White Paper

Realizing the OpenADR Demand and Response specification using SIP

Arjun Roychowdhury
co-chair, SIP Forum SmartGrid Group
Hughes Systique Corporation
arjun@hsc.com

Table of Contents

1.0 INTRODUCTION

A Demand Response (DR) program is a critical component, typically developed and offered by Utility companies/ISOs that offer participants to contribute effectively for better energy load management/ reduction. Typical participants include individuals as well as small and large corporations. A DR program typically ties in with a dynamic pricing scheme for electricity where participants, depending on a variety of factors (such as time of day, price etc.) can actively participate in requesting a Utility for increase or reduction of electricity demand. While this helps in cost reduction for participants, it significantly helps Utility/ISOs reduce their own cost and manage distribution of electricity (which is a finite resource) better. Several Utilities/ISOs offer incentive programs to participants for taking part in DR programs . In addition to DR, *automation* of DR is a key concept which helps reduce human intervention and increases accuracy and responsiveness to the DR program.

1.1 The state of DR today

Various forms of DR already exist with various Utilities. For various reasons, till date, DR involves a lot of manual intervention due to which many Utilities have implemented very basic DR programs.

The core goal for a Utility till date has been usage of DR to 'reduce' peak load. Direct Load Control (DLC) is a simple way to achieve this. Using DLC, a Utility can issue remote commands to the participating customer premise to 'shut off Air-Conditioning' or 'shut off water-pump' etc. for a brief period of time till the peak load demand decreases. There are more complex programs, which are typically made available to industrial and commercial customers where participants can elect to participate or not, based on the current advertized price etc. 'Participation' can be forced, or voluntary. The former means the customer does not have a choice, while the latter means that the customer can choose to participate or not in the DR program. Utilities incentivize customers to participate by offering incentives (in addition to the obvious benefit of a reduced energy bill). In return, there can be penalties if a participating customer does not shed load when peak load has been reached at the utility. As mentioned earlier, from the Utility perspective, there are two key goals for DR:

- Reduce peak load
- Increase reliability (this is really a corollary of the above)

Traditionally, many components of DR have been a manual process. Usually, an industrial administrator would monitor DR events and manually react to peak pricing indications. For enterprises which did not have dedicated personnel, there are several aggregators who liaison between the Utility and the enterprise customer and perform such monitoring and when system peak load reaches close to critical peak, they inform the enterprise customers to start shedding load. While there is a degree of automation here, this too involves significant manual intervention. Often, indications are delivered to customers over phone calls or even in-person visits. Furthermore, there is no widely followed standard in DR which is widely used. Therefore, any automation, however small is mostly proprietary.

1.2 Benefits of automation and open standards

A open-standards based automated DR system has several advantages, including:

- Reduce human error
- More features – if a system does not have an automated fashion on sending, processing and responding to situations, by definition, the goal would be to reduce the number of times or the complexity of features that are implemented in the system
- Better applications – following open standards ensures that innovative applications can quickly and robustly be created on top of existing DR platforms. This could be either by the incumbent vendor of the DR system or a 3rd party vendor

The rest of this paper will describe how SIP can be used as a very effective protocol and architecture to realize the OpenADR system.

It is important to state that this approach note describes SIP as an appropriate protocol and architecture to deliver and manage DR events. It does NOT specify the exact constraints/algorithms that different Utility/ISOs can use for different DR events. The latter is 'application logic' and is out of scope for SIP. SIP can work with any application logic that the Utility/ISOs decide to use.

Furthermore, this generic architecture covers both Managed Energy [Utility solely decides how/when to shed load] and Collaborative Energy models [Utility and participants decide together based on market dynamics]. From an architecture perspective the former model is a subset of the latter model with more constraints.

2.0 DEFINITIONS USED

Term	Meaning
DR	Demand and Response
Utility	Used to refer collectively to Utility Companies and ISOs

3.0 REFERENCES

1. Open Automated Demand Response Communication Standard, Public Draft Review 2008-Revision 2
2. SIP Specific Event Notification, RFC 3265
3. SIP Extension to Event State Notification, RFC 3903
4. CPL:A Language for User Control of Internet Telephony Services, RFC 3880
5. A Presence Event Package for SIP, RFC 3856
6. Interworking between SIP and XMPP, draft-saintandre-sip-xmpp-im-
7. Presence Information Data Format, PIDF, RFC 3863
8. Presence Authorization Rules, RFC 5025
9. DRA Overview of Demand an Response, Yulia Shmidt & Lisa-Marie Salvacion

10. 6lowapp group, http://trac.tools.ietf.org/area/app/trac/wiki/6LowApp

4.0 WHY SHOULD WE USE SIP FOR DEMAND AND RESPONSE?

SIP is mostly known as a very popular 'Session Establishment' protocol for VoIP. This often causes confusion about the applicability of SIP in anything that is not related to 'voice or video transmission'. What some do not know is that SIP has an extremely powerful Event Service model [2]. The IETF SIMPLE working group has designed an extensive set of specifications for a SIP based messaging and event service model which is actually deployed in almost all enterprises as well as wireless carriers that offer IP telephony. (In other words, it is very scalable and widely deployed). It is this event based model of SIP that we believe is ideal to realize the OpenADR specifications.

This chapter describes the benefits of using SIP as a protocol for DR realization. There are several benefits that SIP brings to the table which can significantly benefit management and executing of a DR program:

1. **Very powerful Push architecture:** SIP inherently supports a powerful event based architecture where participants can "subscribe" to the state of any event, as well as each other's state as well. This enables complex service possibilities (for example, the utility DRAS could choose to be a publisher of a DR event as well as a subscriber to another participant's state to get feedback). Furthermore, a push architecture is inherently superior to a pure pull model (which is typically how webservices APIswork) both in terms of performance and accuracy of data relevant to timing.

2. **Easy Firewall/NAT traversal:** There is a lot of confusion on firewall traversal and an application level protocol such as SIP. Very simply put, a problem occurs when someone outside of a secure firewall needs to communicate to someone inside the firewall on a *random* port. This is typically the case with dynamic media- the firewall will typically not allow dynamic port openings unless the entity behind that firewall first opens that connection. There are several ways around this, including opening an *inside* connection out and having responses piggybacking this connection to avoid firewall issues (like what XMPP does), using STUN, TURN, ICE etc. for automatic firewall traversal etc. **In the case of Demand and Response – Firewall traversal is NOT an issue**. **Furthermore, with SIP firewall traversal is a lesser problem than other protocols.** Here is why: Demand and Response is an event based control mechanism. The entire DR sequence can be transmitted and received using SIP's well known port: 5060 (UDP or TCP). No random port openings are required. Furthermore, **SIP is the defacto standard for any enterprise VoIP system. That means that SIP is already there and firewall ports are already opened.** If Demand and Response used the same infrastructure (which is what this paper is proposing), then firewall traversal is not an issue at all. If SIP were not used, IT departments will have to worry about provisioning another port for another protocol in their firewall.

3. **Highly Scalable:** SIP is an extremely scalable protocol. Reality proves this. All VoIP systems in any sized enterprise use SIP today. **All VoIP deployments in carrier networks are all SIP.** We are talking about millions of subscribers here. There can be no argument about this because the data is real and scalability for SIP is a proven fact.

4. **Customizable DR Event Package:** Using SIP's event architecture, each Utility could create new Event packages that serve specific needs of its DR management needs while at the same time, adhere to a 'base specification' which could be mandated as a minimal support set for interoperability between different systems

5. **Capability Based:** Another advantage of using SIP is that based on the capabilities of a participant, it could receive a subset of the complete DR attributes compared to others participating in the same program. This may be very useful, for example, if both residential and commercial units participate in the same DR program and the needs of a residential unit may be a small subset of the needs of an enterprise/commercial system. By specifying participant capabilities, both participants can be correctly served using the same DR event

6. **Multiple Events per participant:** SIP allows for a single participant to subscribe to multiple DR events at the same time. Furthermore, if required, these DR events could be treated separately or, the output of one could be used to change the input of another event the participant is subscribed to very easily

7. **Network Optimized:** SIP supports event aggregators as well as customizable per-participant, group or global event throttling that ensures network load can be reduced if required. Furthermore, SIP works well over both multicast and unicast transport.

8. **Partial notifications:** SIP allows a mechanism using which a notification can be sent out with only those parameters that have changed. This allows for significant bandwidth conservation when only a few attributes of a descriptive DR event has changed.

9. **Opt-out, Opt-in:** SIP Event system can be configured both as an Opt-out as well as Opt-in mechanism. In an Opt-out mode, participants can be configured to have implicitly subscribed to the event (or, have the participants subscribe on start). In the Opt-in mode, participants can send SUBSCRIBE messages to specifically opt-in and out.

10. **DR package discovery:** Using a generic SIP event package, it is very simple to implement a discovery mechanism where participants can query, or be notified when a new DR package that is applicable to that participant is available.

11. **Transport Secure:** SIP messages can be safely transmitted over the wire using TLS as well as a host of other security protocols

12. **User Security:** SIP supports many methods of ensuring user identity security including alias names that are mapped to specific network names, network asserted identity [xx] to ensure a user is who it claims to be and more.

13. **Participant Groups:** SIP allows for different participants to be grouped under common aliases. This allows for easy group based DR management.

14. **Common Identity for multimodal communication:** Since SIP supports multiple modes of operation: - Event based, transaction based and session based and all of these modes can be related using a common participant identity, it is very simple to invoke the other operation modes without losing context (example, if a particular participant in a DR event needs to be sent an sms or voice mail for a particular event, the same participant id can be used as an alias for other communication means – the SIP registrar can take care of resolving the generic alias to a specific URI)

15. 1 User to N device mapping: SIP supports the concept of associating a single 'user' to multiple devices. For example, sip:john@myhome.pepco.com may be the user identity of a Utility subscriber while sip:john@myhome.pepco.com;+sip.instance="pev" could be John's plugin electric car, sip:john@myhome.pepco.com;+sip.instance="thermostat" could be his home thermostat – and all these profiles can be associated together both from a billing perspective as well as share information with each other on DR bidding to provide analytics on total energy consumption and input for further bidding by this user.

16. **Reliability:** SIP is a very reliable protocol. It supports reliability in several ways including: routing messages via alternate intermediaries if one path fails, application level retransmission mechanisms to cover for transport layer retransmission failures, dynamically be able to locate current participant if mobile (based on most recently registered real URL)

17. **3rd party payload friendly:** SIP can easily carry multiple payload formats as part of its messages (XML, MIME, binary, etc.). The advantage of this model is SIP can offer the service subsystem all of the advantages sighted in this section while retaining the data format that makes sense for the payload

18. **Richer presence state:** Today, most DR systems primarily track if a DRAS client is ONLINE or OFFLINE. With rich presence payload support by SIP, a DRAS can act as a presence user agent that can report a much richer set of presence states beyond just on and off, like loaded etc.

5.0 TYPICAL DEMAND RESPONSE ARCHITECTURE

[2] specifies a generic architecture that describes a Demand Response system as below:

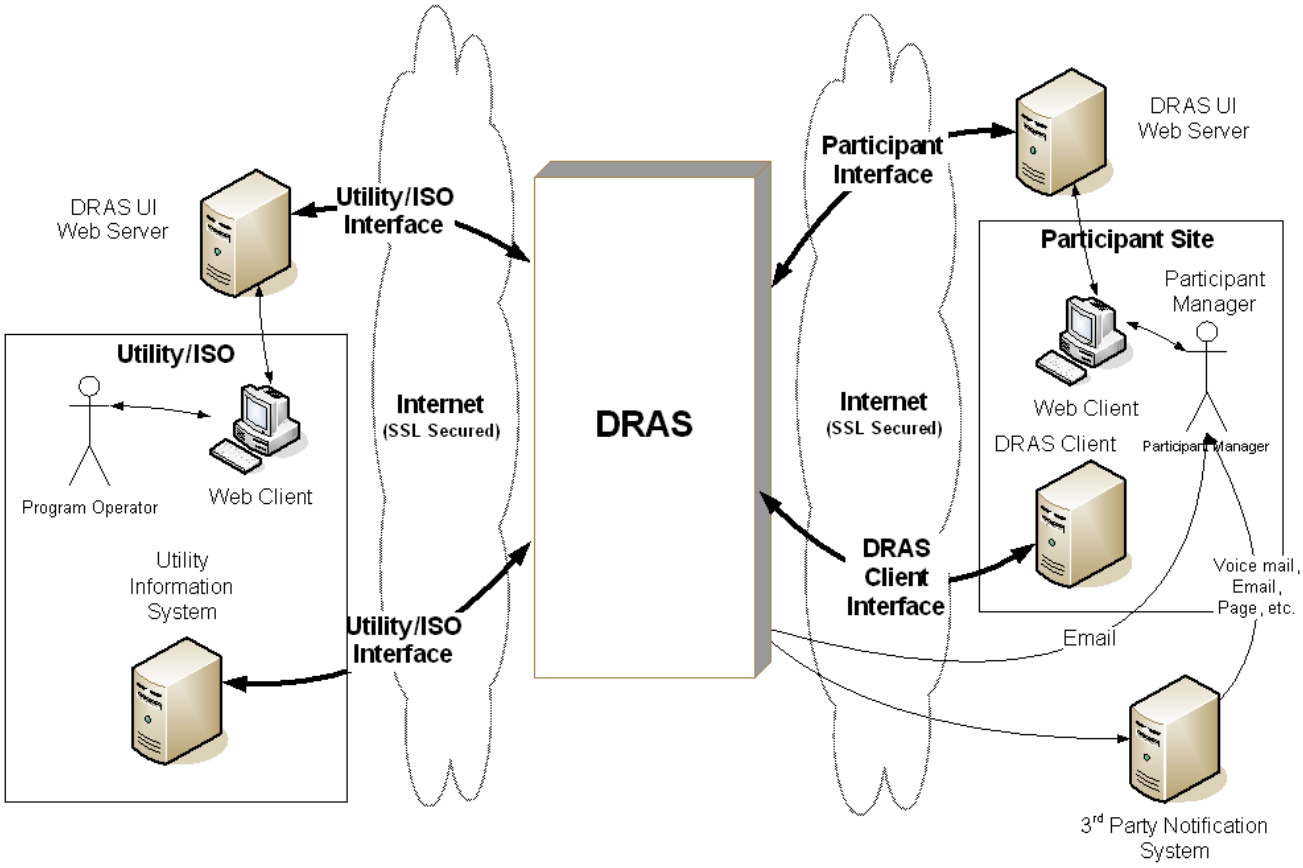

Figure 1: DRAS Generic Architecture

"DRAS" is the Demand and Response Automation Server and is a logical component that interfaces with various players and implements a DR Automation System based on various rules.

Briefly, there are three main interfaces to the DRAS:

1. Utility Operator interface – this is the interface that Utilities will use to set up, manage and monitor a DR program
2. Participant interface – this is the interface that participants of the DR program will use to convey electricity demand requests to the DRAS
3. DRAS Client interface – interface for notifying automation subsystems about DR program events & feedback to DRAS about state and responses for a DR event

It is important to note that the components and interfaces shown in Figure 1 are logical. In a live deployment one or more of the interfaces can be combined.

To understand how the actors interface in more detail, please refer to [1]

6.0 A SIP BASED PROPOSAL

This chapter proposes a realization of the OpenADR specifications using SIP. For the sake of brevity, actors depicting human users (like operator, participant manager) have been removed as they don't affect the core automation procedure for the scope of this paper.

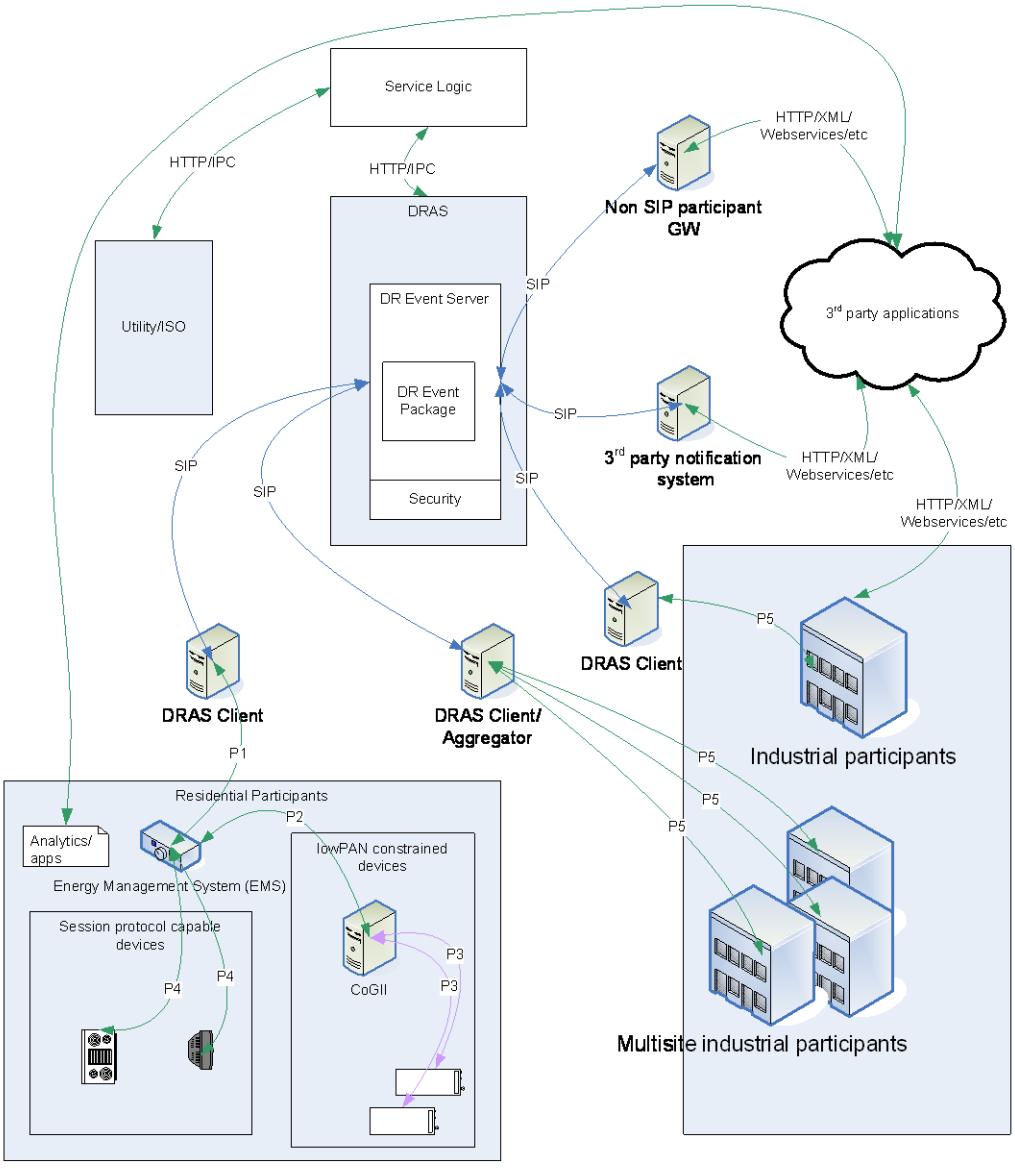

Figure 2: SIP Based DR architecture

6.1 Brief System Description

Figure 2 describes a proposed SIP based Demand Response architecture compliant to the OpenADR specifications. At a high level, the diagram can be broken up into the following components:

1. The Utility/ISO – actors who will be creating DR events
2. The DRAS – which is responsible for managing the execution of demand/response events and ensuring interoperability between participants and the DR system
3. Participants
    a. Industrial participants (hospitals, enterprise etc)
    b. Residential participants (home owners, typically)
        i. Note that the residential network has been further decomposed to show how the DR system interfaces with the residential EMS (via the DRAS client) and how the EMS in turn interacts with individual devices for power management. The terms used in the residential network, like CoGII are taken from current work that is ongoing in the 6lowapp group which is defining appropriate protocols for HAN automation
        ii. As an informational note, 6lowapp is still in early stages of defining the appropriate HAN protocol. Based on list discussions, there are two primary categories of appliances:
            1. 'Constrained Devices' – these are very low power and high constrained devices with around 24K of total memory and are incapable of supporting protocols such as SIP and XMPP
            2. 'Session capable Devices' – these are devices that have sufficient capacity to be able to directly support session oriented stacks like SIP on them thereby greatly enhancing the features and management possibilities with these devices in a HAN (Examples include cameras, washing machines, set top boxes etc)
        iii. To ensure that 'Constrained Devices' are capable of participating in an all IP-managed network, a CoGII (Constrained-to-General-Internet-Intermediary) node has been introduced, which is essentially a 'gateway' to help integrate such devices.

6.1.1 A note on interfaces P1-P5

Figure 2 does not specify the protocol of choice for certain segments, namely:

1. P1 & P5 – interface from DRAS client to EMS (of residential and industrial participants, respectively)
2. P2 – interface from EMS to a CoGII (Constrained-To-General-Internet-Intermediaries)
3. P3 – interface from CoGII to constrained devices
4. P4- interface from EMS to session capable devices

In deployment these interfaces could be realized by different protocols, including SIP. In fact, that authors believe that using SIP will ensure better end-end management and control. However, it is important to state that irrespective of whether these interfaces are realized by SIP or not, it does not affect the SIP based DR proposal.

6.1.2 DRAS AS

The proposed DRAS AS will serve the following roles as defined in section 5.2.1.2 of [1]

1. Event Notifier
2. RTP (Real Time Pricing) Notifier
3. Program Notifier
4. Bidding Proxy

The DRAS AS consists of the following components:

6.1.2.1 SIP DRAS Event Server

The SIP DRAS Event Server is SIP based Event server compliant to [3][2]. The role of the DRAS Event server is to:

- Receive updates from Utilities via SIP PUBLISH for new DR pricing schemes and rules
- Receive SUBSCRIBE requests from participants wanting to participate in a DR program
- SUBSCRIBE to the DRAS client to receive state of DR events
- Potentially SUBSCRIBE to relevant participant states as well for finer tuned feedback
- Issue NOTIFY commands to concerned participants whenever there is a change of state for the DR energy package and/or its participants

An integral part of the SIP DRAS Event Server is a new package manager which needs to be defined that is an XML representation of the DR attributes as specified in [1]. Essentially, this new event package will contain all the state and attribute information required by participants.

Note that as a terminology clarification, in the SIP Events world, all the actors are participants (including Utility). Some participants may be in a position to control/update the DR scheme (like Utility) while some may only be able SUBSCRIBE and report.(like end consumers)

6.1.2.2 DRAS Business logic

The DRAS business logic is an intelligence layer that works in conjunction with the SIP DRAS Event Server module and Service logic to create unique DR programs easily. As an example, it is very easy to implement a bidding module here, where based on current rate of energy and availability, multiple participants can perform a 'bidding war' for excess energy similar to auction sites. The DRAS business logic will also consist of logs, traces, configuration management and other non event reporting related functionality specified in [1]

6.1.3 Security

This module implements the required security functionality as required by [1]

6.1.4 Service Logic

The Service Logic module is a logically independent entity that performs intelligent schedule of DR events. Practically, it could be a part of the DRAS Business logic. It is envisioned that

standards such as [4] ,or variants thereof, could add immense value in an easily scriptable interface (towards the Utilities) that allow for complex distribution and routing of DR allocations to participants based on variety of factors including:

1. Time of day
2. Whoever answers first
3. Sequential and/or parallel allocation
4. Current location of participants (envision a mobile PEV directly participating in a DR program)
5. Current Presence state of participants (one can envision extensions to presence documents that define energy specific states relevant to DR programs)

6.1.5 SIP Event Aggregator

SIP has defined a concept of an 'Event Aggregator'. The role of the aggregator is to subscribe to an event package on behalf of several participants. The primary role of the Event Aggregator is to reduce network load. Instead of each individual participant subscribing independently to the event state, the aggregator can be a 'trusted proxy' which subscribes on behalf of a group of users. A SIP Event aggregator is not mandatory. It can be envisioned that such an entity may be useful in large enterprises with multiple 'energy participants' as well as community neighborhoods.

6.1.6 3rd party notification system

The 3rd party notification system could be a 'gateway' that opens up the doors to a variety of new applications that based on allowable security parameters is able to selectively subscribe to DR events and create a useful application with it. Example of 'useful applications' include:

1. Allowing developers to create intelligent analytics programs that help both participants and Utilities understand effectiveness/cost savings of their program
2. Deliver relevant information of a DR program to participants via IM/email/SMS etc.

6.1.7 Non SIP Participant Gateway

The non-SIP participant gateway ensures that participants which do not communicate using SIP are also able to participate in the DR program. Examples on non-SIP realization could be using HTTP, XMPP, proprietary APIs, Web Services, etc. It is straightforward to define mappings between SIP and such protocols (for example [6]).

6.1.8 DRAS Client

The DRAS client is a logical entity that is responsible for integrating participant energy management systems to the DRAS. Typically, the DRAS client will interface with the participant Energy Management System (EMS) and ensure commands from the DRAS are sent appropriately to/from the participant system.

6.2 Event State Model of actors

The relationship of the various actors in a DRAS when modeled as a SIP event state machine is shown below. The PUSH model interaction from DRAS to DRAS client can be achieved via SIP NOTIFY messages, while the PULL model interaction from DRAS client to DRAS can be implemented using SIP MESSAGE.

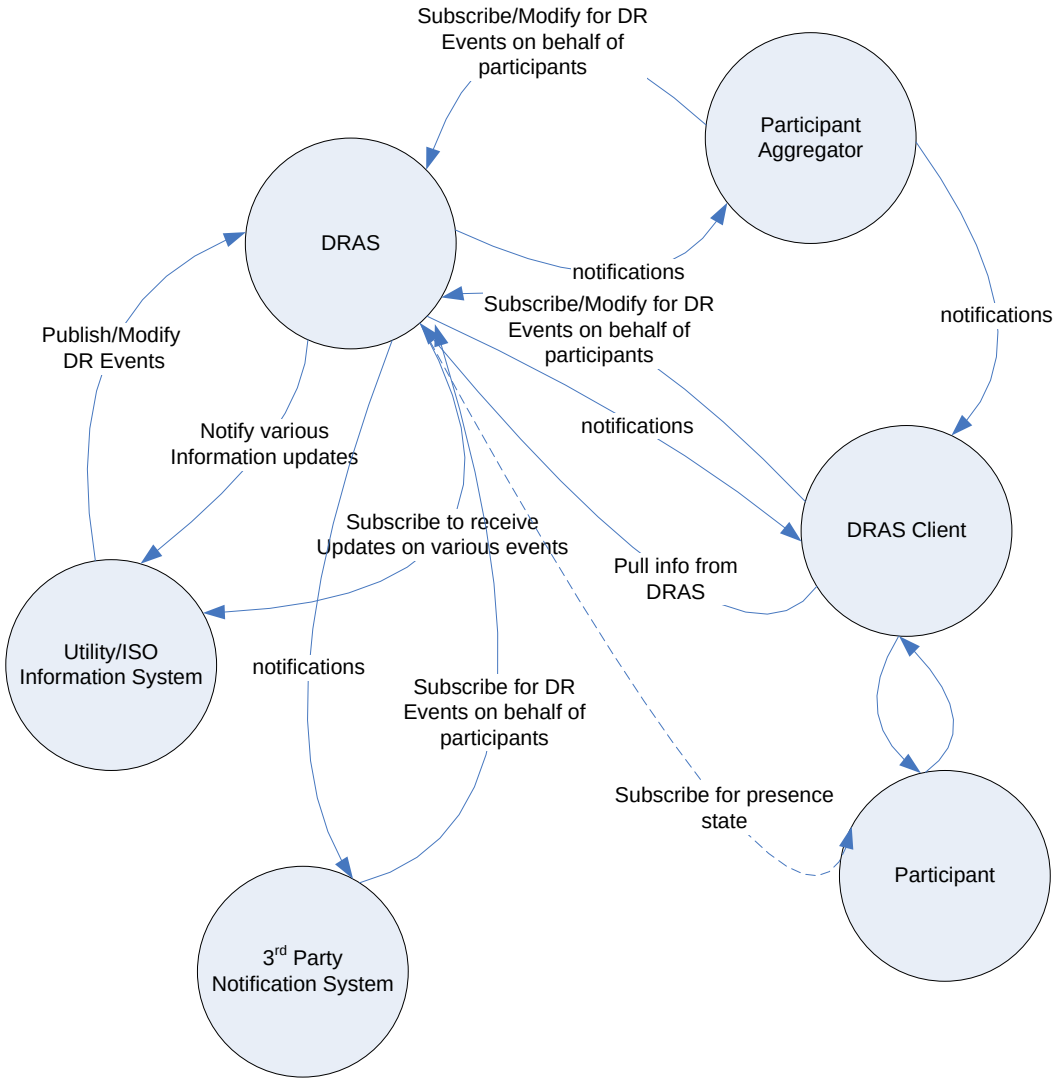

Figure 3: SIP Event State model between DR actors

7.0 SAMPLE CALL FLOWS USING SIP

This chapter illustrates how SIP can be used to realize some of the call flows of the OpenADR usecases.

Note: Message flows have been abbreviated where needed for sake of clarity (example, all message responses not shown)

7.1 Automated DR Event Notification

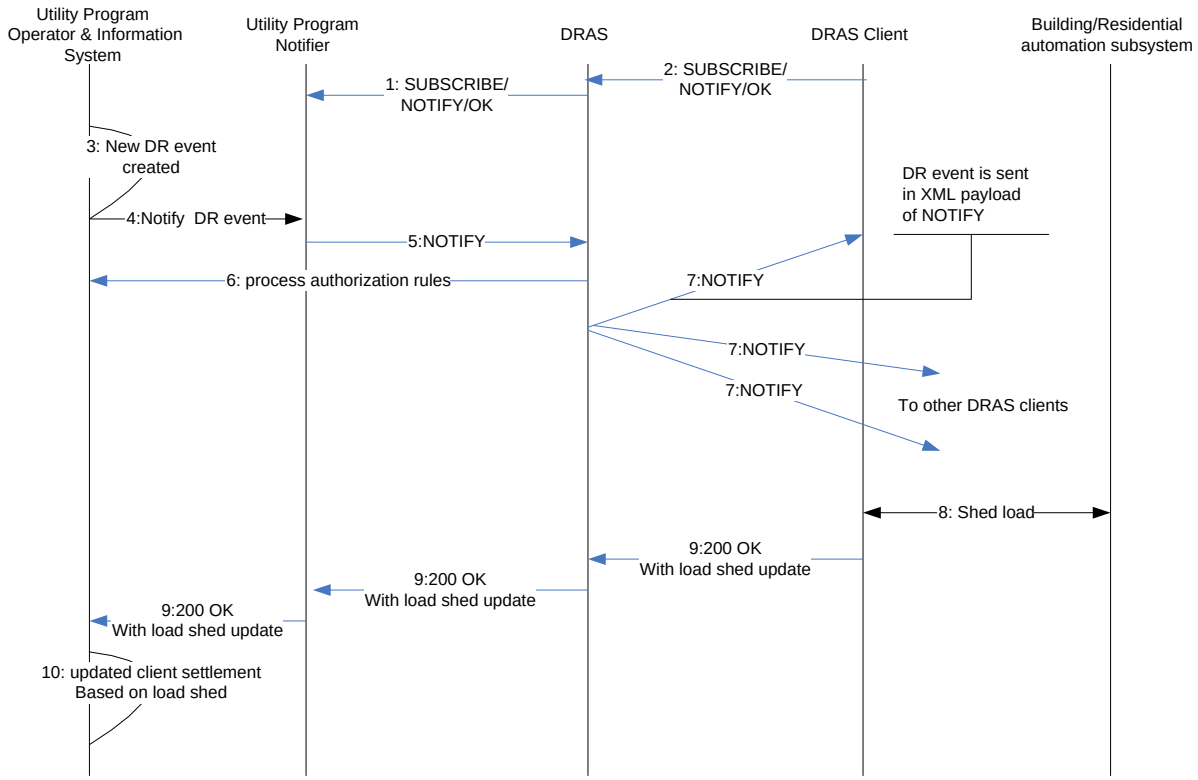

Figure 4: Automated DR Event Notification

1. As part of initial setup, the DRAS subscribes to the utility program notifier to be informed of any new DR events (this step is optional, and can be pre-provisioned).
2. Similarly, different DRAS clients SUBSCRIBE to the DRAS to participate in the DR package
3. A new DR event is created by the operator
4. The occurrence of the DR event is passed on to the Utility program notifier
5. The Utility Program Notifier in turn sends a NOTIFY to the DRAS about the new event
6. The DRAS checks the utility database about access permissions of this DR event (which DRAS clients are authorized, participatory etc)
7. The DRAS parallelly sends out NOTIFY messages to all the subscribed & authorized DRAS clients with the DR event in an XML payload

8. The DRAS clients in turn interact with their building automation subsystems (example, over BACNet, MODBUS etc.) to perform the shedding operation.
9. Once the operation is complete, each DRAS client responds with a 200OK response with the load shedding update for each site. **Note:** SIP also allows a 202 response which means “Shedding in progress”. If a DRAS client sends this response, a subsequent 200 OK response can be sent on load shedding completion. Alternately, a 400 class response can be sent if load was not able to be shed. This response is then propagated all the way back to the UIS which can reconcile any incentives or penalties as required.

7.2 DR Participating Bidding

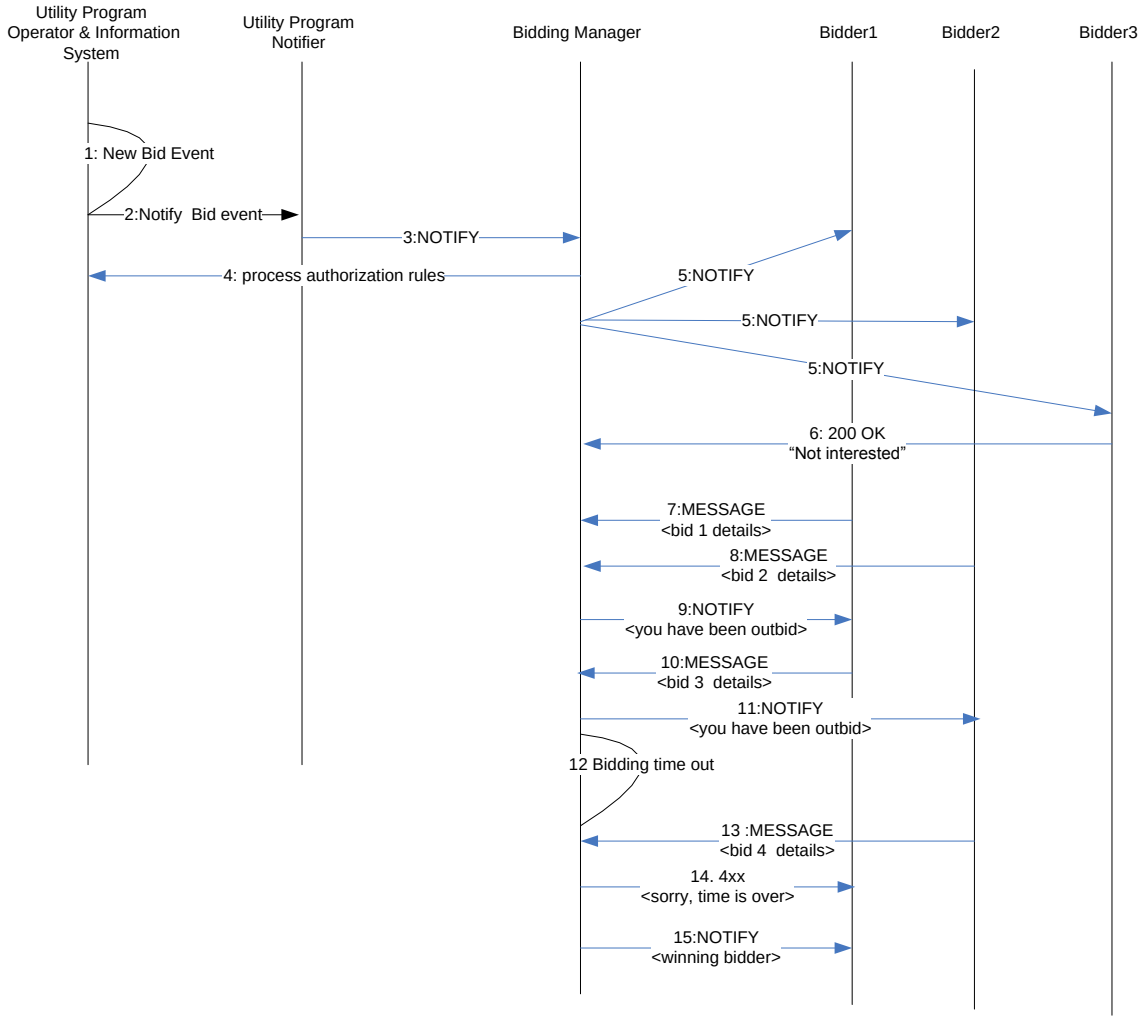

Figure 5: Automated DR Event Notification

1. A new Bid Event is created
2. The new bid is notified to the Utility Program Notifer
3. In turn, the Bidding manager is notified that it needs to collect new bids from participants
4. The Bidding Manager checks the authorization rules for recipients of this bid event
5. The Bid event is parellely sent to all participating bidders (3 in this example)
6. Bidder3 does not wish to participate in this bid. It acknowledges the request but does not provide any further bid
7. Bidder1 bids with a specific bid
8. Bidder2 bids with another bid
9. Bidding Manager (optionally) tells Bidder1 than it has been outbid by a better bid
10. Bidder1 resubmits a bid
11. Bidding manager (optionally) tells Bidder2 that it has just been outbid by a better bid
12. The Bidding manager decides bidding time is over

13. Bidder2 bids with another bid
14. Bidding manager refuses the bid due to timeout of the event
15. A notification is sent to Bidder1 that is has won the bid (optionally, Bidding Manager can send a notification to bidder 2 that it has lost)

8.0 OPENADR APPLICABILITY OF SIP: COMPLIANCY MATRIX

8.1 Where can SIP play in the DR Program Execution?

Table 2, “DR Program Execution” of [1] defines a convenient table for DR Execution use cases. SIP can play a role as the protocol of choice for the following scenarios (highlighted in green)

		DR Program Execution													
		Configuration			Actions on DRAS					Actions By DRAS			Maintenance		
Program		Configure Program	Configure DRAS Client Connection	Configure Bids	Request Bids	Initiate DR Event	Program Opt Out	Set Load Status	Set Current Bids	Send DR Event Info	Notify Request for Bid	Notify Bid Status	Utility Operator Reports	Client Reports	
CPP	Utility Program Operator	X	X										X		
	Utility Program Notifier					X									
	Utility Info System														
	DRAS Client							X		X					
	Participant Manager		X				X							X	
DBP	Utility Program Operator	X	X										X		
	Utility Program Notifier				X	X									
	Utility Info System								X						
	DRAS Client							X		X					
	Participant Manager		X	X			X				X	X		X	
CBP	Utility Program Operator	X	X										X		
	Utility Program Notifier				X	X									
	Utility Info System								X						
	DRAS Client							X		X					
	Participant Manager		X	X			X				X	X		X	
BIP	Utility Program Operator	X	X										X		
	Utility Program Notifier					X									
	Utility Info System														
	DRAS Client							X		X					
	Participant Manager		X				X							X	
PDC	Utility Program Operator	X	X										X		
	Utility Program Notifier					X									
	Utility Info System														
	DRAS Client							X		X					
	Participant Manager		X				X							X	
PCT	Utility Program Operator	X	X										X		
	Utility Program Notifier					X									
	Utility Info System														
	DRAS Client							X		X					
	Participant Manager		X				X							X	
RTP	Utility Program Operator	X	X										X		
	Utility Program Notifier					X									
	Utility Info System														
	DRAS Client							X		X					
	Participant Manager		X				X							X	

Figure 6: Role of SIP in DR Execution

8.2 Role of SIP in Generic Event Based Programs

Similarly, the role of SIP in the Automated GEP use case in Figure 2, section 5.2.3.1 of [1] is shown below. Specifically, the SIP methods that can be used to realize the usecases are shown below:

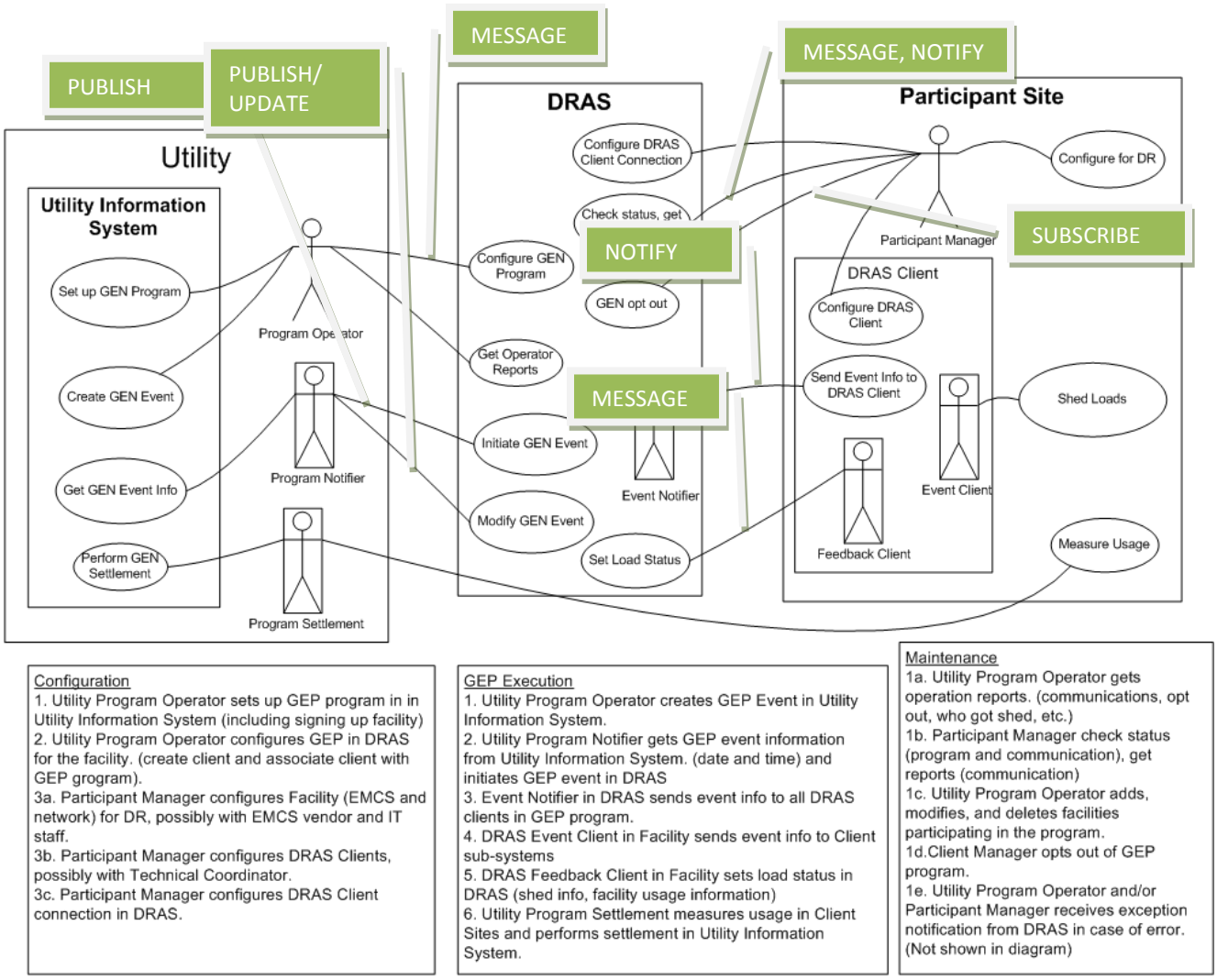

Figure 7: SIP protocol realization of generic event based program

8.3 Role of SIP in Generic Bidding Program (GBP)

SIP plays an important role in the Bidding process as well for DR management. Mapping of potential SIP methods to the GBP usecase as shown in Figure 2, section 5.2.3.2 of [1] is shown below.

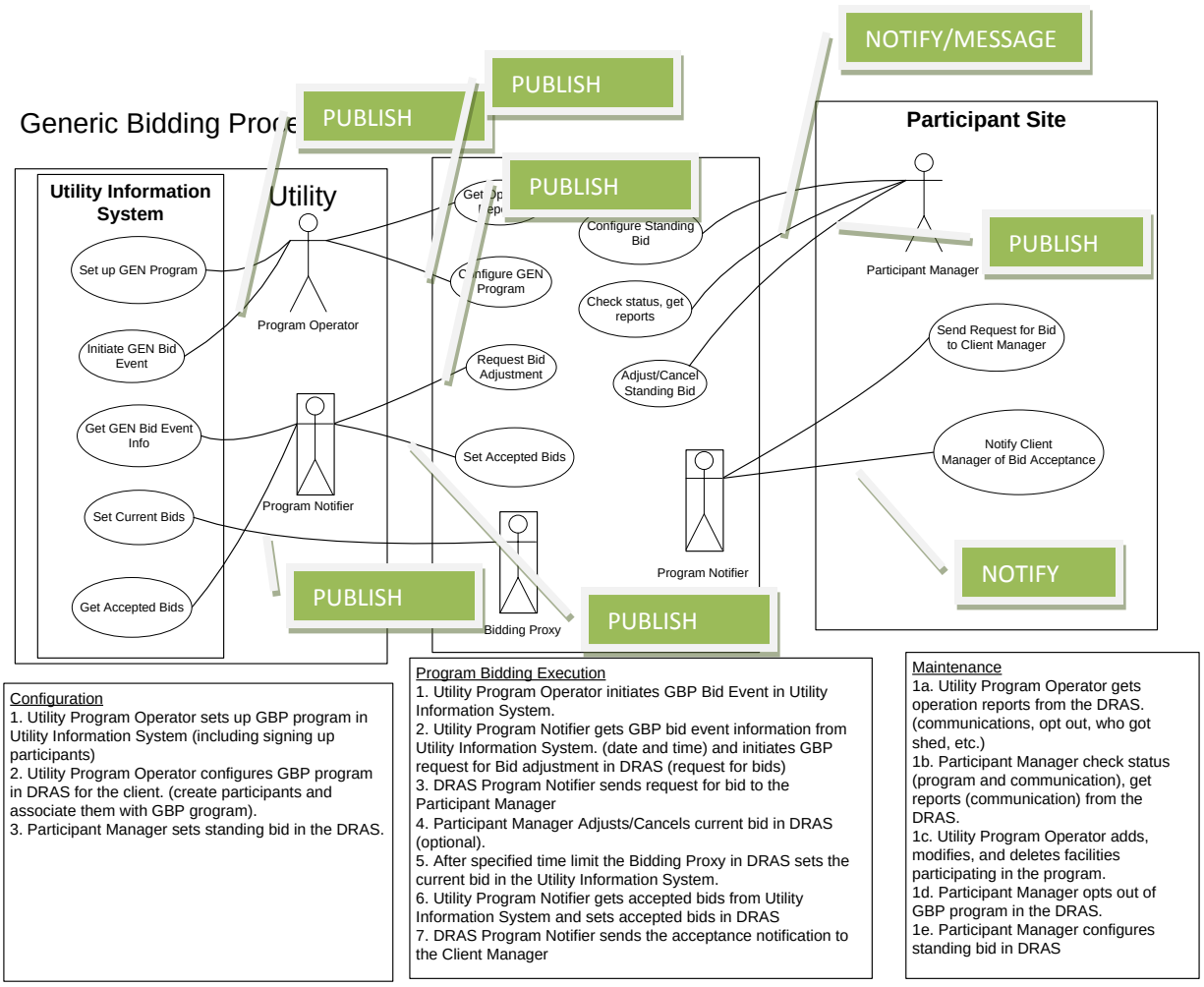

Figure 8: SIP protocol realization of generic bidding program

8.4 Meeting requirements of a DRAS system

Section 5.3 of [1] specifies some minimal requirements, independent of usecases that need to be met by any solution. The table below summarizes the requirements and whether SIP meets these requirements.

#	Requirement	Compliance
1.	Should use industry accepted methods and standards to ease integration and access to DRAS services from third parties	**Y.** 3rd party applications can act as 'subscribers' to DR state and receive as well as participate in the SIP based DR ecosystem. SIP allows for different authentication/authorization and privacy rules on a per-participant basis which can be leveraged to allow for differentiated information access as well. Alternately, one can easily develop a HTTP based webservices API front end that non SIP compliant applications can use to access the same information. Most open source as well as commercial SIP application servers support HTTP servelets for this purpose as well.
2.	Must follow a well established set of security policies to insure that all exchanges of information are authenticated, private, and maintain integrity of the information being exchanged.	**Y.** SIP supports TLS as well as a host of other authentication and encryption standards.
3.	Must allow easy integration with end user facility IT infrastructure:	**Y.**

	• Ease dealing with firewalls • Good IT network citizen (i.e. no security risk, insignificant network load, etc)	There is a lot of confusion on firewall traversal and an application level protocol such as SIP. Very simply put, a problem occurs when someone outside of a secure firewall needs to communicate to someone inside the firewall on a random port. This is typically the case with dynamic media- the firewall will typically not allow dynamic port openings unless the entity behind that firewall first opens that connection. There are several ways around this, including opening an inside connection out and having responses piggybacking this connection to avoid firewall issues (like what XMPP does), using STUN, TURN, ICE etc. for automatic firewall traversal etc. In the case of Demand and Response – Firewall traversal is NOT an issue. Furthermore, with SIP firewall traversal is a lesser problem than other protocols. Here is why: Demand and Response is an event based control mechanism. The entire DR sequence can be transmitted and received using SIP's well known port: 5060 (UDP or TCP). No random port openings are required. Furthermore, SIP is the defacto standard for any enterprise VoIP system. That means that SIP is already there and firewall ports are already opened. If Demand and Response used the same infrastructure (which is what this paper is proposing), then firewall traversal is not an issue at all. If SIP were not used, IT departments will have to worry about provisioning another port for another protocol in their firewall. SIP is the defacto protocol in deployment today for most VoIP systems across wireline and wireless. The large deployment of SIP on security critical networks such as wireless networks is proof of SIP's security considerations. From a load perspective as well, SIP's even notification system incorporates mechanisms like event-throttling and event aggregators that significantly help reduce network load. Again, we do not believe for DR management SIP introduces load overhead – the above mechanisms are for the information of the reader.
4.	The latency of DR Events sent from the Utility to the end user should be no more than 1 minute, depending upon the configuration of the interaction between the DRAS and DRAS Client.	**Y.** SIP is regularly used for call-setup where a 3 way INVITE/OK/ACK is completed in sub second to 2 second durations in a wireless network. SIP easily conforms to the 1 minute requirements.
5.	The DRAS must maintain accurate time within 15 seconds	**NA.** This requirement does not affect SIP.
6.	The DRAS should have a means to allow Participants to participate in multiple DR programs and dynamic pricing through the same DRAS	**Y.** In the SIP Event architecture, a subscriber, identified by its SIP identity can simultaneously be part of multiple DR programs, completely independent of each other.
7.	The DRAS should recover gracefully from facility faults with minimum lost data. Examples of such faults might be power failures or connectivity loss	**Y.** SIP is already deployed in carrier grade voice and messaging systems that have similar requirements. Specific to a DR system, state maintenance is very simple. SIP supports session resumption after connection is broken.

9.0 OPENADR: ACHEVING DRAS FUNCTIONALITY WITH SIP

Section 6 of [1] provides specifications for various components of the DRAS. This chapter provides descriptions of how such functionality can be achieved using SIP.

9.1 Common Requirements (Section 6.3 of [1])

9.1.1 DRAS User Accounts and security roles (section 6.3.1 of [1])

This functionality requires that each of the functions DRAS provides should only be accessed by users with appropriate authorization. It is our understanding that the DRAS will maintain an appropriate database of security requirements. From the SIP perspective, subscription & subsequent notification of DRAS events can be controlled using solutions such as Presence Authorization Rules specifications [8]

9.1.2 Logs and reports (section 6.3.2 of [1])

In general, out of scope for SIP. Section 6.3.2.3 requires that DRAS client status needs to be reported (ON/OFF). This can be achieved by the DRAS Event Server subscribing to presence state of DRAS client. Online/offline presence events will then be reported from the DRAS client to the DRAS Event Server. Note that this can also be achieved by simpler means such as a DRAS client simply having to REGISTER to show ONLINE and de-REGISTER or timeout to show OFFLINE.

9.1.3 Operator Notification (section 6.3.3 of [1])

1. Section 9.1.2 above describes how DRAS client ON/OFF can be reported.
2. According to the event architecture of SIP, all Participant Operators will SUBSCRIBE to receive DR Event state for a particular DR package. By definition, whenever there is a change to this package, subscribers will be automatically notified via NOTIFY
3. A participant can submit its bid to the DRAS using the SIP MESSAGE method. An acceptance or rejection can be confirmed by the DRAS in response to the MESSAGE request (200 OK = accepted, 202 = pending, 4xx – 5xx = rejection, along with result). For pending decisions, the final decision can be conveyed using a SIP NOTIFY. Furthermore, if required, approval/rejection can also be reported to other participants if required using a SIP NOTIFY from the DRAS.

9.1.4 Testing (section 6.3.4 of [1])

This requirement specifies that it should be possible to conduct “test” DR events which would behave just like real DR events but it would include an attribute for field DRAS clients to let them know it is just a test event. How a DRAS client treats a test event depends on the DRAS client. This is easily achieved in SIP by adding a test tag to the Event: header while reporting the event. Doing so at the header level can help in DRAS clients not needing to parse the payload unless it would want to.

9.2 Data Entities used by Interface Functions (Section 6.4 of [1])

In general, all the data entities described in [1] can easily be provisioned as is, as part of the schema definition of the appropriate event package. In fact, [1] already specifies the XSD schema definitions of these entities. The same schema definition files can directly be imported into the event package.

9.3 DR Event Models (Section 6.5 of [1])

This section of the OpenADR specifications deals with how events are modeled at the DRAS and DRAS client. This is mostly an application level logic – SIP can be used to execute in compliance with the appropriate application logic. As mentioned earlier, for each event model, the appropriate parameters can be represented as part of the payload in the respective SUBSCRIBE/NOTIFY/PUBLISH/MESSAGE methods of SIP.

From a authorization & program constraints perspective (i.e. which DRAS clients will be party to specific DR events, based on a variety of attributes like location, authorization etc), such authorization schemes can easily be provision using event rules such as those provided by [8]. Furthermore, since the rules specification is in XML, it allows for extensions as required.

9.3.1 Modes of interaction between DRAS and DRAS client (Section 6.5.3.1 of [1])

As per OpenADR specifications, the DRAS could either send DR event state to a DRAS client when state changes (PUSH Model), or, the DRAS client could request the current state from the DRAS (PULL Model).
From the SIP model perspective a PUSH is analogous to the SIP NOTIFY mechanism while the PULL model can be realized using a SIP MESSAGE method.

9.3.2 Simple Versus Smart DRAS clients (Section 6.5.3.2 of [1])

There are essentially two types of DRAS clients:

1. Simple DRAS client – This is a DRAS client that is not capable of processing all the attributes of a DR event
2. Smart DRAS client – This is a DRAS client that is capable of processing all the attributes of a DR event

From a SIP perspective, this basically means that different DRAS clients (representing different participants) will have separate capabilities on how granular their participation will be, for a given DR event. The capability of each DRAS client can be advertised when they first SUBSCRIBE to the DR Event package and according to the subscription (or pre-provisioning of a DRAS client database), appropriate event information as per section 6.5.3.3.1 or 6.5.3.3.2 of [1] can be sent by the DRAS.