KBase Auth 0.2

RSA OAuth for those with ADD

Steve Chan
sychan@lbl.gov

Globus Online

- Globus Online operates a service that is virtually identical to the original KBase Auth Spec
  - OAuth 2.0 compliant
  - REST based backend service for user profiles and groups
  - Supports interactive and programmatic interfaces
  - Supports delegation
- Tied into Globus Online file transfer service

Trade Offs

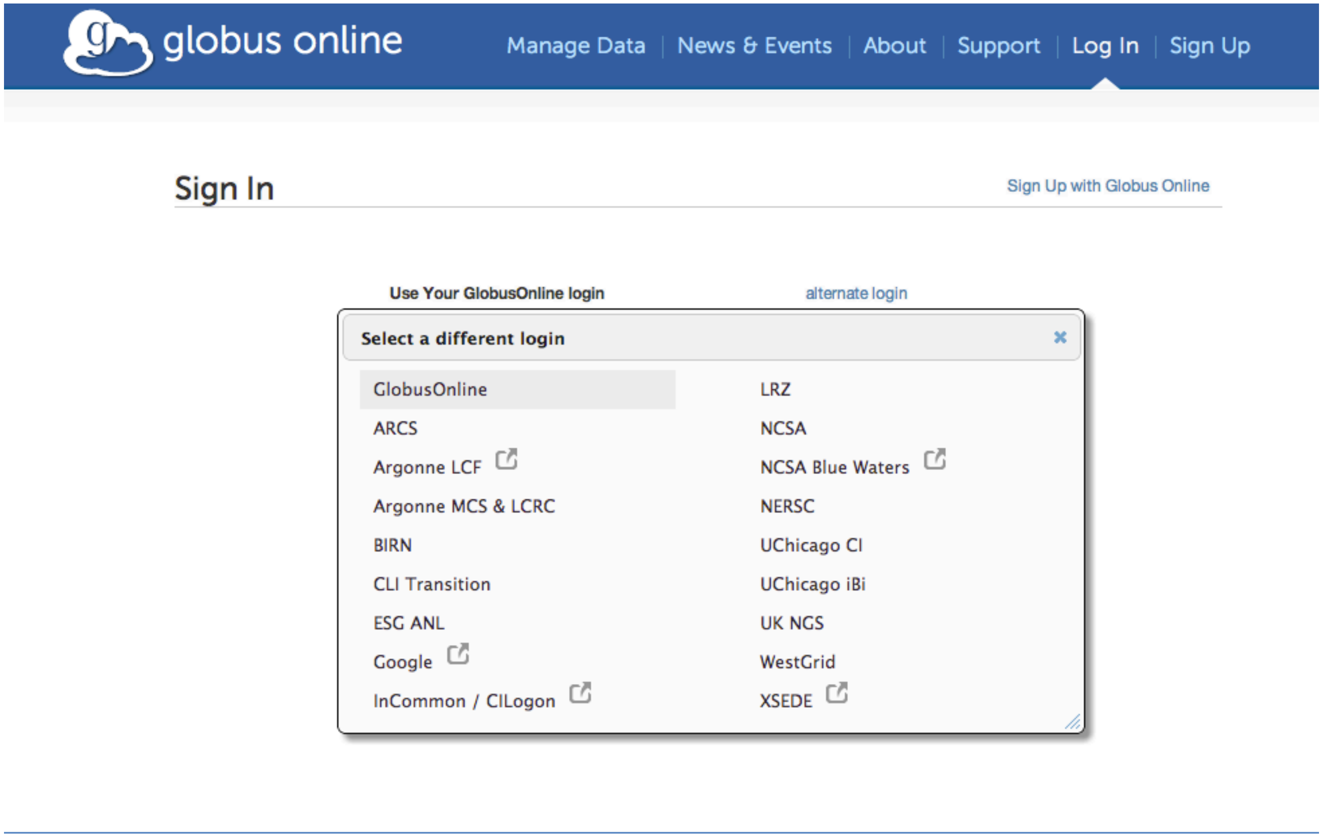
Benefits

- Service is already in operation and has production support
  - User registration service is done
  - OAuth 2.0 support in place
  - More developers and users banging on code, converges faster on stable high performance authentication service
  - Globus Online will provide custom skinned login instance at appropriate URL
- Instant tie-in with Globus Online high performance data transfer service
- Federated logins – users with accounts in the inCommon Federation have "single sign-on", linking their home institution accounts into Globus (more useful for web based apps)

Drawbacks

- No longer in control of user profiles – we are a client, not the master
  - We control authorization via groups, instead of directly manipulating accounts
- We are dependent on another group for feature changes on backend
  - The Globus Online guys are mostly ANL guys, and the ANL KBase guys have ready access to them

Globus Online:
Alternate Login Screen



Sign In

Sign Up with Globus Online

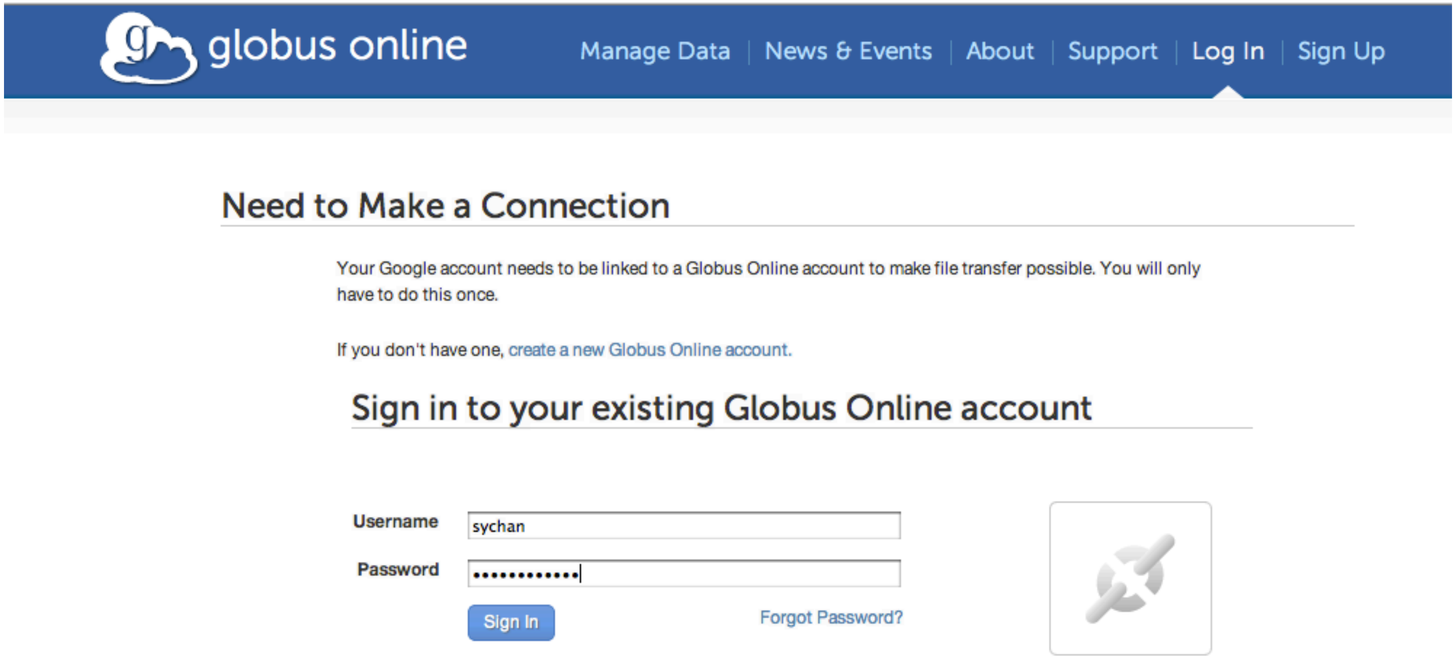
Use Your GlobusOnline login

alternate login

Select a different login ×

GlobusOnline	LRZ
ARCS	NCSA
Argonne LCF	NCSA Blue Waters
Argonne MCS & LCRC	NERSC
BIRN	UChicago CI
CLI Transition	UChicago iBi
ESG ANL	UK NGS
Google	WestGrid
InCommon / CILogon	XSEDE

Globus Online: Associating Remote Account



globus online
Manage Data | News & Events | About | Support | Log In | Sign Up

Need to Make a Connection

Your Google account needs to be linked to a Globus Online account to make file transfer possible. You will only have to do this once.

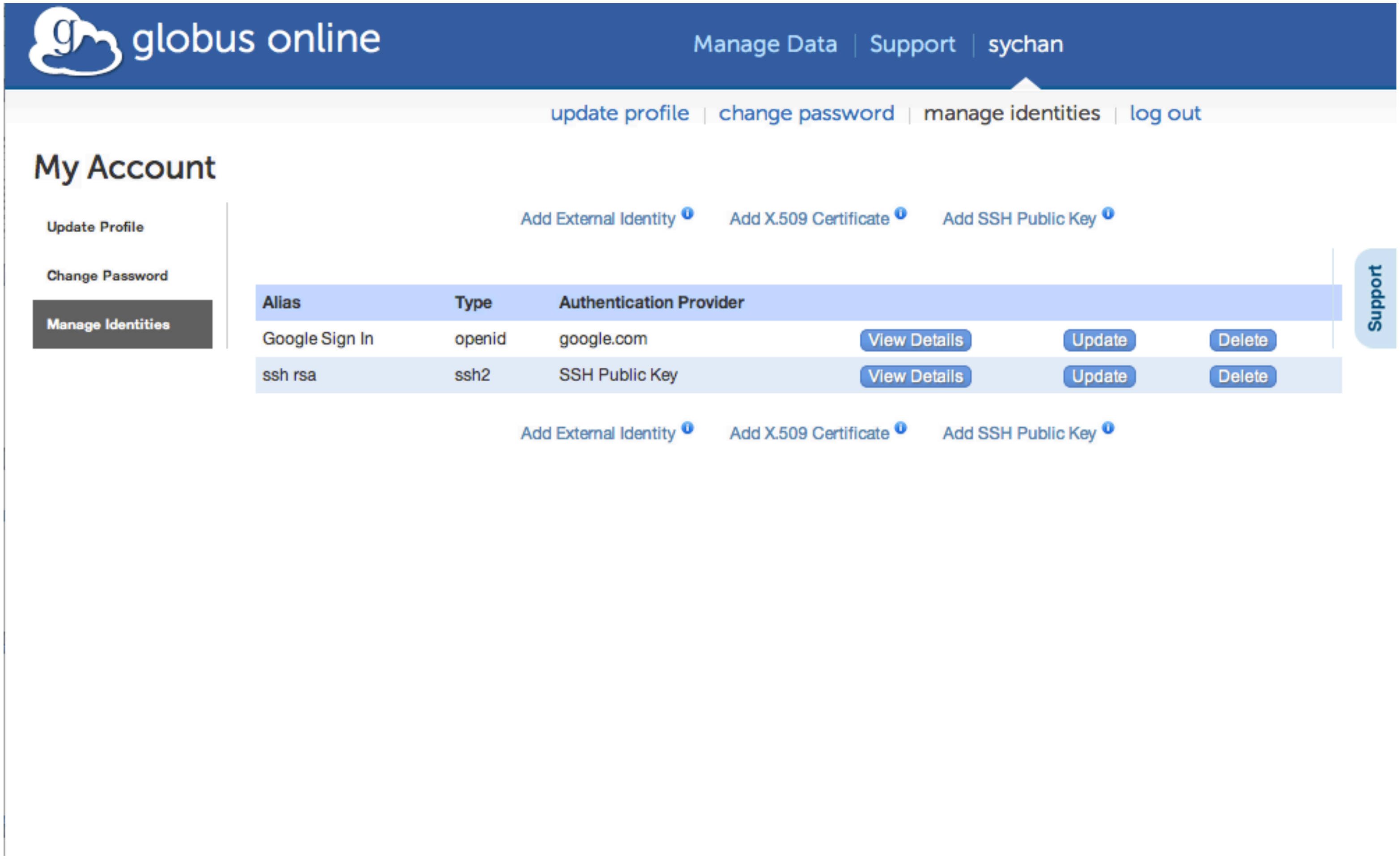
If you don't have one, create a new Globus Online account.

Sign in to your existing Globus Online account

Username: sychan

Password: ••••••••••••

Globus Online:
Account Profile Screen



globus online
Manage Data | Support | sychan
update profile | change password | manage identities | log out

My Account

Alias	Type	Authentication Provider			
Google Sign In	openid	google.com	View Details	Update	Delete
ssh rsa	ssh2	SSH Public Key	View Details	Update	Delete

Consequences of Globus Online

- We have outsourced identity to Globus Online
- We manage authorization internally
- We can create service accounts within Globus Nexus
- Tokens are “bearer tokens”
  - Tokens are not tied to URL, and possession is considered proof of identity
- Tokens are acquired through several forms of credentials
  - RSA signature authentication using SSH public keys
  - Normal passwords
  - X509 client certificates possible (not currently supports in Kbase libraries)

Revision 0.2 of KBase Auth Libraries

- Backend Django service replaced by Globus Online REST service
- Client API refactored based on observations
  - Developers only really seemed to be interested in working with tokens and getting user profiles
  - Abstraction based on Client side vs Server side was too heavyweight
- Tokens are signed using RSA public keys
  - No more shared secrets

Globus Online: Important URLs

Web interface to test and production instances:
https://test.globuscs.info/ (test)
https://www.globusonline.org/ (production)
REST API access:
https://graph.api.test.globuscs.info/ (test)
https://graph.api.globusonline.org/ (production)
REST API documentation:
http://globusonline.github.com/nexus-docs/api.html
Globus Nexus Python Client Git Repo:
https://github.com/globusonline/python-nexus-client.git
KBase Auth Code Git Repo:
https://github.com/kbase/auth.git
Java code to validate Globus Tokens:
ssh://kbase@git.kbase.us/persistent_store.git

persistent_store/DocumentStoreREST/src/Auth

Globus Online:
Command Line Fun

Acquiring an authentication token

```
curl -k -# --user papa:papa 'https://
  graph.api.test.globuscs.info/goauth/authorize?
  response_type=code&client_id=papa' | python -m
  json.tool
########################################################
  ################# 100.0%
{
    "code": "un=papa|clientid=papa|expiry=1376522845|
  SigningSubject=https://graph.api.test.globuscs.info/
  goauth/keys/861eb8e0-e634-11e1-ac2c-1231381a5994|
  sig=3fb54b56c7bb9a5c13ea0027e24118dca5b9805a7e401982e
  4def932a6d78aec34800582c0233da814b57bf438022b0581f013
  c3aded56bada7479b476c1d67c8aff1dc05627473255a97e93cd7
  e13e8add6dbdf7f8b23f88b66681fa119a347b8750a33596ae7ca
  74cfc6355e8e5ba4e32137379abe17fbedeb4520d351315b",
    "state": null
}
```

Globus Online:
Command Line Fun

Token Structure

```
un=papa|
clientid=papa|
expiry=1376522845|
SigningSubject=https://
  graph.api.test.globuscs.info/goauth/keys/
  861eb8e0-e634-11e1-ac2c-1231381a5994|
sig=3fb54b56c7bb9a5c13ea0027e24118dca5b9805a7e4
  01982e4def932a6d78aec34800582c0233da814b57bf4
  38022b0581f013c3aded56bada7479b476c1d67c8aff1
  dc05627473255a97e93cd7e13e8add6dbdf7f8b23f88b
  66681fa119a347b8750a33596ae7ca74cfc6355e8e5ba
  4e32137379abe17fbedeb4520d351315b
```

Globus Online: Command Line Fun

Using authentication to fetch user profile

```
curl -k -# -H "X-GLOBUS-GOAUTHTOKEN: un=papa|clientid=papa|expiry=1376522845|
   SigningSubject=https://graph.api.test.globuscs.info/goauth/keys/861eb8e0-
   e634-11e1-ac2c-1231381a5994|
   sig=3fb54b56c7bb9a5c13ea0027e24118dca5b9805a7e401982e4def932a6d78aec34800582c
   0233da814b57bf438022b0581f013c3aded56bada7479b476c1d67c8aff1dc05627473255a97e
   93cd7e13e8add6dbdf7f8b23f88b66681fa119a347b8750a33596ae7ca74cfc6355e8e5ba4e32
   137379abe17fbedeb4520d351315b" 'https://graph.api.test.globuscs.info/users/
   papa' | python -m json.tool
######################################################################## 100.0%
{
    "custom_fields": {},
    "email": "papa@smurfs.nz",
    "email_validated": true,
    "fullname": "Papa Smurf",
    "opt_in": null,
    "system_admin": true,
    "username": "papa"
}
```

Bio::KBase::AuthToken

Getting a Token on the commandline for the ADD:

Username/Password

```
sychan$ perl -MBio::KBase::AuthToken -e 'print Bio::KBase::AuthToken->new
   ( user_id => "papa", password => "papa")->token,"\n";'
un=papa|clientid=papa|expiry=1376542920|SigningSubject=https://
   graph.api.test.globuscs.info/goauth/keys/861eb8e0-e634-11e1-
   ac2c-1231381a5994|
   sig=2e81ffa18b1803588228f475d35ae9d8c1b5c582a46dbbb49d7944662dc349791b6b
   0ac5e4a6b89c406de1dc83eff1a961cb9892a45398636809fce4ff7f70b2a76a384cd336
   1a4f6bdd25d72b1affb413717abf6b499ecffff19037cce7fd2367135c3e736398c1c26d
   429ecb827635a3aae8a92c7beae1c87760df958eecb2
```

Username/RSA

```
sychan$ perl -MBio::KBase::AuthToken -e '$key=`cat ~/.ssh/id_rsa`; print
   Bio::KBase::AuthToken->new( user_id => "sychan", client_secret => $key)-
   >token,"\n";’
un=sychan|clientid=sychan|expiry=1376543282|SigningSubject=https://
   graph.api.test.globuscs.info/goauth/keys/861eb8e0-e634-11e1-
   ac2c-1231381a5994|
   sig=afa6f7ad6389f3775a1142cc16b9f223e299d0e75e154182731e8405167ac447281b
   a70b6eee97939276463f71ecdb466f65ab70f6b96ebec9666e443b81fb795eace1b9fe1d
   05a055edb6afb623bbf601330f2363db0701c82656de81a10e58fa0c955b4794e08c0407
   a48771e1c660ad78cd2d0338c6b4a72e514892d32f22
```

Bio::KBase::AuthToken
Less ADD

```
use Bio::KBase::AuthToken;
use Bio::KBase::AuthUser;
use Data::Dumper;

$t = Bio::KBase::AuthToken->new( user_id => "papa",
                  password => "papa");
if ($t->validate()) {
   print "Yeehaw!!! It works!\n";
   print Dumper( $t);
   $u = Bio::KBase::AuthUser->new( token => $t->token);
   if ($t->user_id()) {
      print "And here's my profile object:\n";
      print Dumper( $u);
   }
} else {
   die "Doh!\n";
}
```

Bio::KBase::AuthToken

```
Yeehaw!!! It works!
$VAR1 = bless( {
                 'password' => 'papa',
                 'error_message' => undef,
                 'user_id' => 'papa',
                 'token' => 'un=papa|clientid=papa|expiry=1376544425|SigningSubject=https://
     graph.api.test.globuscs.info/goauth/keys/861eb8e0-e634-11e1-ac2c-1231381a5994|
     sig=805ab020da27e5b77416ce8e76431cd5941a1a2c527ef3c89ff04a4b2a2de1fa94e7e40517364224b89aeec19269aabb9
     60435a6d144b7911ee62801b2143c5061808504f9815930dd7f55150cbdfec7203331466c744b9dd62d674d459b4efbd39f2f
     f2d3249ae8f2d4eb41d17d5263ae6ddd8a43bed0a555005c2edc299076'
               }, 'Bio::KBase::AuthToken' );
And here's my profile object:
$VAR1 = bless( {
                 'system_admin' => 1,
                 'name' => 'Papa Smurf',
                 'phone' => '8888888888',
                 'email' => 'papa@smurfs.nz',
                 'oauth_creds' => {
                                    'auth_token' => 'un=papa|clientid=papa|expiry=1376544425|
     SigningSubject=https://graph.api.test.globuscs.info/goauth/keys/861eb8e0-e634-11e1-ac2c-1231381a5994|
     sig=805ab020da27e5b77416ce8e76431cd5941a1a2c527ef3c89ff04a4b2a2de1fa94e7e40517364224b89aeec19269aabb9
     60435a6d144b7911ee62801b2143c5061808504f9815930dd7f55150cbdfec7203331466c744b9dd62d674d459b4efbd39f2f
     f2d3249ae8f2d4eb41d17d5263ae6ddd8a43bed0a555005c2edc299076'
                                  },
                 'error_message' => undef,
                 'current_project_name' => 'Papa',
                 'verified' => 1,
                 'user_id' => 'papa',
                 'token' => undef,
                 'organization' => 'papa',
                 'institution' => 'papa',
                 'opt_in' => undef
               }, 'Bio::KBase::AuthUser' );
```

Sample Client and Server Code: “main()”

```
$d = HTTP::Daemon->new( LocalAddr => '127.0.0.1');

printf "Server listening at %s\n",$d->url;

my $child = fork();
if ($child) {
   testClient( $d->url);
} else {
   testServer( $d);
}
```

Sample Client and Server Code: testServer()

```
sub testServer {
    my $d = shift;
    my $res = new HTTP::Response;
    my $msg = new HTTP::Message;
    my $at = new Bio::KBase::AuthToken;
    while (my $c = $d->accept()) {
        while (my $r = $c->get_request) {
            my $token = $r->header('Authorization');
            $at->token( $token);
            if ($at->validate()) {
                $res->code(200);
                $body = sprintf( "Successfully logged in as user %s\n", $at->user_id);
                $au = Bio::KBase::AuthUser->new( 'token' => $at->token);
                printf "Server: User %s (%s) found for token %s.\n", $au->name,$au->user_id, $at->token;
            } else {
                $res->code(401);
                printf "Server: token failed validation.\n";
                $body = sprintf("You failed to login: %s\n", $at->error_message);
            }
            $res->content( $body);
            $c->send_response($res);
        }
        $c->close; } }
```

Sample Client and Server Code: testClient()

```
sub testClient {
    my $server = shift;
    my $ua = LWP::UserAgent->new();
    my $req = HTTP::Request->new( GET => $server. "someurl" );

    $at = Bio::KBase::AuthToken->new('user_id' => 'kbasetest', 'password' =>
    '@Suite525');
    $req->header("Authorization" => $at->token);

    $res = $ua->request( $req);
    printf "Client: Recieved a response: %d %s\n", $res->code, $res->content;

    # As a sanity check, trash the token and make sure that
    # we get a negative result
    $req->header("Authorization" => "bogo token");

    printf "Client: Sending bad request: %s %s (expecting failure)\n",$req-
    >method,$req->url->as_string;
    $res = $ua->request( $req);
    printf "Client: Recieved a response: %d %s\n", $res->code, $res->content;

}
```

Sample Client and Server Code: output

```
Server listening at http://127.0.0.1:54079/
Server: User KBase Test Account (kbasetest) found for token
   un=kbasetest|clientid=kbasetest|expiry=1376547165|
   SigningSubject=https://graph.api.test.globuscs.info/goauth/
   keys/861eb8e0-e634-11e1-ac2c-1231381a5994|
   sig=9a8fcd2a41e877aec5b0ed5a0d0315c258b5466d4862d17b5ce33f2
   703d2a32e8179556579998a8379271360251593b9db86430fb9bd1b9bf0
   81b39dc6ff978bde834a9a9cb19eacc46f0b4ada93048c10b8aa2bfa232
   5ba5fab6993042b51bc7e3ab1545831c261e96972f3789fd20b6c1c5742
   f937ffda8a6f08c3019ca0d1.
Client: Recieved a response: 200 Successfully logged in as
   user kbasetest

Client: Sending bad request: GET http://127.0.0.1:54079/
   someurl (expecting failure)
Server: token failed validation.
Client: Recieved a response: 401 You failed to login: Failed
   to verify token: Token lacks signature fields at lib/Bio/
   KBase/AuthToken.pm line 237.
```

Token Validation Plumbing

Sample Token:

```
un=kbasetest|clientid=kbasetest|expiry=1376547165|
   SigningSubject=https://
   graph.api.test.globuscs.info/goauth/keys/861eb8e0-
   e634-11e1-ac2c-1231381a5994|
   sig=9a8fcd2a41e877aec5b0ed5a0d0315c258b5466d4862d1
   7b5ce33f2703d2a32e8179556579998a837927136025l593b9
   db86430fb9bd1b9bf081b39dc6ff978bde834a9a9cb19eacc4
   6f0b4ada93048c10b8aa2bfa2325ba5fab6993042b51bc7e3a
   b154583lc261e96972f3789fd20b6c1c5742f937ffda8a6f08
   c3019ca0d1
```

Token Contents = Green

Token Signature = Blue

Token Validation Plumbing

From sample token:

```
SigningSubject=https://graph.api.test.globuscs.info/
  goauth/keys/861eb8e0-e634-11e1-ac2c-1231381a5994

SigningSubject points to JSON object:
{
    "expiry": 1345569705.0,
    "id": "861eb8e0-e634-11e1-ac2c-1231381a5994",
    "pubkey": "-----BEGIN RSA PUBLIC KEY-----
  \nMIGJAoGBAOI4NghMHlbChfx07ZeLY4ei0iJUb9ofnv+cnFOFEmP
  +BrxZObT3RpuD
  \nC2jjIX78lj16LaQPxz5iaFth3FJCxERgcjnlQ5nvgHAwisRG
  +76Y8fIpgGAdWL5S\nzrlgDOXVieYs
  +N06fGPmlsMRtQVEzy0fUa5421epiNRKy5QYZskPAgMBAAE=\n-----
  END RSA PUBLIC KEY-----\n",
    "valid": true
}
```

Token Validation Plumbing

Sample Token:

```
un=kbasetest|clientid=kbasetest|expiry=1376547165|
   SigningSubject=https://graph.api.test.globuscs.info/
   goauth/keys/861eb8e0-e634-11e1-ac2c-1231381a5994|
   sig=9a8fcd2a41e877aec5b0ed5a0d0315c258b5466d4862d17b5ce3
   3f2703d2a32e817955657999 8a8379271360251593b9db86430fb9bd
   1b9bf081b39dc6ff978bde834a9a9cb19eacc46f0b4ada93048c10b8
   aa2bfa2325ba5fab6993042b51bc7e3ab1545831c261e96972f3789f
   d20b6c1c5742f937ffda8a6f08c3019ca0d1
```

```
If (RSA_SHA1_Base64(Green Stuff, pubkey) == Blue Stuff) {
  # Happiness!! ☺
} else {
  # Sucks ☹
}
```

Caveats

- OAuth 2.0 compliant …but…
  - OAuth 2.0 bearer token structure undefined
  - The Globus Online bearer tokens are custom and use our own validation scheme
  - Focus has been on REST interface for authentication, have not pushed on full OAuth browser flows
    web interfaces with callbacks not yet implemented
- Globus Online has assigned a hierarchy of groups rooted at “kbase”
  - See groups API at http://globusonline.github.com/nexus-docs/api.html#groups
  - Create an account on Globus, and we can assign admin privs on that tree to that account

Issues: We Got Issues!

- Globus provides groups for authorization
  - How do we want to use groups?
  - What authorization model do we use?
    - Complex policies or something simple like POSIX ACLs
  - Where do we put the Kbase Authz service?
    - We have MongoDB instance. Put a REST interface on it and run with it?
- Idiosyncratic token format
  - Works okay for us, but have floated the idea of JSON Web Token to Globus folks, IETF track JSON token used by Google http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-03
- What else do we want from Globus in terms of features?
  - Need a KBase themed login page
  - What else?
- Do we want a session service?
  - Only accessible from inside KBase
  - Cache auth/authz and application session state across multiple requests to services