User Manual for ADPAR

(API for DDoS Prevention And Recovery)

Written by Ben Jones

**Contents:**

1. **Interface to the Web Service**
2. **Interface to the API**
3. **Best Practices**
4. **Interface to the Web Service**

To utilize ADPAR, your web service must implement hooks for several Ruby methods. These methods should be wrapped into a Ruby class, hereafter called a service object. The service object should be instantiated, and passed to the API’s constructor. If the hooks are not ready when the API is instantiated or if the service object needs to change, the service object can be set with the set\_service\_object method.

In order to receive data, several methods must be implemented in your service object. These methods are as follows:

1. get\_data(string data, int id)
   1. the variable data will be set to the exact string received on the connection and id will be set to the id of that connection
2. to\_yaml(nil)
   1. this method should take a backup of your web service and convert it into YAML
3. init\_fail(reference \* data)
   1. this method will take in an array of variables and have your web service use them to take over for another server. This array of variables will be the latest backup received for that connection.

In order to send data, your web service should call your instance of ADPAR’s send\_data method with your data. The specifications for how to use the send\_data method are in section 2 with the rest of the API’s interface.

1. **Interface to the API**

While there are a lot of functions in the API, this section will only cover the public methods. For complete coverage of every method and variable in the API, please see the API interface document. It should also be noted that every one of these methods is designed to be called as an instance method of the service object’s API object.

1. send\_data
   1. This method will be used to send data from the service to another server and returns the id of the connection. This id can be used with various methods to specify the connection to use.
   2. send\_data(string address, int port\_num, data)
      1. Should be used to send data to another server when the connection has yet to be opened
   3. send\_data(int connect\_id, data)
      1. should be used to send data to another server when the connection has already been opened
2. send\_backup
   1. this method will be used to send a backup from the local service to another server
   2. send\_backup(int id, string backup\_data)
      1. send a backup containing backup\_data to the server specified by id
   3. send\_backup(int id)
      1. send a backup of the current service state to the given server
   4. send\_backup(nil)
      1. checkpoint the server and send a copy of the current service state to all connections that the API is set up to send backups to
3. kill(nil)
   1. will suspend all network communication through every instance of the API on the local server
   2. can be reversed with unkill(nil)
4. **Best Practices**

ADPAR is designed to be used to help prevent a denial of service attack. To ensure the best functionality, a list of best practices and how to implement them has been provided below.

1. Create a service object that implements get\_data, to\_yaml, and init\_fail methods as outlined in the interface document.
2. Create an instance of the API using the call DDOS\_API.new(timeout, ip, port, object) where timeout is the number of seconds to wait before failing over, ip is a string with the local ip, port is an int with the local port, and object is a reference to the service object. More details can be found in the interface file.
3. Set up each service with a list of servers that can receive failovers from this server and a list of servers that are dependent upon the service using a text file. More details can be found in the interface file, but the file should contain a line with the API of the form “local\_ip local\_port fail\_rec monitor\_ip1 monitor\_port1 monitor\_ip2 monitor\_port2 fail\_to back\_ip1 back\_port1 back\_ip2 back\_ip2” . Where local\_ip and local\_port are the local connection info, fail\_rec are the connections to monitor, and fail\_to are the connections who will be a failover for this server.
4. After starting operation, the server should send data to other servers through the instance of the API using the send\_data method.
5. Data will be received via the get\_data method of the service object.
6. The service should regularly checkpoint itself using the send\_backup(nil) method of the API.