June 2017

etherFAX REST API Reference 2.0

A web based API for sending, receiving and managing fax and document delivery through the etherFAX network.

Interface Overview

The etherFAX API is a web interface based on REST semantics using simple HTTP GET/POST web operations compatible with many operating systems and programming languages. This interface is provided to all application and fax server partners that wish to integrate fax delivery into their applications and services.

This web API provides access to simple resource controllers. At this time, supported controllers are Accounts, Outbox, Status, Inbox and Routes. Each resource supports optional parameters within the URI scheme and provides responses in various data formats (JSON, XML, simple URL encoded data, etc.).

Web Service Base URL

All requests referenced in this document use the following base URL:

https://na.connect.etherfax.net/rest/2.0/api

If a simple GET is performed on the root API URI, the following text will be returned with the remote client's IP address displayed.

> etherFAX REST Services 2.0
>
> Your client address is: 150.76.16.64
>
> Copyright © 2008-2017, etherFAX, LLC

Supported Request Formats

The etherFAX REST API observes the Content-Type value in the HTTP header when performing POST operations (i.e. sending a fax). Supported Content-Type values are:

- application/x-www-form-urlencoded
- application/json

Supported Response Formats

The optional &f parameter may be used to explicitly request JSON or XML responses from the etherFAX web service. If omitted, all responses will be returned in the JSON format.

Date and Time Formats

All date/time values are represented in UTC.

Authentication

The etherFAX REST web service supports the Basic HTTP authorization model. Each client must use the etherFAX

account number, user name and password provided by etherFAX personnel. When authenticating against the REST web services, you must make sure all GET/POST operations have correctly added the Authorization header to HTTP request using basic authentication.

Example:

```
EFAX-9999/user:password
```

HTTP Header:

```
Authorization: Basic RUZBWC05OTk5L3VzZXI6cGFzc3dvcmQ=
```

Accounts Controller

This resource is used to query information from the current account, features that are enabled, supported file formats and other information.

Uri Template:

```
[GET] “accounts”
```

Parameters:

This controller supports no parameters or additional functions at this time.

Example:

```
[GET] https://na.connect.etherfax.net/rest/2.0/api/accounts
```

Response:

```
{
  "Account": "EFAX-9999",
  "Name": "etherFAX Dev",
  "AccountId": 101,
  "Ports": 4,
  "Enabled": true,
  "Features": [
    "FaxResume",
    "NoRedialAttempts",
    "InternationalDialing",
    "InactiveCallRejection"
  ],
  "AcceptedFormats": [
    "image/tiff",
    "application/pdf"
  ],
  "Numbers": 2,
  "Country": 1,
  "CreatedOn": “2009-01-12T11:26:48.237”
}
```

Example: XML response

This example is shown to demonstrate the use of the optional &f parameter supported by each controller.

```
[GET] https://na.connect.etherfax.net/rest/2.0/api/accounts?f=xml
```

Response:

```
<?xml version="1.0" standalone="no"?>
<FaxAccount xmlns:i="http://www.w3.org/2001/XMLSchema-instance
xmlns="http://schemas.datacontract.org/2004/07/EtherFax.Common>
    <AcceptedFormats
xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays>
        <d2p1:string>image/tiff</d2p1:string>
        <d2p1:string>application/pdf</d2p1:string>
    </AcceptedFormats>
    <Account>EFAX-9999</Account>
    <AccountId>101</AccountId>
    <Country>1</Country>
    <CreatedOn>2009-01-12T11:26:48.237</CreatedOn>
    <Enabled>true</Enabled>
    <Features
xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays>
        <d2p1:string>FaxResume</d2p1:string>
        <d2p1:string>NoRedialAttempts</d2p1:string>
        <d2p1:string>InternationalDialing</d2p1:string>
        <d2p1:string>InactiveCallRejection</d2p1:string>
    </Features>
    <Name>etherFAX Dev</Name>
    <Numbers>2</Numbers>
    <Ports>4</Ports>
</FaxAccount>
```

Outbox Controller

This resource is used to send faxes through the etherFAX network to a destination fax number (or etherFAX endpoint).

Uri Template:

```
[POST] "outbox"
```

Parameters:

The following parameters are supported when performing a POST operation to the outbox controller. If you are using the urlencoded form of this function (Content-Type: application/x-www-form-urlencoded), each parameter must be URL encoded in the post data associated with the HTTP request.

Parameter	Description
DialNumber	Specifies the number to be dialed. This may be a long distance human readable number or E.164 (+18005551234) format.
LocalId	(Optional) Sets the fax id (CSID) for the send request. This is the fax CSID displayed by the remote fax system. This field supports up to 21 characters.
CallerId	(Optional) Sets the caller id for this send request. Where possible, this number will appear as the calling party/id to the remote system. For maximum effectiveness within carrier networks, a "toll" DID is highly recommended.
TotalPages	Specifies the total number of pages in the fax image file.
TimeZoneOffset	Specifies the time zone offset of the originator. This value is used to maintain the local time of the originator in the fax header (i.e. -5 for Eastern Standard Time).
Tag	(Optional) Sets a user defined string associated with the fax request. This string will appear in <FaxStatus> responses making it easier for applications to track their fax messages. This field supports up to 32 characters.
FaxImage	Contains the base64 encoded string representing the binary image data (tiff, pdf or other supported document format).
Header	Specifies the header format string displayed at the top of each page of this fax job. Keywords and examples are shown below.
TZ	Specifies the time zone offset of the originator and functions similar to TimeZoneOffset. However, this parameter also supports Linux style time zones such as "America/Chicago", "America/New_York", etc. Using this parameter overrides the TimeZoneOffset value (if specified) and automatically adjusts for daylight savings for the time zone.

Remarks:

After the POST operation, the etherFAX service will respond with an abbreviated <FaxStatus> response indicating whether the fax operation was successfully submitted to the etherFAX network. A FaxResult value of *InProgress* indicates that that the fax has been accepted and a unique identifier has been assigned to the event. All events within the etherFAX network have globally unique identifiers assigned for their lifetime. These are also known as GUIDs.

All fax images presented for transmission should confirm to the TIFF-F minimum baseline (see TIFF specification). For best results, only submit fax images that are bi-tonal (black & white), have a standard fax width of 1728 pixels and are roughly 200x200 (fine) or 200x100 (coarse) dots per inch (resolution).

etherFAX also supports PDF document delivery.

Note, each etherFAX account is configured with a maximum number of ports (channels) to limit the number of outbound fax operations. If the number of active channels exceeds this configuration, the fax event will be rejected with a FaxResult value of *ChannelUnavailable*.

* See FaxResult section for explanation of the <FaxResult> values.

Response:

```
{
  "JobId": "aa52f281-44dd-45bf-a586-d12d127dd733",
  "FaxResult": 2,
  "State": 0,
  "PagesDelivered": 0,
  "ConnectTime": 0,
  "ConnectSpeed": 0,
  "RemoteId": null,
  "Tag": null,
  "CompletedOn": null
}
```

The following actions are also supported on the outbox controller.

Parameter	Description
pending	Returns the number of active/pending events in the outbox for the given account, including the number of channels available for sending.

Example:

```
[GET] https://na.connect.etherfax.net/rest/2.0/api/outbox?a=pending
```

Response:

```
{
  "PendingFaxes": 0,
  "AvailableChannels": 4
}
```

The *header* parameter supports the following keywords and optional formatting:

Keyword	Description
{date}	Shows current date adjusted to account's local time zone.
{date:*format*}	Shows current date adjusted to account's local time zone with optional formatting.
{time}	Shows current time adjusted to account's local time zone.
{time:*format*}	Shows current time adjusted to account's local time zone with optional formatting.
{utcdate}	Shows the current date in UTC.
{utcdate:*format*}	Shows the current date in UTC with optional formatting.
{utctime}	Shows the current time in UTC
{utctime:*format*}	Shows the current time in UTC with optional formatting.
{csid} {from}	Shows the sender CSID's (Call Subscriber Identification) information.
{number} {to}	Shows the destination fax number.
{page}	Shows the current page number.
{pages}	Shows the current page number and total pages in fax. This is equivalent to using: "{page} / {total}"
{total}	Shows the total page count.

Default header used by etherFAX:

```
"  {date}  {time}   FROM: {csid}  TO: {number}   P. {page}"
```

Output:

```
"  05/06/2014  08:46 AM   FROM: etherFAX, LLC  TO: +18005551234    P. 1"
```

Example using date formatting:

```
"  {date:d-MMM-yyyy}  {time}   FROM: {csid}  TO: {number}   P. {page}"
```

Output

```
"  6-May-2014  08:46 AM   FROM: etherFAX, LLC  TO: +18005551234    P. 1"
```

Note: *It is best to leave 2 spaces at the beginning of the header string to avoid the header starting too close to the left edge of the page.*

Status Controller

This resource is used to monitor the status of one or more fax events using their unique identifier (GUID).

Applications SHOULD NOT invoke this method excessively and no more frequently than 1-2 times per minute when polling for fax completion.

The status function now provides the ability to wait for events within the etherFAX network, such as new fax received, fax completed, and fax job status changed (dial state changes, page completed, etc.). This implementation is used as a pseudo "push-notification" service that allows the remote client to respond to events in real-time as they occur vs. simply polling every 30 to 60 seconds. An example of its use is described further in this section.

Uri Template:

```
[GET] "status?id={id}&wait={timeout}&type={type}
[GET] "status?id={id},{id},…
```

Parameters:

The following parameters are supported when performing a GET operation on the status controller. This method is designed to query multiple status' in one single call. This is more resource friendly and produces less network traffic.

Parameter	Description
id	Specifies one or more unique job identifiers (GUID) separated by commas.
timeout	Specifies the number of seconds for the wait timeout. The wait timeout has 2 uses: • Wait for an event to transpire for the specified job id (send complete, in-job status, etc.). • Wait for account wide events to occur (new job received, send completed, etc.).
type	Specifies the type of events to monitor when the wait timeout has been specified. Valid values are: receive, send, status and any. receive wait for receive event (account wide) send wait for send event (id or account wide) status wait for send job status (id or account wide) any any of the above conditions

Remarks:

When using the wait/timeout options in this function, please note that your HTTP client/code must be configured to timeout at or greater than the timeout value you're using. As a general rule, add 5-10 seconds to your HTTP client timeout as compared to the wait timeout.

Example:

[GET] https://na.connect.etherfax.net/rest/2.0/api/status?id=ba93e53b-70b9-433d-a84b-0000013437e6

Response:


```
{
  "JobId": "ba93e53b-70b9-433d-a84b-0000013437e6",
  "FaxResult": 0,
  "State": 0,
  "PagesDelivered": 1,
  "ConnectTime": 41,
  "ConnectSpeed": 14400,
  "RemoteId": "9085551234",
  "Tag": "ID12345",
  "CompletedOn": "2015-04-28T00:35:23.113"
}
```


Example: Status with wait timeout

This example queries the status of the same job id as above, but instructs the service to not return until 60 seconds have transpired -or- the job has completed.

[GET] https://na.connect.etherfax.net/rest/2.0/api/status?id=ba93e53b-70b9-433d-a84b-0000013437e6&wait=60&type=send

Example: Wait for account wide events

This example performs an account wide status query with a 60 second timeout. The response will indicate if any of the conditions have been triggered: new fax received, fax completed, or fax job status.

[GET] https://na.connect.etherfax.net/rest/2.0/api/status?&wait=60&type=any

Response:

The following response indicates that a new fax was received while waiting for the status.


```
{
  "FaxReceived": true,
  "FaxCompleted": false,
  "FaxStatus": false
}
```

Inbox Controller

This resource is used to handle all receive fax operations. The actions described below allow the application to check for newly received faxes and mark them as received once downloaded successfully.

Uri Template:

```
[GET] "inbox?a={action}&id={id}&count={count}&wait={timeout}&download={download}
&sid={sid}"
```

Parameters:

The following actions/parameters are supported when performing a GET operation on the inbox controller. Each parameter must be URL encoded in the post data associated with the HTTP request.

Parameter	Description
list	Returns a list of unread faxes for the given account. All faxes are returned as <FaxReceive> child nodes in the response. The optional {count} value may be specified to limit the number of unread faxes returned in the list. Setting this value to 0 (or omitted) returns all unread faxes. This function only returns the information associated with the received fax and not the actual fax image data.
get	Gets/downloads the <FaxReceive> information for the specified fax {id}. All information including the <FaxImage> content is returned using this method.
getnext	Gets and optionally downloads the next unread fax (if one is available) and returns the <FaxReceive> information. The {download} parameter may be set to 1 or 0 when using this function. This function is designed for multi-process environments that attempt to receive a fax using the same account. To resolve contention, only one peer may gain access to an unread fax at a time. If the peer fails to download the fax and mark it as received within a 5-minute window, the automatic lock on the fax is released and another peer may download the unread fax. Additionally, the peer may also set the {sid} parameter which is a GUID representing the site-id. etherFAX will record the site id responsible for downloading the fax document.
received	Marks the specified fax {id} as "received" indicating that it has been successfully read or delivered. Once an application has guaranteed that it has received the fax intact, it should mark the fax as received, thus permanently removing the fax image resource.
unread	Returns the number of unread fax events for the given account. When using the unread action, you may optionally provide the wait *timeout* value to wait for a new fax.
wait	Specifies the timeout in seconds to wait for a new fax to arrive. This wait method is only valid when using the *unread* action. When waiting, the

	function will return immediately if unread faxes are available for download or a new fax has arrived during the wait period.

Example: Get unread fax count

[GET] https://na.connect.etherfax.net/rest/2.0/api/inbox?a=unread

Response:

```
{
  "UnreadFaxes": 0
}
```

Example: Get unread fax count, wait timeout

The following example queries the number of unread faxes available. If no unread faxes are available at the time this function is called, it will wait up to 120 seconds for a new fax to arrive. If a new fax arrives during the wait/timeout period, this function will return immediately allowing the application to retrieve the new fax the instant it arrives within the etherFAX network.

[GET] https://na.connect.etherfax.net/rest/2.0/api/inbox?a=unread&wait=120

Example: List all unread faxes

[GET] https://na.connect.etherfax.net/rest/2.0/api/inbox?a=list

Response:

```
[
  {
    "JobId": "833fd712-2ca2-4c7b-8b26-8166629963de",
    "FaxResult": 0,
    "ConnectTime": 56,
    "ConnectSpeed": 33600,
    "PagesReceived": 2,
    "RemoteId": "9085551234",
    "CalledNumber": "+18005551234",
    "CallingNumber": "2125551234",
    "ReceivedOn": "2015-10-27T21:47:37.92",
    "FaxImage": null
  },
  {
    "JobId": "c12e1f95-77e7-4471-a456-508e43daa9a9",
    "FaxResult": 0,
    "ConnectTime": 68,
    "ConnectSpeed": 33600,
    "PagesReceived": 3,
```

```
    "RemoteId": "9085551234",
    "CalledNumber": "+18005551234",
    "CallingNumber": "2125551234",
    "ReceivedOn": "2015-10-27T21:47:43.14",
    "FaxImage": null
  }
]
```


Example: Mark fax received

[GET] https://na.connect.etherfax.net/rest/2.0/api/inbox?a=received&id=c12e1f95-77e7-4471-a456-508e43daa9a9

Routes Controller

The routes controller provides information about routes associated with the account, number parsing and information about an endpoint (supported document formats, etc.).

Uri Template:

```
[GET] “routes?a={action}&id={id}&country={country}”
```

Parameters:

The following actions are supported for the numbers controller. Each parameter must be URL encoded in the post data associated with the HTTP request.

Parameter	Description
parse	Parses the specified number in {id} and returns information such as destination country, etc. The {country} parameter may be used to specify the country of origin. If omitted, country code assigned to the account will be used.
list	Lists all inbound numbers associated with the given account.
info	Retrieves information and supported file formats for the specified number in {id}.
enable disable	Enables or disables the route specified in {id}. Only numbers/routes owned by the account may be enabled/disabled using this function. If the route is not found, a 204 result is returned.
id	Specifies the phone number or route/endpoint to analyze.

Example:

[GET] https://na.connect.etherfax.net/rest/2.0/api/routes?a=parse&id=8005551234

Response:


```
{
  "InternationalDialPrefix": "011",
  "Normalized": "8005551234",
  "Origin": "+1/United States/Canada",
  "E164": "+18005551234",
  "Destination": "+1/United States/Canada",
  "Route": "8005551234"
}
```

Example

[GET] https://na.connect.etherfax.net/rest/2.0/api/routes?a=info&id=8005551234

Response

```
{
  "Route": "+18005551234",
  "AcceptedFormats": [
    "image/tiff",
    "application/pdf"
  ]
}
```

Example

[GET] https://na.connect.etherfax.net/rest/2.0/api/routes?a=enable&id=8005551234

Response

200/OK
204/Not Found

Devices Controller

The devices controller provides the required functions that allow customer managed devices to register with the etherFAX network securely, send and receive fax documents as well as manage specific attributes and features associated with each device.

The functions made available are functionally similar to the account level REST functions, but are specific to the scope (or perspective) of the device. For instance, requesting a list of unread faxes will only yield results for the specific device in question. Inbound DID/routes dictate which device a particular fax will be bound to upon arrival within the etherFAX network.

Uri Template:

```
[GET] "devices?a={action}&id={id}&addr={addr}&token={token}"
```

Parameters:

The following actions are supported for the devices controller.

Parameter	Description
register	Performs a device registration to the etherFAX network and returns a security token used for subsequent authentication. Until a device has been created/enabled within the etherFAX network, this function cannot be completed successfully. Devices already registered within the system may not be re-registered until reset within the etherFAX administration portal.
unread	Returns the number of unread faxes waiting for the associated device. Since the default device is selected as part of the authentication request, the {id} parameter is not required. This function behaves similar to the inbox controller, except an additional *action* value is provided in the response.
actionreset	Though the device's action is implicitly reset when the "unread" action is invoked, the remote client may explicitly call this function to clear the action state of the device.
list	Returns a list of unread faxes associated with the specified device.
status	Returns the current status of the device including properties and features associated with the device. This should be invoked is the action response is greater than 0.
setaddr	Allows the remote client to set the local IP address of the device. This function serves no other purposes than to report the current IP address of the device that may be viewed within the etherFAX administrative portal.
id	Specifies the unique serial number of the device.

Example:

[GET] https://na.connect.etherfax.net/rest/2.0/api/devices?a=unread

Response:

```
{
  "UnreadFaxes": 0
  "Action": 0
}
```

Example

[GET] https://na.connect.etherfax.net/rest/2.0/api/devices?a=status

Response

```
{
  "SerialNumber": "MT873615726",
  "Enabled": true,
  "SendEnable": true,
  "ReceiveEnable": true,
  "Reporting": 0,
  "SkipLines": 45,
  "LocalId": null,
  "CallerId": null,
  "PrinterIp": "192.168.1.100",
  "DeviceFlags": 3,
  "CreatedOn": "2016-09-19T17:15:38.707",
  "RegisteredOn": "2016-12-29T14:18:56.443",
  "TimeZone": "America/New_York"
}
```

Registering A Device

Before a device can operate within the etherFAX network, it must be created as a customer device and enabled for operation. To begin the registration process, each device must have the current date/time set correctly and create a registration URI following these steps.

In this example, our device serial number is: MT873615726

Step 1: Create initial MD5 hash using the serial number as input.

Result: 3239f210aefade4a5de84cd018d18b44

Step 2: Create temporary string using current date/time and initial hash.

"YYYY:MM:*hash*.DDD"

Where YYYY = year since 1900, MM = current month (0..11), hash (lower case) and DDD is the current day of the year (0..365).

Example: Feb 2, 2017

"0117:01:3239f210aefade4a5de84cd018d18b44.032"

Step 3: Perform final MD5 hash of temporary string.

Result: 414d6d04931da79f6d9925e0c30b85a3

Example:

[GET]
https://na.connect.etherfax.net/rest/2.0/api/devices?a=register&id=MT873615726&token=414d6d04931da79f6d9925e0c30b85a3

Response:

```
{
  "DeviceToken": "54524a6421f1ddaf6781c79f9e176cfc"
}
```

On success, an HTTP 200/OK response and device security token will be returned. If the device is not enabled or has been previously registered, an HTTP 403/Forbidden is issued.

Once a DeviceToken has been created, the device may authenticate against the etherFAX REST using the same basic authorization semantics. However, the user name will must be replaced with the literal string of "device/" followed by the device serial number. The password is the DeviceToken returned from the register action. After the device security token is issued, it is the device's responsibility to persist the security token for later use.

Device Actions

Though most of the following actions are reserved for embedded device platforms (such as the etherFAX A2E), device developers should at least support UpdateConfig as a baseline requirement. This allows all devices to be remotely managed from the etherFAX administration portal.

```
/// <summary>
/// DeviceAction values.
/// </summary>
public enum DeviceAction
{
      UpdateConfig = 1,
      Ping,
      Reboot,
      Unregister,
      UpdateFirmware,
      UploadLogs
}
```

FaxResult and FaxState Values

The following constants are used to describe the various fax results (disposition) and states of transmission.

FaxResult values:

```
/// <summary>
/// FaxResult values.
/// </summary>
public enum FaxResult
{
	Success = 0,
	Error,
	InProgress,
	LineBusy,
	LineDead,
	LineFailure,
	NoDialTone,
	NoAnswer,
	InvalidOrMissingNumber,
	InvalidOrMissingFile,
	InvalidChannel,
	UnexpectedDisconnect,
	NoChannelsAvailable,
	ChannelUnavailable,
	NothingToCancel,
	DeviceTimeout,
	DeviceBusy,
	NotFaxMachine,
	IncompatibleFaxMachine,
	FileError,
	FileNotFound,
	FileUnsupported,
	CallCollision,
	Cancelled,
	CallBlocked,
	DestinationBlackListed,

	Unauthorized = 100,
	InvalidParameter,
	NotImplemented,
	ItemNotFound,
}
```

FaxState values:

```
/// <summary>
/// FaxState values.
/// </summary>
public enum FaxState
{
	Idle = 0,
	Initializing,
	Dialing,
	Answering,
	Negotiating,
	Sending,
	Receiving,
	Cancelling,
	Disconnecting,
	Conversion
}
```

Date/Time Formatting Values

The following parameters may be used when formatting the fax header.

Parameter	Description	Examples
"d"	The day of the month, from 1 through 31. More information: The "d" Custom Format Specifier.	6/1/2009 1:45:30 PM -> 1 6/15/2009 1:45:30 PM -> 15
"dd"	The day of the month, from 01 through 31. More information: The "dd" Custom Format Specifier.	6/1/2009 1:45:30 PM -> 01 6/15/2009 1:45:30 PM -> 15
"ddd"	The abbreviated name of the day of the week. More information: The "ddd" Custom Format Specifier.	6/15/2009 1:45:30 PM -> Mon (en-US) 6/15/2009 1:45:30 PM -> Пн (ru-RU) 6/15/2009 1:45:30 PM -> lun. (fr-FR)
"dddd"	The full name of the day of the week. More information: The "dddd" Custom Format Specifier.	6/15/2009 1:45:30 PM -> Monday (en-US) 6/15/2009 1:45:30 PM -> понедельник (ru-RU) 6/15/2009 1:45:30 PM -> lundi (fr-FR)
"h"	The hour, using a 12-hour clock from 1 to 12. More information: The "h" Custom Format Specifier.	6/15/2009 1:45:30 AM -> 1 6/15/2009 1:45:30 PM -> 1
"hh"	The hour, using a 12-hour clock from 01 to 12. More information: The "hh" Custom Format Specifier.	6/15/2009 1:45:30 AM -> 01 6/15/2009 1:45:30 PM -> 01
"H"	The hour, using a 24-hour clock from 0 to 23. More information: The "H" Custom Format Specifier.	6/15/2009 1:45:30 AM -> 1 6/15/2009 1:45:30 PM -> 13

"HH"	The hour, using a 24-hour clock from 00 to 23. More information: The "HH" Custom Format Specifier.	6/15/2009 1:45:30 AM -> 01 6/15/2009 1:45:30 PM -> 13
"m"	The minute, from 0 through 59. More information: The "m" Custom Format Specifier.	6/15/2009 1:09:30 AM -> 9 6/15/2009 1:09:30 PM -> 9
"mm"	The minute, from 00 through 59. More information: The "mm" Custom Format Specifier.	6/15/2009 1:09:30 AM -> 09 6/15/2009 1:09:30 PM -> 09
"M"	The month, from 1 through 12. More information: The "M" Custom Format Specifier.	6/15/2009 1:45:30 PM -> 6
"MM"	The month, from 01 through 12. More information: The "MM" Custom Format Specifier.	6/15/2009 1:45:30 PM -> 06
"MMM"	The abbreviated name of the month. More information: The "MMM" Custom Format Specifier.	6/15/2009 1:45:30 PM -> Jun (en-US) 6/15/2009 1:45:30 PM -> juin (fr-FR) 6/15/2009 1:45:30 PM -> Jun (zu-ZA)
"MMMM"	The full name of the month. More information: The "MMMM" Custom Format Specifier.	6/15/2009 1:45:30 PM -> June (en-US) 6/15/2009 1:45:30 PM -> juni (da-DK) 6/15/2009 1:45:30 PM -> uJuni (zu-ZA)
"s"	The second, from 0 through 59. More information: The "s" Custom Format Specifier.	6/15/2009 1:45:09 PM -> 9
"ss"	The second, from 00 through 59.	6/15/2009 1:45:09 PM -> 09

	More information: The "ss" Custom Format Specifier.	
"t"	The first character of the AM/PM designator. More information: The "t" Custom Format Specifier.	6/15/2009 1:45:30 PM -> P (en-US) 6/15/2009 1:45:30 PM -> 午 (ja-JP) 6/15/2009 1:45:30 PM -> (fr-FR)
"tt"	The AM/PM designator. More information: The "tt" Custom Format Specifier.	6/15/2009 1:45:30 PM -> PM (en-US) 6/15/2009 1:45:30 PM -> 午後 (ja-JP) 6/15/2009 1:45:30 PM -> (fr-FR)
"y"	The year, from 0 to 99. More information: The "y" Custom Format Specifier.	1/1/0001 12:00:00 AM -> 1 1/1/0900 12:00:00 AM -> 0 1/1/1900 12:00:00 AM -> 0 6/15/2009 1:45:30 PM -> 9
"yy"	The year, from 00 to 99. More information: The "yy" Custom Format Specifier.	1/1/0001 12:00:00 AM -> 01 1/1/0900 12:00:00 AM -> 00 1/1/1900 12:00:00 AM -> 00 6/15/2009 1:45:30 PM -> 09
"yyy"	The year, with a minimum of three digits. More information: The "yyy" Custom Format Specifier.	1/1/0001 12:00:00 AM -> 001 1/1/0900 12:00:00 AM -> 900 1/1/1900 12:00:00 AM -> 1900 6/15/2009 1:45:30 PM -> 2009
"yyyy"	The year as a four-digit number. More information: The "yyyy" Custom Format Specifier.	1/1/0001 12:00:00 AM -> 0001 1/1/0900 12:00:00 AM -> 0900 1/1/1900 12:00:00 AM -> 1900 6/15/2009 1:45:30 PM -> 2009