Protocol Compiler - Feature #21660

Feature # 21659 (New): Redesign the output from the Javascript generator.

Use class syntax for request, reply, and structs in Javascript generator

01/11/2019 10:02 AM - Richard Neswold

Status:	New	**Start date:**	01/11/2019
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Javascript Generator	**Estimated time:**	0.00 hour
Target version:	Protocol Compiler v1.2		

Description

class syntax

Change the Javascript output to use the class keyword syntax for messages. For example, the following source file:

```
request example {
  string name;
  int32 age;
}
```

generates:

```
var JUNK_request_example = function () {
    this.name = "";
    this.age = 0;
};

JUNK_request_example.prototype.marshal = function* () {
    yield* [83, 68, 68, 2, 81, 3, 20, 12, 130, 31, 103, 18, 245, 24, 81, 4];
    yield* [18, 147, 28];
    yield* PROTOCOL.m_string(this.name);
    yield* [18, 125, 39];
    yield* PROTOCOL.m_int(this.age);
};
```

With the new generator, it should look something like:

```
class JUNK_request_example {
    constructor () {
        this.name = "";
        this.age = 0;
    }

    *marshal() {
        yield* [83, 68, 68, 2, 81, 3, 20, 12, 130, 31, 103, 18, 245, 24, 81, 4];
        yield* [18, 147, 28];
        yield* PROTOCOL.m_string(this.name);
        yield* [18, 125, 39];
        yield* PROTOCOL.m_int(this.age);
    }
};
```

Using Inheritance

The code can use minimal inheritance features to logically tie all requests together and all replies together. Requests and replies would be derived from a simple request and reply base class, respectively, that only has a static method to unmarshal messages of that type.

In the case of the request base class, we would also add to its prototype property a reference to the reply base class' unmarshal method. Currently, when the ACNET Javascript code is given a message to send, it checks to see if the message object has a marshal() method and, if so, uses it to encode the message. The ACNET module could also use this extra property to automatically

unmarshal replies before delivering them to the associated callback.

```
// Start of Reply hierarchy.

class JUNK_replybase {
    static unmarshal(o) {
        ...
    }
}

// ... other Reply messages ...

// Start of Request hierarchy.

class JUNK_requestbase {
    constructor() {
        this.replyDecoder = JUNK_replybase.unmarshal;
    }

    static unmarshal(o) {
        ...
    }
}

class JUNK_request_example extends JUNK_requestbase {
    constructor () {
        super();
    }

    ...
};
```

Note the base class names don't have an underscore before the "base" portion of the name. This is intentional due to how we generate names; if a protocol had a message named "base", it would get mapped to the name PROTO_request_base, which would conflict with the automatically generated base class.

History

#1 - 01/11/2019 10:03 AM - Richard Neswold

- Assignee deleted (Richard Neswold)

No ready to claim this issue yet.

#2 - 01/11/2019 03:05 PM - Richard Neswold

- Description updated

Add more features after discussion with Beau.

#3 - 01/11/2019 03:08 PM - Richard Neswold

- Description updated

Fix typo.

#4 - 04/01/2019 03:08 PM - Richard Neswold

- Tracker changed from Support to Feature

#5 - 04/05/2019 12:49 PM - Richard Neswold

- Description updated

Rewrite inheritance example. This new layout doesn't require the use of `Object.defineProperty()`.