, the result is:

```
You are not allowed to proceed to the site.
```

For users where the AGE field is null, the result is:

```
What is your age?
```

Forms: Linking to an Oracle Responsys form

The following example creates a link to an Oracle Responsys form called *Preference_Center.*

	Syntax	Example
BUILT-IN FUNCTION	`$FORMLINK()$`	`$FORMLINK(Preference_Center)$`
RPL	`${form()}`	`${form('Preference_Center')}`

Forms: Passing a field value in a form link

The following example creates a link to an Oracle Responsys form called *Preference_Center* and passes the value of the AGE field in CONTACTS_LIST.

	Syntax	Example
BUILT-IN FUNCTION	`$FORMLINK()$`	`$FORMLINK(Preference_Center, AGE)$`
RPL	`${form()}`	`${form('Preference_Center', {}, 'CONTACTS_LIST.AGE')}`

Forms: Assigning and passing a parameter/value in a form link

The following example creates a link to an Oracle Responsys form called *Preference_Center* and assigns the value *Summer_Sweepstakes* to the CONTEST parameter.

	Syntax	Example
BUILT-IN FUNCTION	`$FORMLINK()$`	`$FORMLINK(Preference_Center, CONTEST=Summer_Sweepstakes)$`
RPL	`${form()}`	`${form('Preference_Center', {}, 'tablename.CONTEST=Summer_Sweepstakes')}`

Forms: Creating a View as Webpage form link

The following example creates a View as Webpage link that opens the email as a web page and passes the value of the FIRST_NAME field in CONTACTS_LIST.

	Syntax	Example
BUILT-IN FUNCTION	`$FORMLINK()$`	`$FORMLINK(CAMPAIGNNAME())$`
RPL	`${form()}`	`${form(campaign.name, { }, 'CONTACTS_LIST.FIRST_NAME')}`

Links: Referencing a link in a link table

The following example references a link called *Header_Logo* in the link table.

	Syntax	Example
BUILT-IN FUNCTION	`$CLICKTHROUGH()$`	`$CLICKTHROUGH(Header_Logo)$`
RPL	`${clickthrough()}`	`${clickthrough('Header_Logo')}`

Links: Passing a field value when referencing a link in a link table

The following example references a link called *Header_Logo* in the link table and passes the value of the AGE field in CONTACTS_LIST.

	Syntax	Example
BUILT-IN FUNCTION	`$CLICKTHROUGH()$`	`$CLICKTHROUGH(Header_Logo, AGE)$`
RPL	`${clickthrough()}`	`${clickthrough(‘Header_Logo’, ‘CONTACTS_LIST.AGE’)}`

Links: Assigning and passing a parameter/value when referencing a link in a link table

The following example references a link called *Header_Logo* in the link table and assigns the value *Summer_Sweepstakes* to the CONTEST parameter.

	Syntax	Example
BUILT-IN FUNCTION	`$CLICKTHROUGH()$`	`$CLICKTHROUGH(Header_Logo, CONTEST=Summer_Sweepstakes)$`
RPL	`${clickthrough()}`	`${clickthrough(‘Header_Logo’, ‘CONTEST=Summer_Sweepstakes’)}`

Calculations: Addition

The following example adds 2 to the value of the AGE field in CONTACTS_LIST.

	Syntax	Example
BUILT-IN FUNCTION	`$ADD()$`	`$ADD(AGE, 2)$`
RPL	`${value1 + value2}`	`${CONTACTS_LIST.AGE + 2}`

For the user with CUSTOMER_ID_=1234, the result is:

```
40
```

For the user with CUSTOMER_ID_=5678, the result is:

```
16
```

Calculations: Subtraction

The following example subtracts 2 from the value of the AGE field in CONTACTS_LIST.

	Syntax	**Example**
BUILT-IN FUNCTION	`$SUB()$`	`$SUB(AGE, 2)$`
RPL	`${value1 - value2}`	`${CONTACTS_LIST.AGE - 2}`

For the user with CUSTOMER_ID_=1234, the result is:

```
36
```

For the user with CUSTOMER_ID_=5678, the result is:

```
12
```

Calculations: Multiplication

The following example multiplies the value of the AGE filed in CONTACTS_LIST by 2.

	Syntax	**Example**
BUILT-IN FUNCTION	`$MUL()$`	`$MUL(AGE, 2)$`
RPL	`${value1 * value2}`	`${CONTACTS_LIST.AGE * 2}`

For the user with CUSTOMER_ID_=1234, the result is:

```
76
```

For the user with CUSTOMER_ID_=5678, the result is:

```
28
```

Calculations: Division

The following example divides the value of the AGE field in CONTACTS_LIST by 2.

	Syntax	Example
BUILT-IN FUNCTION	`$DIV()$`	`$MUL(AGE, 2)$`
RPL	`${value1 / value2}`	`${CONTACTS_LIST.AGE / 2}`

For the user with CUSTOMER_ID_=1234, the result is:

```
19
```

For the user with CUSTOMER_ID_=5678, the result is:

```
7
```

Lookups: Looking up all matching records from a supplemental table

The following example looks up all matching values of the COLOR field in the FAVORITE_COLORS supplemental table that have a one-to-many relationship with the CONTACTS_LIST, using the CUSTOMER_ID_ as the match key.

	Syntax	Example
BUILT-IN FUNCTION	`$LOOKUPRECORDS()$`	`$LOOKUPRECORDS(!MasterData, FAVORITE_COLORS, CUSTOMER_ID_, LOOKUP(CUSTOMER_ID_), COLOR)$`

RPL	`<#data>`
	`<#filter>`
	`<#fields>`
	`</#data>`
		`<#data FAVORITE_COLORS as fav_colors>`
		`<#filter CUST_ID=CONTACTS_LIST.CUSTOMER_ID_>`
		`<#fields COLOR>`
		`${fav_colors.COLOR}`
		`</#data>`

For the user with CUSTOMER_ID_=1234, the result is:

```
Baby Blue Blue Green
```

For the user with CUSTOMER_ID_=5678, the result it:

```
Pink
```

NOTE: Notice that the matching COLOR values are displayed in alphanumeric order rather than in the order they appear in the FAVORITE_COLORS supplemental table. If you need the COLOR values to appear in a specific order, create a SQL query that sorts the FAVORITE_COLORS supplemental table data in the desired order and select that SQL as a data source.

Lookups: Looking up all matching records from a supplemental table and limit results returned

The following example looks up matching values of the COLOR field in the FAVORITE_COLORS supplemental table that have a one-to-many relationship with the CONTACTS_LIST and returns a limit of 2 values, using the CUSTOMER_ID_ as the match key.

	Syntax	Example
BUILT-IN FUNCTION	`$LOOKUPRECORDS()$`	`$LOOKUPRECORDS(!MasterData, FAVORITE_COLORS, CUSTOMER_ID_, LOOKUP(CUSTOMER_ID_), COLOR)$`

RPL	`<#data>`
	`<#filter>`
	`<#fields>`
	`</#data>`
		`<#data FAVORITE_COLORS as fav_colors limit=2>`
		`<#filter CUST_ID=CONTACTS_LIST.CUSTOMER_ID_>`
		`<#fields COLOR>`
		`${fav_colors.COLOR}`
		`</#data>`

For the user with CUSTOMER_ID_=1234, the result is:

```
Baby Blue Blue
```

For the user with CUSTOMER_ID_=5678, the result it:

```
Pink
```

NOTE: Notice that the matching COLOR values are displayed in alphanumeric order rather than in the order they appear in the FAVORITE_COLORS supplemental table. If you need the COLOR values to appear in a specific order, create a SQL query that sorts the FAVORITE_COLORS supplemental table data in the desired order and select that SQL as a data source.

Lookups: Looking up all matching records from a supplemental table and retrieving additional data from another supplemental table

The following example looks up all matching values of the PRODUCT_ID field in the CART_ABANDONMENT supplemental table that have a one-to-many relationship with the CONTACTS_LIST, using the CUSTOMER_ID_ as the match key. The example then retrieves the PRODUCT_NAME from the PRODUCTS table using PRODUCT_ID as the match key.

	Syntax	Example

BUILT-IN FUNCTION	`$FOREACH()$`
	`$LOOKUPRECORDS()$`
	`$LOOKUPTABLE()$`
		`Main Document`
		`$FOREACH(CA_Loop, PAIRSLIST(1, PRODUCT_ID, LOOKUPRECORDS(!MasterData, CART_ABANDONMENT, CUSTOMER_ID_, LOOKUP(CUSTOMER_ID_), PRODUCT_ID)), Cart_Abandon, Cart_Abandon_Subdoc)$`
		
		`Cart_Abandon_Subdoc.htm Subdocument`
		`$SETVARS(LOOKUP(CA_Loop))$`
		`$LOOKUPTABLE(!MasterData, PRODUCTS, PRODUCT_ID, LOOKUP(PRODUCT_ID), PRODUCT_NAME)$`

RPL	`<#data>`
	`<#filter>`
	`<#fields>`
	`</#data>`
		`<#data CART_ABANDONMENT as cart_abandon>`
		`<#filter ABN_CUST_ID=CONTACTS_LIST.CUSTOMER_ID_>`
		`<#fields ABN_PRODUCT_ID>`
		`<#data PRODUCTS as products>`
		`<#filter PRODUCT_ID= cart_abandon.ABN_PRODUCT_ID>`
		`<#fields PRODUCT_NAME>`
		`${products.PRODUCT_NAME}<br>`
		`</data>`
		`</#data>`

For the user with CUSTOMER_ID_=1234, the result is:

```
Dress
Shorts
```

For the user with CUSTOMER_ID_=5678, the result is:

```
Boots
Jeans
Cap
```

NOTE: Notice that the matching values are in alphanumeric order by PRODUCT_ID, which was the original lookup source, rather than by PRODUCT_NAME. If you need the PRODUCT_NAME values to appear in a specific order, create a SQL query that joins the CART_ABANDONMENT and PRODUCTS supplemental tables and pre-sorts the data in the desired order. Then, select that SQL as a data source.