

String Patterns in CA DevTest

Daniel Mazzei, Engineering Services Architect

February 2017

© 2016 CA. All rights reserved. CA confidential & proprietary information. For CA, CA Partner and CA Customer use only. No unauthorized use, copying or distribution. All names of individuals or of companies referenced herein are fictitious names used for instructional purposes only. Any similarity to any real persons or businesses is purely coincidental. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies. These Materials are for your informational purposes only, and do not form any type of warranty. The use of any software or product referenced in the Materials is governed by the end user's applicable license agreement. CA is the manufacturer of these Materials. Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-19(c)(1)-(2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

What are String Patterns?

- String patterns are special types of property expressions
 - Use the syntax: `{{=[:patternname:]}}`
 - Where “{{” denotes a property, “=[” signals a string pattern, and “:” encloses the pattern name
- There are many different String Patterns available out-of-the-box
 - First and last names and middle initials
 - Street addresses, states, countries, and zip codes
 - Telephone numbers and email addresses
 - Social Security numbers, credit card numbers, and expiration dates
- You can create your own String Patterns!
- Of course there are also property expressions and regular properties

Where is the test data stored?

This data comes from the TESTDATA table in the DEVTEST database

The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'SQLQuery1.sql - mazda03N83701.master (MAZDA03N83701\Administrator (62)) - Microsoft SQL Server Management Studio (Administrator)'. The Object Explorer on the left shows the database structure, including tables like 'dbo.LISA_FRAME_CATEGORY', 'dbo.LISA_FRAME_DATA', and 'dbo.TESTDATA'. The 'dbo.TESTDATA' table is selected, and its columns are visible: ID, BIRTHDAY, CCEXPRES, CCNUMBER, CCTYPE, CV2, CITY, COUNTRY, EMAILADDRESS, GENDER, GIVENNAME, MIDDLEINITIAL, MOTHERSMAIDEN, SSN, and ZIPCODE. The 'Results' pane shows a query executed successfully, displaying 22 rows of data. The query is a 'SELECT TOP 1000' statement. The 'Properties' pane on the right shows connection details for 'mazda03N83701 (MAZDA03N83701\Administrator (62))'. The status bar at the bottom indicates 'Query executed successfully.' and 'mazda03N83701 (11.0 SP1) MAZDA03N83701\Administrator... master 00:00:00 1000 rows'.

ID	BIRTHDAY	CCEXPRES	CCNUMBER	CCTYPE	CV2	CITY	COUNTRY	EMAILADDRESS	GENDER	GIVENNAME	MIDDLEINITIAL	MOTHERSMAIDEN	SSN
1	1963-11-03 00:00:00.000	3/2013	5227081949027029	MasterCard	330	SHONGALOO	US	Rosie S Stevens@trashymail.com	female	Rosie	S	Dove	434-47-5409
2	1964-02-04 00:00:00.000	11/2013	4716806516757835	Visa	293	San Diego	US	Joni J Huber@pookmail.com	female	Joni	J	McAfee	614-40-1053
3	1973-04-22 00:00:00.000	8/2011	5381678851559758	MasterCard	875	B Segundo	US	Lisette E Williams@mailinator.com	female	Lisette	E	Dewitt	546-71-4973
4	1971-04-28 00:00:00.000	2/2010	533564583658827	MasterCard	773	Arlington Heights	US	Lawrence K Poplar@mailinator.com	male	Lawrence	K	Howard	350-02-1232
5	1959-09-10 00:00:00.000	5/2012	492953478645432	Visa	570	Los Angeles	US	Bernadette S Fritson@mailinator.com	female	Bernadette	S	Anderson	616-51-0344
6	1971-11-28 00:00:00.000	6/2009	4532059646183744	Visa	716	NEWARK	US	Tanya P Peret@trashymail.com	female	Tanya	P	Perry	140-72-2944
7	1958-04-06 00:00:00.000	11/2011	4716733199801526	Visa	055	Maunee	US	Lorraine J Flanagan@trashymail.com	female	Lorraine	J	Nemeth	295-20-0146
8	1969-01-02 00:00:00.000	7/2010	5400986330646501	MasterCard	856	Wauauga	US	Roberta P Watts@dogdgt.com	female	Roberta	P	Stewart	643-54-3377
9	1952-01-19 00:00:00.000	12/2010	530229934374363	MasterCard	684	Washington	US	Jaime J Petersen@mailinator.com	female	Jaime	J	Salinas	578-13-7390
10	1968-01-23 00:00:00.000	1/2011	5541793892786168	MasterCard	522	Portsmouth	US	Christine R Fraser@trashymail.com	female	Christine	R	Whitehouse	001-52-1429
11	1960-07-13 00:00:00.000	6/2010	4485383854173670	Visa	982	NOBLE	US	Emily W Tarrant@mailinator.com	female	Emily	W	Donovan	357-46-1417
12	1966-10-07 00:00:00.000	3/2013	4929705541689695	Visa	754	Orcutt	US	Levi H Starley@pambob.com	male	Levi	H	Hughes	257-42-5949
13	1965-10-05 00:00:00.000	11/2012	5397825486111503	MasterCard	114	Baltimore	US	Roselyn A Hanson@pookmail.com	female	Roselyn	A	Ponce	213-39-0889
14	1948-03-09 00:00:00.000	7/2009	5479252930945408	MasterCard	615	ESPAÑOLA	US	Willie S Redmond@mailinator.com	female	Willie	S	Stogner	649-05-9980
15	1973-01-18 00:00:00.000	10/2013	5573472351506461	MasterCard	222	Philadelphia	US	Shelly R Orth@mailinator.com	female	Shelly	R	Loth	174-76-1853
16	1948-08-05 00:00:00.000	7/2013	4929868376231589	Visa	763	OMAHA	US	Barbara S Singleton@pambob.com	female	Barbara	S	Cox	505-86-4602
17	1980-10-30 00:00:00.000	2/2010	4916567392796274	Visa	830	NEWARK	US	Charles J Mitchell@pambob.com	male	Charles	J	Gil	139-80-7550
18	1976-07-02 00:00:00.000	2/2010	5195201275951893	MasterCard	542	Newalla	US	John A Babineaux@dogdgt.com	male	John	A	Bentéz	446-15-5222
19	1964-03-27 00:00:00.000	11/2012	4556841767513309	Visa	552	San Jose	US	Jean B Leclerc@dogdgt.com	female	Jean	B	Felder	615-51-8177
20	1969-01-24 00:00:00.000	7/2013	5358582131651889	MasterCard	766	Hemdon	US	Alvin K Shaffer@trashymail.com	male	Alvin	K	Butler	697-03-8809
21	1961-11-03 00:00:00.000	2/2009	4929937456430514	Visa	899	South Burlington	US	Alan K Darrow@pambob.com	male	Alan	K	Stom	008-88-7447
22	1970-06-24 00:00:00.000	10/2013	556378193789997	MasterCard	453	RUTLAND	US	Joseph S Clinton@pambob.com	male	Joseph	S	Bishop	327-22-9956

How do I use pre-define String Patterns?

- To insert a first name, the string pattern property would be

```
{{=:First Name:}}}
```

- The property would evaluate to a first name from the test data DB and look like a real name

```
Andrea
```

- This behavior is much better than the possibility of dealing with random strings that do not look like a real name!

How do I create my own String Patterns?

- Use the Custom String Generator!
- Or simply follow the syntax
- Example telephone number pattern
 - `{{=[\](DDD\)DDD\-DDDD)}}}`
 - or
 - `{{=[(DDD)DDD-DDDD)}}}`

These patterns generate strings based on the following definitions:

Pattern-based

D - digit (0-9)

H - hex digit (0-9, A-F)

h - hex digit (0-9, a-f)

X - hex digit (0-9, A-F, a-f)

L - letters (A-Z)

l - letters (a-z)

A - alphanumeric

P - punctuation

. - (dot) anything printable

[1,2,3] - randomly choose among the options shown

{0,9,8} - do not allow these on the previous spec

\ - take the following char as a literal

*(N[-M]) - repeat the pattern N times, or randomly N-M times if M is present

Anything that is not a pattern char IS a literal, for example Jo\hnDDD would be "John123" or the like.

Example String Patterns

The screenshot displays the DevTest Solutions v9.5.0 (9.5.0.954) interface. The main window shows the 'Output Log Message' tab, which contains a list of string patterns. A red box highlights the following patterns:

```
(([:First Name:])) (([:Middle Initial:])) (([:Last Name:]))  
(([:Street Address:]))  
(([:City:])) (([:State Code:])) (([:Zip Code:]))  
(([:Country:]))  
SSN: (([:SSN: DDD-DD-DDDD]))  
Card: (([:Credit Card:])) Expires (([:CC Expiry:]))  
Home Phone: (([:Telephone:]))  
Mobile Phone: (([:DDD\DDD-DDDD]))  
Mobile Phone (literal): (([:DDD-DDD-DDDD]))  
Phone 2 Custom String Gen: (([:Phone2:DDD-DDD-DDDD]))  
Email: (([:Email:]))  
Email 2: (([:L\.([:Last Name:]]@company\.com]))  
Fake Email with (ALID): (([:ALID@company\.com]))  
Login: (([:First Name:]))(([:Last Name:]))  
Password (LIDPAAA): (([:LIDPAAAA]))  
Random number not odd: (([:D(1,3,5,7,9)]))  
Randomly choose from selection: (([:[A,B,C]]))
```

The right-hand pane shows the 'Patterns' section, which includes a table of built-in string generator patterns:

Name	Pattern	Category
BirthDay		TestData
CC Expiry		TestData
CC Verification Code		TestData
City		TestData
Country		TestData
Credit Card		TestData
Email		TestData
First Name		TestData
Last Name		TestData
Middle Initial		TestData
SSN		TestData
State Code		TestData
Street Address		TestData
Telephone		TestData
UPS Tracking Code		TestData
Zip Code		TestData

Below the table, there is a 'Find:' search bar and a 'Custom String Generator Patterns' section with a table for custom patterns.

Example String Pattern Response

The screenshot displays the DevTest Solutions v9.5.0 (9.5.0.954) interface. The main window shows the 'Output Log Message' tab, which contains a list of test results. A red box highlights the 'Response' tab, which displays a string pattern response for a test run. The response is a JSON object containing various fields and their values, including personal information, contact details, and a password.

Response

```
{
  "Name": "Kenneth T Mayer",
  "Address": "2700 Harley Brook Lane",
  "City": "Franklin, PA 16323",
  "Country": "US",
  "SSN": "198-03-0490",
  "Card": "5126-7891-6194-1801 Expires 5/2013",
  "Home Phone": "814-518-9418",
  "Mobile Phone": "(707)192-3131",
  "Mobile Phone (literal)": "(234)258-7722",
  "Phone 2 Custom String Gen": "267-134-0694",
  "Email": "Kenneth.T.Mayer@epamob.com",
  "Email 2": "P.atterson@company.com",
  "Fake Email with (ALID)": "1Ht5@company.com",
  "Login": "Kenneth.Mayer",
  "Password (L1DFAAAA)": "Di6\\mz6C",
  "Random number not odd": 2,
  "Randomly choose from selection": B
}
```

The interface also shows a 'Patterns' panel on the right, which lists built-in string generator patterns and custom string generator patterns. The 'Custom String Generator Patterns' section is currently empty.

Demo Time!

Questions? Submit a comment to the event!

DevTest Community Webinars x

Secure | <https://communities.ca.com/events/3387>

Webex + Dial-In

Event Type: **Demo**
Visibility: Open Event • 205 Views
Last modified on Feb 22, 2017 12:12 PM

Tags: [webcast](#) | [ca service virtualization](#) | [ca sv](#) | [february](#) | [2017](#) | [custom pattern matching](#) [Edit tags](#)

Categories: [CA Service Virtualization](#) [Edit categories](#)

[f](#) [t](#) [in](#) [+](#) 0

0 Comments

[Add a comment](#)

Related Content

- [DevTest Community Webcast - CA Service Virtualization for SAP](#)
- [DevTest Community Webcast - DevTest & Selenium](#)
- [DevTest Community Webcast - APIM and Service Virtualization](#)
- [DevTest Community Webcast - Overview of Service Virtualization 9.5](#)
- [DevTest Community Webcast - Scaling out Virtual Service Environments or Simulators](#)

Recommended Content

- [Is there any way to extract a list of operations from a VSI file](#)
- [Webcast Recap: CA Service Virtualization & CA Test Data Manager Integration](#)
- [How to create stub response with pdf attachment?](#)
- [February 2017 Webcast Replay - LDAP/AD Integration for DevTest](#)
- [Service Virtualization 9.x Implementation Proven Professional Exam](#)

Incoming Links

- [This Week: \[WEBCAST\] Custom Pattern Matching on Service Virtualization](#)
- [What You May Have Missed in the DevTest Community \(January 24 - February 22\)](#)

[Contact Us](#) [Privacy Policy](#) [Legal Notices](#) [Data Transfers](#) [Site Map](#)
Copyright © 2016 CA. All rights reserved.

[f](#) [in](#) [g+](#) [t](#)

[Home](#) | [Top of page](#) | [Help](#)

© 2017 Jive Software | Powered by **jive**



Daniel Mazzei

Engineering Services Architect

Daniel.Mazzei@ca.com

