

Index

Symbols and Numbers

- # (hashtags). *See* **hashtags (#)**
- \$ (stock tickers), **Twitter status, 62**
- / (forward slash) **character, restricted in URIs, 9–10**
- : (operator, **Search API negativity, 142**)
- :) operator, **Search API positivity, 142**
- ? (your application), **Twitter status, 62**
- ? operator, **Search API, 142**
- @ (mentions)
 - overview of, 66–67
 - referencing users in **Search API, 142**
 - Twitter status, 61**
- 200 (OK) HTTP response code, Twitter errors, 96**
- 304 (Not Modified) HTTP response code, Twitter errors, 96**
- 400 (Bad Request) HTTP response code, Twitter errors, 96**
- 401 (Not Authorized) HTTP response code, 34–35, 96**
- 403 (Forbidden) HTTP response code, Twitter errors, 96**
- 404 (Not Found) HTTP response code, Twitter errors, 96**
- 406 (Not Acceptable) HTTP response code, Twitter errors, 96**
- 500 (Internal Server Error) HTTP response code, Twitter errors, 96**
- 502 (Bad Gateway) HTTP response code, Twitter errors, 96**
- 503 (Service Unavailable) HTTP response code, Twitter errors, 96**

A

access tokens

- accessing protected resources with, 199–202
- authenticating **ASP.NET** applications, 206–208
- exchanging request tokens for, 197–199
- saving, 200

accounts

- Gnip, 291
- signup for **Twitter** account, 55–56
- Windows Azure** services, 345–346

accounts, REST API

- defined, 58
- end session method, 75
- overview, 74
- rate limit status method, 77–78
- signing up for, 55–56
- update delivery service method, 75–76
- update profile background image method, 77
- update profile colors method, 76
- update profile image method, 76–77
- update profile method, 78–79
- verify credentials method, 75

activities

- notifications compared with, 292
- polling with **Gnip** API, 292–293
- pushing, 293

ad injection, bot accounts, 252

addressability, REST principle of, 4–5

ADO.NET Data Services, 349

aging mechanism, building into caching, 221

`AllowPartiallyTrustedCallers`

attribute, 30

AND (implicit) operator, **Search API, 140**

API rate limits. *See* **rate limits**

application directories, third-party, 261

applications, cross-platform. *See* **cross-platform applications**

applications, **Windows Azure**

- creating, 353–354
- deploying and promoting, 354–355

`AppManifest.xml` file, **Silverlight, 398**

ASP.NET

- application authentication, 205–208
- Membership Provider, 383–388**

ASP.NET AJAX

- calling **JSONP** services in, 135
- `JavaScriptSerializer`, 110–123

ASP.NET compatibility mode, WCF, 295

assemblies, accessing web from inside compiled, 30

asynchronous operation

- priority queues and, 241–242
- rate throttling and, 328
- TweetSharp, 310–311, 326–327

Atom feeds

- consuming feeds with LINQ to XML, 163–167
- consuming feeds with `SyndicationFeed` and `SyndicationItem`, 167–171
- output, 160–162
- payload size, 256
- reasons for syndication, 162–163
- synchronizing applications with feed updates, 171–177
- syndication on web with, 157–158
- TweetSharp representational formats, 309

AtomEntry class, 163–165

AtomFeed class, 163–165

atomic statuses, 221

attributes, XML, 106–110

authentication

- Azure tables and, 360
- Basic. *See* Basic authentication
- Digest, 181
- Gnip accounts, 291
- HTTPS, 180
- OAuth. *See* OAuth
- shared key authentication, 346–348
- storing authentication values, 230
- third-party applications for, 257
- TweetSharp support, 310
- for Twitter users, 60

authorization

- Basic, 42–43
- improving for desktop applications, 211
- NURL enhancing, 357–358
- out-of-band authorization, 197
- redirecting user to provider authorization site, 196–197

authorization header

- restricted in Silverlight, 17
- sending OAuth credentials through, 189–190

Azure cloud

- ASP.NET membership in, 383–388
- designing applications for, 348–350
- Hosted Services, 343–345
- hosting Twitter proxy in, 371–377
- running global user cache in, 377–383
- Storage Services, 341–343

Azure Development Fabric, 343

Azure Hosted Services, 343–345

Azure Storage Services, 341–343

B

background events

- priority queues and, 244–247
- working effectively with rate limits, 241

background image, updating, 77

bandwidth, compression and, 253

base class, programming Windows Azure against, 346

Base64 encoded values, in Basic authentication, 34, 179–180

Basic authentication

- alternatives to, 180–181
- data portability and, 181–182
- defined, 179
- Gnip accounts and, 291
- HTTP, 33–35
- pitfalls of, 179–180
- storing authentication values, 230
- third-party applications and, 257
- TweetSharp supporting, 310
- of Twitter users, 60

Before extension methods, TweetSharp, 309

Binary Large Objects (BLOBs), 342, 349

biographical data, LoremIpsum class for generating, 266

birdog option, user stream methods, 301

bit.ly, URL shortening service, 331

BLOBs (Binary Large Objects), 342, 349

blocking method, 82

blocks, in REST API, 58, 81–83

BMP image format, 237

body styles, WebMessageBodyStyle, 295

Boolean logic operator, Search API, 140

bot accounts, 252

C

Cache class, 229

CacheEntry class, 217–218, 227

CacheWith method, 325–326

caching

- global user cache, 247–252, 377–383
- queries with TweetSharp, 324–326
- rate limits and, 213–214
- real-time relevance and, 221

- statuses (tweets), 221–227
 - thread safety and exception handling and, 218–221
 - user photos, 214–218
 - users, 227–229
 - callback URLs, authenticating ASP.NET applications, 205–206**
 - callbacks, Search API, 144**
 - case sensitivity, `HttpUtility` and URI specification, 12**
 - `ChannelManager`, **WCF duplex service, 282–285**
 - characters**
 - limits on Twitter messages, 85
 - restricted in URIs, 9–10
 - Twitter status, 61–62
 - URI escaping/encoding and, 10–13
 - clients**
 - building to consume server messages in data push, 281–289
 - custom client configuration with `TweetSharp`, 307–308
 - defining contracts for WCF duplex service, 274–275
 - Cloud Service Project type, 353–354**
 - clouds, 340. See also Azure cloud**
 - code feature, `TweetSharp`, 306–307**
 - component instances, creating new URI from, 7**
 - compression**
 - response data, 253–256
 - `TweetSharp` requests, 326
 - console applications, OAuth authentication for, 208–210**
 - consumers, OAuth specification, 183**
 - context, Search API, 143**
 - contracts, defining for WCF duplex service, 274–275**
 - Create, Retrieve, Update, and Delete (CRUD), 5–6**
 - create method**
 - adding status to favorites, 79–80
 - blocks, 81
 - friendships, 72
 - credentials**
 - Basic authentication and, 179–180
 - checking, 75
 - encrypting on a machine key, 232–237
 - `NetworkCredential` and pre-authentication, 38–40
 - OAuth authentication and, 183, 189–191
 - passing Basic authentication in URLs, 35
 - passing user credentials as HTTP POST parameters, 259
 - posting tweet with `WebClient` requiring, 24
 - protecting passwords and user names, 230–231
 - storing authentication values, 230
 - cross-platform applications**
 - ASP.NET membership in Azure cloud, 383–388
 - Azure Hosted Services and, 343–345
 - Azure Storage Services and, 341–343
 - base class for programming against Azure, 346
 - BLOBs (Binary Large Objects), 349
 - building image handler for Silverlight application, 388–391
 - building search poller for Silverlight application, 392–398
 - creating Azure services account, 345–346
 - creating new Azure application, 353–354
 - debugging, 350–351
 - deploying and promoting Azure applications, 354–355
 - designing applications for Azure Cloud, 348
 - hosting Twitter proxy in Azure cloud, 371–377
 - logging, 351–353
 - overview of, 339–340
 - problem and design, 340
 - queues, 349
 - reasons for using Azure, 340–341
 - running global user cache in Azure cloud, 377–383
 - shared key authentication, 346–348
 - Silverlight going cross-platform, 398–399
 - `StoredClient`, 365–367
 - tables, 349
 - web and worker roles, 348–349
 - working with Azure queues, 355–360
 - working with Azure tables, 360–365
 - working with table entities, 367–370
 - CRUD (Create, Retrieve, Update, and Delete), 5–6**
 - current trends feature, Search API, 139**
 - custom application source, 93–94**
- ## D
- daily trends feature, Search API, 139–140**
 - data**
 - caching data elements, 221
 - compressing, 253–256
 - filtering, 252–253

data (continued)

- generating in unit testing, 265–266, 336–337
- portability, 181–182
- pulling. *See* pulling data
- pushing. *See* pushing data
- retrieving for popular users, 247–252

data class conversion feature, TweetSharp, 311

Data Services, ADO.NET, 349

`DataContractJsonSerializer`, **123–126, 154–155**

`DataGeneration` utility class, **265–266, 268–270**

`DataTemplate` class, **267–268**

`DateTime` format

- Azure tables and, 364
- generating timestamps for OAuth, 184
- overview of, 92–93
- RSS and Atom and, 167
- Search API, 138
- TweetSharp, 320
- using `DataContractJsonSerializer`, 126
- using `JavaScriptSerializer`, 119–123
- using LINQ to XML, 101–102
- using XML attributes and `XmlSerializer`, 109

debugging, Windows Azure and, 350–351

Decrypt operations, DAPI support and, 231–232

Deflate compression, 254–256

deploying Azure applications, 354–355

`Dequeue` method, **242**

`Deserialize` method,
`JavaScriptSerializer`, **114**

desktop applications

- improving user authorization for, 211
- OAuth authentication for, 208–210

destroy method

- blocks, 82–83
- direct messages, 71–72
- friendships, 72–73
- statuses, 67, 80

Development Fabric, Windows Azure, 350–351

Digest authentication, 180

Digg, 292

direct messages

- defined, 57
- destroy method, 71–72
- elements of, 90
- generating, 268–270
- new method, 71
- object model, 89–90

- overview of, 70
- sent method, 70–71
- TweetSharp working with, 315

`DirectMessages` extension method,
TweetSharp, 315

discoverability, TweetSharp feature set, 306

disk, caching to, 217–218, 221

DPAPI security, 230

duplex services, 274. *See also* WCF duplex service

E

EAV (Entity-Attribute-Value) format, Azure tables, 349

elements

- caching data elements, 221
- direct message, 90
- error, 92
- rate limit status, 91
- REST API, 59
- source element, 93–94
- status, 86
- user, 88–89

encoding

- Base64 encoded values, 34, 179–180
- ISO-8859-1 encoding type, 237
- OAuth specification for, 183
- raw binary, 237–239
- URI escaping as form of, 10–13

Encrypt operations, DAPI support and, 231–232

encryption

- credentials, 232–236
- OAuth support for signature, 188–189
- passwords, 236–237
- RSA encryptions keys, 230

`EncryptionSurrogate` class, **236–237**

end session method, 75

Entity-Attribute-Value (EAV) format, 349

errors. *See also* exception handling

- elements of, 92
- handling, 95–96
- object model and, 91–92
- TweetSharp, 329–330

`EscapeUriString` method, **12**

escaping, URI, 10–13

exact phrase matching, Search API, 141

exception handling. *See also* errors

- caching multithreaded applications, 218–221
- .NET tools for web communication, 32–33
- overview, 42

exclusions, Search API, 141`ExecutePut` method, 357–358**exists method**

- blocks, 81–82
- friendships, 73

Expect method, TweetSharp, 335–336`Expect100Continue` property, 29, 36–37**explicit OR operator, Search API, 141****extensions, third-party. See third-party applications****F****factories, WCF duplex service, 275–277****fail whale, error handling, 95****favorites**

- create method, 79–80
- data representation through, 79
- defined, 58
- destroy method, 80

feed updates, synchronizing feeds with polling, 172–177`filter:links` operator, Search API context, 143**filters**

- data, 252–253
- setting up Gnip publisher filters and rules, 292

firehose, public stream methods, 300**firewalls, 35–36****fluent interface, TweetSharp as, 305–306****followers method**

- social graphing, 74
- users, 68–69

following mechanism

- filtering data and, 252
- notifications and, 80
- TweetSharp working with, 316–317

form controls, synchronizing feeds with polling, 172–177**formats**

- REST API, 59
- REST principle of, 5
- TweetSharp, 309

friends method

- social graphing and, 74
- users and, 68

friends timeline

- favorites and, 79
- notifications and, 80–81
- statuses and, 63–64

friendships, REST API

- create method, 72
- defined, 57
- destroy method, 72–73
- exists method, 73
- methods for, 72

friendships, TweetSharp, 316–317**from: operator, referencing users in Search API, 141****G****gardenhose, public stream method, 300****gateways, working with, 35–36****geo location, Search API parameters, 143****GET. See HTTP GET**`GetCredentials` method, 39**GIF file format, 237****global user cache**

- retrieving data for popular users, 247–252
- running in Azure cloud, 377–383

globalization, Search API parameter for, 144**Globally Unique Identifier (GUID), 344****Gnip API, 289–300**

- authenticating Gnip accounts, 291
- .NET library provided with, 290–291
- overview, 289
- polling activities, 292–293
- pushing activities, 293
- setting up publisher filters and rules, 292
- using `WebHttpBinding` for REST services, 293–300

Google

- Base specification, 167
- user experience research, 211

GUID (Globally Unique Identifier), 344**GZIP**

- compressing response data, 254–256
- enabling for use with TweetSharp, 326

H**handling errors. See errors****handling exceptions. See exception handling hashtags (#)**

- monitoring hashtag channel for updates, 252
- Search API, 141
- statuses and, 61
- URI escaping and, 11

headers, HTTP

- adding or changing request headers, 16
- authorization header, 17, 189–190
- customizing request headers, 94
- features of, 20–21
- general request headers, 13–16
- overview of response headers, 19–20
- restricted request headers, 16–17
- selected request headers, 14–15
- TweetSharp and, 307

help section, Twitter API, 83

HMAC-SHA1 encryption, 188–189

host, WCF duplex service, 275–277

Hosted Services, Windows Azure, 343–345

HOSTs file, authenticating ASP.NET applications, 206

HTML 4.01 specification, 12

HttpUtility

- escaping and encoding with `Uri` and, 11–12
- URI specification and, 12

HTTP

- anatomy of requests, 13–17
- anatomy of responses, 17–20
- Basic authentication, 33–35
- features of headers, 20–21
- mapping verbs to REST actions, 5–6

HTTP DELETE

- deleting Azure queues, 358–359
- overview of, 50–52
- security restrictions with JSONP, 134

HTTP GET

- anatomy of, 13
- listing all Azure queues, 359–360
- mapping services to HTTP GET requests, 294–295
- methods for, 43–45

HTTP POST

- creating Azure tables, 363–365
- overview of, 45–48
- passing user credentials as HTTP POST parameters, 259
- security restrictions with JSONP, 134
- sending over socket connection, 30–31
- uploading files with multi-part form posts, 237–239

HTTP PUT, 48–50

HTTPS authentication, 180

HttpRequest class

- adding or changing request headers using, 16
- authenticating Gnip accounts, 291

Basic authentication, 34

- defined, 22
- enhancing control with, 27–30
- mocking and, 262–263
- restricted headers, 16–17
- security restrictions with JSONP, 133–134
- sending OAuth requests, 193
- using correct proxy settings with, 35–36

HttpResponse class

- defined, 22
- enhancing control with, 27–30
- handling exceptions, 32–33
- mocking and, 262
- rate limits and, 239–240

I

IDs

- destroying specific status by, 67
- disambiguating, 64
- show method for retrieving status based on, 65–66
- TweetSharp time windows and, 309

Image control, Silverlight, 388–389

images

- building image handler for Silverlight application, 388–391
- caching, 214–218
- updating user's profile and background, 77
- uploading image files with multi-part form posts, 237–239

ImageShack, 259

in_reply_to_status_id, status updates, 67

Intellisense, TweetSharp queries and, 309

interface, REST uniform, 5–6

InvalidOperationException, XML serialization issue, 108

IPushClient, 275

is.gd, URL shortening service, 331

ISO-8859-1 encoding type, 237

IsolatedStorage, caching to disk and, 221

J

JavaScriptConverter class, 111

JavaScriptSerializer class

- caching to disk and, 217–218
- converting results to trends, 150–155
- deserialization of streamed data, 301–303
- using ASP.NET AJAX's, 110–123

JPG image files, 237

jQuery, for web requests on client side, 97

JSON

payload size, 256

Search API and, 144

TweetSharp and, 309

WebMessageFormat, 295

JSON responses

ASP.NET AJAX's `JavaScriptSerializer`,
110–123

WCF's `DataContractJsonSerializer`,
123–126

working with, 110

JSON.NET open source project, 122–123, 306

JSONP (JSON with padding)

calling back to client code with, 126–127

calling services in ASP.NET AJAX, 134

Search API supporting callbacks, 144

security restrictions with, 133–134

using Silverlight to support, 127–133

L

lang parameter, Search API globalization, 144

leave method, notifications, 81

links, Search API context, 143

LINQ

filtering data, 252–253

priority queues and, 242

LINQ to XML

consuming feeds with, 163–167

using, 100–106

logging, in Windows Azure, 351–353

login, improving authorization for desktop applications, 211

`LoremIpsum` class, 266

M

machine key, methods for accessing or generating, 236

`MappingJavaScriptConverter` class,
114–121

mash ups, social services, 388

max_id parameter, sequencing using, 63

MD5 hashes, 360

Membership Provider, ASP.NET, 383–388

`MembershipProvider` class, encrypting user

credentials on a machine key, 232–236

mentions. *See* @ (mentions)

method chaining, fluent interfaces and, 306

MIT license, for TweetSharp, 305

mobile applications, authentication for, 208–210

mocking

responses in unit testing with TweetSharp,
335–336

RESTful services and, 262–265

text, 266–269

monitoring hashtag channel, for updates, 252

monitoring network traffic, with Wireshark, 25

multithreaded applications, caching, 218–221

N

namespaces, Search API, 167

`NameValueCollection`, posting parameters
with `WebClient` using, 25–26

negativity, Search API, 142

**nested representations, LINQ to XML for,
103–105**

.NET Framework

TweetSharp supporting, 307

Twitter compared with .NET, 36

.NET libraries

Gnip API providing, 290–291

TweetSharp as, 305

.NET web communication tools

Basic authentication, 33–35

Basic authorization, 42–43

creating request utility, 41

enhancing control with `HttpRequest /
WebResponse`, 27–30

exception handling, 32–33, 42

`Expect100Continue`, 36–37

HTTP DELETE, 50–52

HTTP GET, 43–45

HTTP POST, 45–48

HTTP PUT, 48–50

low level control using sockets and

`TcpClient`, 30–31

`NetworkCredential` and pre-authentication,
38–40

running NUrl on command line, 53

simplifying with `WebClient`, 22–27

URI validation, 41–42

`UseNagleAlgorithm`, 37–38

working with proxies, gateways and firewalls,
35–36

`NetworkCredential`, pre-authentication and,
38–40

new method, direct message, 71

Newton-King, James, 122

nonces, OAuth specification for, 184–185

normalized URLs, 186–188

notifications

defined, 58

Gnip API and, 292

removing users from, 81–82

REST API, 80–81

NUrl

caching to disk and, 217–218

command line access to Twitter API, 53

enhancing for custom authorization, 357–358

for HTTP requests against Twitter API, 305

O

OAuth

accessing protected resources, 199–202

authenticating ASP.NET web applications,
205–208

authenticating desktop, console or mobile
applications, 208–210

configuring custom source for, 93–94

consumers and publishers and, 183

defined, 33

Digest authentication vs., 181

encoding, 183

exchanging request token for authorized access
token, 197–199

functions for sending requests, 193

generating nonces, 184–185

generating signatures, 185–189

generating timestamps, 184

improving authorization for desktop

applications, 211

overview of, 182–183

redirecting users to provider authorization site,
196–197

retrieving unauthorized request token, 192–195

sending credentials, 189–191

setting up Twitter application for, 203–205

storing authentication values, 230

test servers, 192

third-party applications and, 257

TweetSharp supporting, 310, 320–323

Twitter preference for, 60, 179

workflow, 191–192

**object graphs, syndicated content vs., 158,
169–171**

objects, REST API, 84–92

direct messages, 89–90

errors, 91–92

rate limit status, 91–92

statuses, 84–86

users, 86–89

objects, search, 144–150

**offline development/testing, mocking
and, 335**

opaque strings, realms, 34–35

OpenSearch **technology, 167**

operation modes, TweetSharp, 310–311

operators, Search API, 140–143

OR (explicit) operator, Search API, 141

out-of-band authorization, 197

P

page **parameter, Search API, 144**

pagination

Search API, 144

TweetSharp, 309

Twitter API, 63

parameters

passing user credentials as HTTP POST
parameters, 259

posting with `WebClient`, 25–26

search, 143–144

sequencing using, 63

`ParseQueryString` **method,**

`HttpUtility`, **193**

parsing, using LINQ to XML, 101

passwords

Basic authentication, 34, 179–180

OAuth, 183

protecting, 230–231

URLs for passing authentication, 35

`WebClient` and, 24

payload, comparing payload size by format, 256

performance

caching statuses and users, 221–229

caching user photos, 213–221

compressing response data, 253–256

extending Twitter with third-party applications,
256–261

filtering data, 252–253

rate limits and, 239–247

retrieving data for popular users, 247–252

storing authentication values, 230–237

TweetSharp and, 323

unit testing and, 262–270
 uploading image files with multi-part form posts,
 237–239

photos

third-party applications for, 257–259
 TweetSharp photo posting feature, 332–333

PNG image files, 237

polling

building search poller for Silverlight applications,
 392–398
 duplex support in Silverlight 3, 277
 with Gnip API, 292–293
 pushing data vs., 273–274
 synchronizing feeds with, 172–177

positivity, Search API, 142

POST. *See* HTTP POST

priority queues, for working with rate limits, 241–242

PriorityQueue **class**, 242

profiles, obtaining user profiles with TweetSharp, 313–314

proxies

hosting in Azure cloud, 371–377
 TweetSharp and, 334–335
 working with, 35–36

public stream methods, 300

public timelines, Twitter API, 62

publishers

exchanging request token for access token
 from, 197–199
 OAuth specification for, 183
 publishing Azure applications, 354–355
 redirecting users to, 196–197
 setting up Gnip filters and rules, 292

pulling data. *See also* polling

overview of, 273–274
 RSS Time to Live property and, 172

pushing data

advantages/disadvantages, 273–274
 Gnip API and, 293
 overview, 273
 Streaming API for, 300–303
 WCF duplex service. *See* WCF duplex service
 web syndication and, 172

Q

queries

absence of query language in Twitter, 253
 caching, 324–326
 converting, 150–155

method chaining and, 306
 rate throttling and, 328
 search, 144–148
 TweetSharp query format, 309

questions, Search API, 142

Queue **class**, 242

queues, Windows Azure

creating, 356–358
 deleting, 358–359
 listing all queues on an account, 359–360
 overview of, 349
 Storage Services and, 342
 working with, 355–356

R

rate limit status

accounts, 77–78
 DateTime format and, 93
 elements of, 91
 object model, 91–92

rate limits

caching and, 213
 designing applications and, 241–247
 determining in TweetSharp, 328–329
 popular users and, 247
 push data and, 273
 responses and, 239–240
 Search API and, 138
 Twitter API and, 60–61

rate throttling, TweetSharp, 328

raw binary encoding type, HTTP POST with image file, 237–239

ReadLine **method**, StreamReader, 303

Really Simple Syndication. *See* RSS (Really Simple Syndication) feeds

realm identity, Basic authentication, 34–35

real-time

caching and, 221
 filtering data and, 252

referencing users, Search API operators for, 141–142

relational data, on Windows Azure, 350

relative time feature, TweetSharp, 331

Remote Procedure Call (RPC), 2

RepeatAfter **method**, TweetSharp, 327–328

RepeatEvery **method**, TweetSharp, 327–328

replay attacks, avoiding, 183

Representational State Transfer. *See* REST (Representational State Transfer)

representations, REST principle of, 3

request formats, `WebMessageFormat`, 295

request headers

adding or changing on .NET, 16

customizing Twitter client, 94

features of, 20–21

general, 13–16

restricted, 16–17

selected general, 14–15

`TweetSharp` and, 307

Request Line, HTTP, 13

request utility, creating, 41

`RequestAsync` method, `TweetSharp`, 326–327

requests, HTTP

anatomy of, 13–17

compressing with `TweetSharp`, 326

creating utility for, 41–42

mocking, 262–264

optimizing, 27–30

requesting data from Twitter, 96–97

requests, OAuth

authenticating ASP.NET applications, 206–208

exchanging request token for access token,
197–199

functions for sending, 193

retrieving unauthorized request token, 192–195

reserved characters, URIs, 9–13

resources

addressability principle of REST and, 4–5

REST API, 57–58

REST principle of, 3

response headers

features of, 20–21

overview, 19–20

responses, HTTP

anatomy of, 17–20

compressing response data, 253–256

error codes, 96

handling exceptions using, 32–33

mocking, 262–264

optimizing, 27–30

rate limits and, 239–240

representation, in REST API, 58–60

REST (Representational State Transfer)

addressability and, 4–5

defined, 1

encapsulating REST data formats in

`DataTemplate` class, 267–268

formats, 5

overview, 1–2

resources and representations, 3

stateless operation, 5

Twitter deviations from concepts of, 21–22

uniform interface, 5–6

REST API

accounts, 75–79

authenticating users, 60

blocks, 81–83

building, 293–295

configuring custom application source, 93–94

`DateTime` format, 92–93

direct messages, 70–72, 89–90

errors, 91–92

favorites, 79–80

friendships, 72–73

handling errors, 95–96

help, 83

notifications, 80–81

overview, 55–57

rate limits, 60, 91–92

requesting data from Twitter, 96–97

resources, 57–58

representation, 58–60

setting up to use OAuth, 203–205

social graphing, 73–74

statuses, 61–67, 84–86

users, 67–70, 86–89

RESTful services

anatomy of HTTP requests, 13–17

anatomy of HTTP responses, 17–20

anatomy of URI, 7–13

application directories, 261

`Azure StorageClient` and, 366

mocking, 262–265

planning features using HTTP headers, 20–21

streaming implemented as REST service, 300

Twitter and, 21–22

web communication tools. *See* .NET web
communication tools

`WebHttpBinding` for, 293–300

restricted characters, URIs, 9–13

retrieving data, for popular users, 247–252

`Retweet` method, `TweetSharp`, 333–334

Reuse and Recycle: Online Password

Management, 180

roaming users, authentication and, 232–233

`RoleManager` class, for logging in Azure, 351

rollback scenarios, Windows Azure, 355

RPC (Remote Procedure Call), 2

`rpp` parameter, `Search API`, 144

RSA encryptions keys, 230**RSS (Really Simple Syndication) feeds**

- consuming feeds with LINQ to XML, 163–167
- consuming feeds with SyndicationFeed and SyndicationItem, 167–171
- output, 158–160
- payload size, 256
- reasons for syndication, 162–163
- synchronizing applications with feed updates, 171–177
- syndication on web, 157–158
- TweetSharp representational formats, 309

rules, setting up Gnip publisher filters and rules, 292**running Twitter streams, 301–303****S****saving access tokens, 200****screen names, disambiguating, 64**

ScriptableMember, supporting JSONP with Silverlight, 130

ScriptableType, supporting JSONP with Silverlight, 130

Search API, 137–156

- converting queries and trends, 150–155
- DateTime format and, 93
- features, 138–140
- namespaces, 167
- objects, 144–150
- operators, 140–143
- overview of, 137–138
- parameters, 143–144
- updating content in polling application based on, 177

search extension methods, TweetSharp, 318**search poller, for Silverlight applications, 392–398****searches, TweetSharp, 317–319****Secure Sockets Layer (SSL), 180****security**

- Basic authentication. See Basic authentication
- Digest authentication, 181
- HTTPS authentication, 180
- JSONP restrictions, 133–134
- OAuth authentication. See OAuth

sent method, direct messages, 70–71**sequencing, Twitter API, 63****sequential operation mode, TweetSharp, 310–311****serialization/deserialization**

- of CacheEntry object, 217–218
- of streamed data, 301–303

Serialize **method**, JavaScriptSerializer, 114

server messaging, WCF duplex service providing, 278–281**server overhead, push services and, 274**

ServiceContract, **WCF, 274–275, 294**

ServiceHost, **WCF, 275**

ServicePointManager.Expect100Continue **class**, WebClient **and**, 37

services

- ADO.NET Data Services, 349
- Azure account services, 345–346
- Azure Hosted Services, 343–345
- Azure Storage Services, 341–343
- RESTful. See RESTful services
- WCF duplex service. See WCF duplex service
- WCF Syndication Services, 167–171

session state, ending user sessions, 75**shadow option, user stream methods, 301****shared key authentication, Windows Azure, 346–348****show method**

- statuses, 65–66
- users, 69–70

signature base, OAuth, 186–188**signature generation**

- OAuth specification for, 185–189
- retrieving unauthorized request tokens and, 192–193

signup, Twitter account, 55–56**Silverlight**

- Authorization header restricted in, 17
- building image handler for Silverlight applications, 388–391
- building search poller for Silverlight applications, 392–398
- cross-platform applications, 398–399
- JSONP support, 127–133
- overview of Silverlight applications, 388
- web requests on client side, 97

Since **extension method**, TweetSharp, 309

since: **operator**, Search API **positivity**, 142

since_id **parameter**

- Search API timelines, 144
- sequencing using, 63

Skip **method**, TweetSharp **pagination**, 309

sliding expiration window, caching and, 227

social graphing

- followers method, 74
- friends method, 74
- overview, 73–74
- REST API and, 58
- retrieving data for popular users, 247–248

sockets, low level control using, 30–31

source element, configuring custom application source, 93–94

source link, identifying applications by source, 308

source: operator, Search API context, 143

SQL Server Data Services (SSDS), 350

SSDS (SQL Server Data Services), 350

SSL (Secure Sockets Layer), 180

stateless operation, REST principle of, 5

status codes for HTTP

- responses, 17–20
- Twitter deviations from REST, 21–22

status line, HTTP responses, 17

statuses (tweets)

- 140-character limit on, 85
- atomic, 221
- caching, 221–227
- defined, 57
- destroy method, 67
- elements of, 86
- generating with `DataGeneration` utility class, 268–270
- generating with `LoremIpsum` class, 266
- mentions (@), 66–67
- object model, 83–86
- overview, 61
- posting with `TweetSharp`, 315
- sequencing and pagination, 63
- show method, 65–66
- special characters for, 61–62
- thread safety and, 224
- updates, 66
- viewing friends timeline, 63–64
- viewing public timeline, 62
- viewing user timeline, 64–65

storage services, Windows Azure, 341–343

`StoredClient`, Windows Azure, 365–367

Streaming API

- overview, 300
- public stream methods, 300
- running Twitter streams, 301–303
- user stream methods, 301

`StreamReader`, 303

`StringBuilder` class, 306

strings, creating new URI from, 7

styles, updating user's profile page, 76

syndication

- Atom. See Atom feeds
- RSS. See RSS (Really Simple Syndication) feeds

Syndication Services, WCF, 167–171

`SyndicationFeed` class

- consuming feeds with, 167–171
- synchronizing applications with feed updates using, 171–177

`SyndicationItem` class, 167–171

`System.Net.Uri` class

- defined, 7
- URLs and, 9

`System.Web.HttpUtility`. See `HttpUtility`

`System.Web.Script.Serialization` namespace, 110

T

tables, Windows Azure

- in Azure Storage Services, 342
- creating, 363–365
- overview of, 349
- working with, 360–363
- working with table entities, 367–370

`Take` method, `TweetSharp` pagination methods, 309

`TcpClient` class

- low level control using, 30–31
- security restrictions with JSONP, 133–134

test method, Twitter API help section, 83

test servers, OAuth implementation, 192

testing. See unit testing

text, mocking, 266–269

third-party applications

- application directories, 261
- extending Twitter with, 256–257
- for photo features, 257–259
- for trend analysis, 259–261

thread safety

- caching multithreaded applications, 218–221
- statuses cache and, 224

time. See also DateTime format

- relative time feature in `TweetSharp`, 331
- `TweetSharp` time extension helpers, 320

Time to Live (TTL) property, RSS feeds, 172

timed tasks, TweetSharp, 327–328

timelines

- fetching with TweetSharp, 312–313
- removing users from, 81–82
- Search API operators for, 142–143
- Search API parameters for, 144

timestamps

- generating nonces for, 184–185
- OAuth credentials using, 183
- OAuth specification for, 184

tinyurl, URL shortening service, 331

to: **operator, referencing users in Search API, 141**

to.m8, URL shortening service, 331**trends**

- converting, 150–155
- defined, 138
- object model for, 148–150
- third-party applications for, 259–261
- TweetSharp working with, 317–319

tr.im, URL shortening service, 331**try pattern validation, 7****TryParse pattern, using LINQ to XML, 101****TTL (Time to Live) property, RSS feeds, 172**

TweetBox, **synchronizing feeds with polling, 172–177**

TweetDeck, 257**tweets. See statuses (tweets)****TweetSharp**

- asynchronous operation, 326–327
- authentication, 310
- caching queries, 324–326
- code feature, 306–307
- community and resources supporting, 314
- compressing requests, 326
- custom client configuration, 307–308
- custom proxy values supported by, 372
- data class conversion, 311
- data generation for unit testing, 336–337
- fetching timelines with, 312–313
- as fluent interface, 306
- handling errors, 329–330
- mock responses in unit testing, 335–336
- .NET Framework support, 307
- OAuth authentication and, 320–323
- obtaining user profiles, 313–314
- operation modes, 310–311
- overview, 305
- pagination methods, 309
- performance, 323
- photo posting feature, 332–333

- posting statuses, 315
- query format, 309
- rate throttling, 328
- relative time feature, 331
- retweeting, 333–334
- search extension methods, 318
- Since, 309
- time extension helpers, 320
- timed tasks, 327–328
- unit testing, 335
- URL shortening feature, 331–332
- working with direct messages, 315
- working with friends and followers, 316–317
- working with proxies, 334–335
- working with rate limits, 328–329
- working with searches and trends, 317–319

Twitgoo, 257–259**Twiticism.com, 339. See also cross-platform applications****TwitPic**

- adding third-party photo features, 257–259
- TweetSharp support for, 332–333

TwitScoop, 259–261**Twitter**

- vs. .NET, 36
- contextual namespace, 167
- hacking of, 181
- REST and, 21–22

TwitterClientInfo **class, TweetSharp, 307–308**

TwitterExplorer application, 57

TwitterExtensions **code, 153**

TwitterFeedInfo **class, 169**

TwitterFeedResult **class, 169**

TwitterRateLimitStatus **object, 247**

TwitterRequestQueue **class, 243–244**

TwitterSearchStatus **object**

- converting to, 154–155

- search queries, 146–148

TwitterStatus **object, 144–148**

U

UnescapeUriString **method, Uri, 12**

unit testing

- generating data, 265–266
- mocking RESTful services, 262–265
- mocking text, 266–269
- overview, 262
- TweetSharp and, 335

Universal Resource Indicators. See URIs (Universal Resource Indicators)

Universal Resource Locators. See URLs (Universal Resource Locators)

Unix Epoch time, OAuth timestamps in, 184
until: operator, Search API positivity, 143
update delivery service method, Twitter API accounts, 75–76

update profile background image method, Twitter API accounts, 77

update profile colors method, Twitter API accounts, 76

update profile image method, Twitter API accounts, 76–77

update profile method, Twitter API accounts, 78–79

updates

statuses, 66
synchronizing applications with feed, 171–177

uploading files, with WebClient, 26–27

Uri class

escaping and encoding with, 9–13
overview, 8

URIs (Universal Resource Indicators)

addressability principle of REST and, 4–5
anatomy of, 7–9
escaping and encoding with Uri and HttpUtility, 9–13
format principle of REST and, 5
reading and downloading data with WebClient, 22–24
URLs vs., 3
validation of, 41–42

UriTemplates, 295

URL shortening, 257, 331–332

UrlDecode, HttpUtility, 11

UrlEncode, HttpUtility, 11

URLs (Universal Resource Locators)

authenticating ASP.NET applications, 205–206
compact URL, 257
identifying applications by source, 308
mocking and, 262
normalized, 186–188
OAuth specification encoding, 183
passing Basic authentication in, 35
providing endpoint URL for push service, 293, 299
search parameters, 143
sending OAuth credentials through request, 190–191

Unicode symbols for reducing size of, 85
URLs vs., 3

UseNagleAlgorithm, 37–38

UseProxy method, TweetSharp, 334–335
user agent, configuring custom application source, 93–94

user names

Basic authentication, 34, 179–180
LoremIpsum class for generating, 266
OAuth benefits, 183
passing authentication in URLs, 35
protecting, 230–231
working with WebClient, 24

user representations, using LINQ to XML, 103–105

user sentiment, Search API, 142

user stream methods, Streaming API, 301

UserControl, synchronizing feeds with polling, 172–177

users

authentication, 60
caching, 227–229
caching user photos, 214–218
defined, 57, 67
elements of, 88–89
followers method, 68–69
friends method, 68
generating with DataGeneration utility class, 268–270
OAuth sending to publisher's site, 196–197
object model, 86–89
obtaining user profiles with TweetSharp, 313–314
referencing in Search API, 141–142
retrieving data for popular, 247–252
show method, 69–70
viewing friends timeline, 63–64
viewing user timeline, 64–65

Using the Web Permission Setting in ASP.NET Partial Trust (Schakow), 97

V

validation, URI, 8, 41–42

verify credentials method, Twitter API accounts, 75

Visual Studio

building REST API, 293–295
project types, 274
Windows Azure Tools for Visual Studio, 354

W**W3C (World Wide Web Consortium), 12****WCF (Windows Communication Foundation)**

- consuming feeds, 167–171
- defined, 123
- ServiceContract for REST API, 294
- Syndication Services, 167
- using DataContractJsonSerializer, 123–126
- wrapping REST APIs, 33

WCF duplex service

- client consumption, 281–289
- configuring services, 277–278
- creating host and factory, 275–277
- defining contracts, 274–275
- overview, 274
- server messaging, 278–281

Web Application Project, Visual Studio, 274**web communication tools, .NET**

- Basic authentication, 33–35
- Basic authorization, 42–43
- creating request utility, 41
- enhancing control, 27–30
- exception handling, 32–33, 42
- Expect100Continue, 36–37
- HTTP DELETE, 50–52
- HTTP GET, 43–45
- HTTP POST, 45–48
- HTTP PUT, 48–50
- low level control using sockets and TcpClient, 30–31
- NetworkCredential and preauthentication, 38–40
- running NUrl on command line, 53
- simplifying with WebClient, 22–27
- Twitter vs. .NET, 36
- URI validation, 41–42
- UseNagleAlgorithm, 37–38
- working with proxies, gateways and firewalls, 35–36

Web references

- data portability, 181–182
- Google's user experience research, 211
- JSON.NET open source project, 123
- OAuth, 182–183, 192

web roles, Windows Azure, 348, 350**WebClient class**

- adding or changing request headers using, 16
- disabling Nagle algorithm-based optimization, 38

- making calls to Twitter API with, 37
- optimizing with, 27
- simplifying web communication with, 22–26

web.config file, 236**WebException class, 32–33****WebGet attribute, 294–295****WebHttpBinding, for RESTful services, 293–300****WebInvoke attribute, supporting POST, PUT, and DELETE, 294–295****WebMessageBodyStyle, 295****WebMessageFormat, 295****WebPermission class**

- accessing web from inside compiled assemblies, 30
- adding Twitter domain to, 97

WebResponse class, for exception handling, 32–33**weekly trends feature, Search API, 140****white-listed IP addresses**

- API rate limits and, 213–214
- popular users and, 247

Windows Azure

- ASP.NET membership in Azure cloud, 383–388
- base class for programming against, 346
- BLOBs (Binary Large Objects), 349
- creating applications, 353–354
- creating services account, 345–346
- debugging, 350–351
- deploying and promoting applications, 354–355
- designing applications for Azure Cloud, 348
- further reading, 348
- Hosted Services, 343–345
- hosting Twitter proxy in Azure cloud, 371–377
- logging, 351–353
- queues, 349
- reasons for using, 340–341
- relational data on, 350
- running global user cache in Azure cloud, 377–383
- shared key authentication, 346–348
- Storage Services, 341–343
- StoredClient, 365–367
- tables, 349
- web and worker roles, 348–349
- working with queues, 355–360
- working with table entities, 367–370
- working with tables, 360–365

Windows Azure Tools for Visual Studio, 354**Windows Forms, synchronizing feeds with polling, 172–177**

Wireshark

- monitoring network traffic with, 25
- tracing request/response with, 255

worker roles, Windows Azure, 348–350

X

XContainer, **using LINQ to XML, 102–103**

XDocument **instance, using LINQ to XML, 100**

XElement **node, using LINQ to XML, 100**

XML

- payload size, 256

- TweetSharp representational formats, 309

- WebMessageFormat and, 295

XML responses

- using LINQ to XML, 100–106

- using XML attributes and XmlSerializer, 106–110

- working with, 99

XmlSerializer, **106–110**

x-ms-date, **360**

XName **class, 165–166**

Y

Yahoo!'s Media RSS, 167

yFrog

- adding third-party photo features, 257–259

- TweetSharp support for, 332–333