



Applied Systems Engineering, Inc.

Technical Note #36 DAC Request Optimization

Technical Note 37: Request Optimization

1. Introduction	3
2. Running 2.x DAC with version 3.0 GPT.....	Error! Bookmark not defined.
2.1 Building a 2.x version.....	Error! Bookmark not defined.
2.2 Defines.....	Error! Bookmark not defined.
2.3 GPT Data Types	Error! Bookmark not defined.
2.4 Copying version 3.0 files.....	Error! Bookmark not defined.
2.5 Upgrading to COMM version 3.0.....	Error! Bookmark not defined.
2.5.1 Migrating ReadComm	Error! Bookmark not defined.
2.5.2 Migrating WriteComm	Error! Bookmark not defined.
2.5.3 Supporting ControlComm	Error! Bookmark not defined.
2.6 Upgrading to DAC version 3.0.....	Error! Bookmark not defined.

1. Introduction

There exists a class of protocols that allow multiple DAC Requests to be transmitted as a single protocol request. Examples of protocols that support this feature include DNP 3.0, Telegyr 8979 and SES92. Currently the GPT provides no request optimization. This document defines a number of possible approaches that could be used.

2. Optimization Flag.

The first approach adds an optimization flag to a request. The flag would be implemented as follows:

```
REQUESTINPUT.Value.Optimize =0, no optimization, 1 optimization allowed.
```

When parsing a request stream the GPT would attempt to combine all requests with the optimize flag = TRUE into a single request transmitted to a remote RTU. If for example the request stream for DNP 3.0 was as follows:

```
Read Variables Request    All analogs, optimize = TRUE
Read Variables Request    All binary inputs, optimize = TRUE.
```

The GPT would combine the two requests into a single DNP read request to read two objects. Features of this flag are the following:

- **Protocol Support.** The flag is only processed by protocols that can optimize requests. Protocols that can't ignore the flag and process the requests as they currently do.
- **Optimization Rules.** Each protocol implements its own optimization rules. Typically reads cannot be combined with variable writes, but if a protocol existed that allowed this combination it would be optimized.

There is a burden on the user in the above approach. The request stream must be ordered such that the optimizer can work. Interleaving requests to different devices would prevent the optimizer from working.

3. MACRO Requests.

The second approach allows the user to build a single request that is the combination of one or more individual requests. The DAC REQUESTINPUT structure would be modified as follows:

```
DAC_REQ_MACRO           ! Defines that a request is a MACRO
request.

typedef struct {

    DACFUNC             Function;
    RQVALUE             Request;
```

Technical Note 37: Request Optimization

```
} RQDEF;
```

The request object structure above can be used as an argument to a macro request. To read two objects analog inputs and binary input for DNP 3.0 the following request would be build.

```
RQDEF                Def[2];
REQUESTINPUT        Rq;

Def[0].Function = DAC_REQ_READ_VARIABLES;
Def[0].Object = DNP_BINARY_INPUT;
Def[0].Qualifier = DAC_REQ_QUALIFIER_ALL;

Def[1].Function = DAC_REQ_READ_VARIABLES;
Def[1].Object = DNP_ANALOG_INPUT;
Def[1].Qualifier = DAC_REQ_QUALIFIER_ALL;

Rq.Function = DAC_REQ_MACRO;
Rq.Qualifier = DAC_REQ_QUALIFIER_LIST;
Rq.Count = 2;
Rq.Args = &Def;

/* Activate request */
DacRequestActivate()
```

Technical Note 37: Request Optimization