Research Report
Research Project T9903, Task 30
IVHS—Network and Data Fusion

A SELF-DESCRIBING DATA TRANSFER METHODOLOGY FOR ITS APPLICATIONS: EXECUTIVE SUMMARY

by

D.J. Dailey, D. Meyers, and L. Pond
ITS Research Program
College of Engineering, Box 352500
University of Washington
Seattle, Washington 98195-2500

Washington State Transportation Center (TRAC)
University of Washington, Box 354802
University District Building
1107 NE 45th Street, Suite 535
Seattle, Washington 98105-4631

Washington State Department of Transportation
Technical Monitor
Dave Peach

Prepared for

Washington State Transportation Commission
Department of Transportation
and in cooperation with
U.S. Department of Transportation
Federal Highway Administration

December 2000

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
WA-RD 463.2		

4. TITLE AND SUBTITLE	5. REPORT DATE
A Self-Describing Data Transfer Methodology for ITS Applications: Executive Summary	December 2000
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
D.J. Dailey, D. Meyers, L. Pond	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
Washington State Transportation Center (TRAC) University of Washington, Bx 354802 University District Building; 1107 NE 45th Street, Suite 535 Seattle, Washington 98105-4631	11. CONTRACT OR GRANT NO. Agreement T9903, Task 30
12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT AND PERIOD COVERED
Washington State Department of Transportation Transportation Building, MS 7370 Olympia, Washington 98504-7370 Project Manager: Gary Ray, 360-705-7975	Research report 14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

This study was conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. ABSTRACT

A wide variety of remote sensors used in Intelligent Transportation Systems (ITS) applications (loops, probe vehicles, radar, cameras) has created a need for general methods by which data can be shared among agencies and users who own disparate computer systems.

In this paper, we present a methodology that demonstrates that it is possible to create, encode, and decode a self-describing data stream using the following:

1. existing data description language standards
2. parsers to enforce language compliance
3. a simple content language that flows out of the data description language
4. architecture neutral encoders and decoders based on ASN.1.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Self-describing data, SQL, ITS data, ITS architecture	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
None	None		

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Washington State Transportation Commission, Department of Transportation, or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

TABLE OF CONTENTS

Section	Page
Describing "Self-Describing Data": Looking Under the Hood at the Seattle MDI Data Engine	**1**
Self-Describing Data for Dummies	**19**
Introduction	20
What Is Self-Describing Data?	20
What Kind of Actual Data Should be Made Self-Describing?	21
About the Description	21
More Wrapping for SDD	23
What Goes into a Data Dictionary?	24
When Do You Get a Data Dictionary and When Do You Just Get Data?	25
How Do You Know Whether or Not You Received a New Data Dictionary?	**25**
Software for SDD	25
How Do I Tap into an SDD Data Stream?	26

Describing "Self-Describing Data"

Looking Under the Hood at the Seattle MDI Data Engine

(http://www.nawgits.com/sdd_dd.html)

The Seattle Smart Trek *(http://www.smarttrek.org/)* Project, perhaps more than any other Model Deployment Initiative *(http://www.its.dot.gov/mdi/)* (MDI) project, includes the participation of a wide range of private-sector firms as so-called "Information Service Providers" or "ISPs." These ISPs often take raw data about traffic conditions (typically from loop or video detectors) and then "add value" by post-processing that data and communicating it to travelers through a variety of channels and devices. A key question in that scenario: how best to acquire that data while requiring a minimum of hand-holding on the part of its source (in this case, the Washington State Department of Transportation)? Many of the Smart Trek projects have found the answer in so-called "self-describing data," in which the data stream itself contains not only raw traffic data but information about where the data comes from and what it means. ICDN Editor Jerry Werner recently discussed the background and potential national applicability of SDD with its developer and foremost proponent, Professor Dan Dailey of the University of Washington. Dailey, the director of UW's Intelligent Transportation Systems *(http://www.its.washington.edu/)* program, is a research track Associate Professor in the Department of Electrical Engineering who also holds adjunct appointments in the departments of Civil Engineering and Technical Communication in the College of Engineering.

The Discussion

ICDN: How long have you been involved with ITS projects?

Dailey: My relationship to using traffic data in modeling goes back to 1989. My involvement was initially modeling using data from the Washington State Department of Transportation traffic management center. The present ITS backbone activity evolved as sort of a defense mechanism. We had a series of students who wanted to do modeling. Each student that came along said 'well, how do I get the data?' and invented a new way to do so. Eventually, we came up with the idea for reusable code to make the data available and not spend the first six months of a project redoing the data acquisition task. Instead, students can spend time actually doing something with the data. So, we created a set of Unix applications that were designed to extract data from remote places, in this case the Washington State Dept. of Transportation's traffic management center. We obtained data from loop sensors for about 3,000 locations around the region's highways. We can measure something about the duration of time a vehicle is over the loop and from that do modeling. At about the same time, we also learned that the people at Metro Transit, the local transit operator, put a vehicle location system in operation — a way to track their buses. A student took a "sniffer" and eavesdropped on their network in their building downtown. He found that we could collect packets off their network containing data about how often the bus wheels turn and bring that data back to the UW to do modeling.

ICDN: Sounds like your students will be well prepared for a career in espionage!

Dailey: You use what tools you have available. Actually, espionage is perhaps an apt term, because what you have to do to obtain data from many agencies is essentially espionage. The agencies have a funded mission directed toward doing a particular task, and that mission really has nothing to do with providing data to "those guys at the University" for research. They're not going to allocate staff resources or any dollars or time to support the University if it detracts from their mission. They'll let

you put your computer under the coffeepot, and if you can hook up to the wire and it doesn't cost them anything but the electricity to run your computer — that's okay!

ICDN: Did Washington State DOT fund this early work?

Dailey: Yes, WSDOT and the USDOT gave me little bits of funding, starting at about $21,000-a-year, several years ago, which was steadily built up by a series of additional projects. The ITS notion of combining data from a lot of places and making products downstream of it was becoming popular then. We had already been in the business of pulling the data out of agencies to do these smaller projects. We were already doing what needed to be done to build a backbone. Actually, I had attended a number of meetings as early as 1992 and suggested that most regions need a backbone to make modeling and applications possible in more communities. That idea has had a mixed appeal: It has a positive appeal to public-sector transportation agencies in terms of potentially creating a level playing field, but it has a negative appeal for commercial entities — that would like to control that data.

ICDN: Do companies have an interest in pulling out data through some sort of backbone, but not of putting new information in?

Dailey: Yes, they have an interest in controlling data access, as information is potentially money. There is this notion that if you're the only company that can sell data to others — you have a contract to be the single source — that you can then charge whatever fees you want. Conversely, if your vision is for a State to have an obligation to create a level playing field among commercial information providers, you would want to come up with a set of paradigms and protocols to effectively give away that data in a clearly understandable format. That is the "self-described data" concept, which largely eliminates the need to then answer downstream questions about the data. We used to receive requests for traffic data from Research Triangle (North Carolina) and California among others, and we would provide that data to them. Then they'd call back the next week and want one of our programmers to consult with them for a month to explain the data to them. We

didn't have that resource. So we created a protocol called self-describing data that in theory addressed that problem.

ICDN: Self-describing data is being used on a number Smart Trek projects, right?

Dailey: Yes. The idea is that you need to get one self-describing data receiver and then you can understand data from a variety of sources.

ICDN: Users tap into that data via the Internet, is that right?

Dailey: We use a set of Internet protocols because the Internet exists and it's the cheapest way to do it. You could have dedicated connections, but the Internet is the low-cost way of getting the data.

ICDN: I understand that this receiver you refer to is written in the Java language.

Dailey: Yes, we give away a Java-based receiver.

ICDN: You chose Java so it will run on different platforms?

Dailey: That is correct. We originally programmed it in C on a Unix platform, but the MDI project said to us, "You have to run a Windows NT environment,' and we in turn asked, 'What version?' Their reply was, 'There are versions?' So we adopted Java in order to be platform independent.

ICDN: Which specific Smart Trek applications are using SDD right now?

Dailey: Traffic TV *(http://www.its.washington.edu/trafchan/)*, Buslink *(http://www.its.washington.edu/buslink/)*/Transit Watch *(http://www.its.washington.edu/transitwatch/)*, and Busview *(http://www.its.washington.edu/busview/)* all use it. We're giving it away to the North Seattle ATMS, which is a flow of data from WSDOT and then back into WSDOT. North Seattle ATMS is a new management system they're bringing on-line, but the easiest way for them to get the data out of their historical systems was just tapping into the backbone. It's also being used

by Fastline, which markets traffic applications on hand-held, palm-top computers, and it has been used by a number of research institutions around the country.

ICDN: Is it being used yet in any other similar ITS projects, like TravInfo in San Francisco's Bay Area.

Dailey: No, not at this point. I don't know the details of the other projects, but until you have some thing that you hand them — some code you can use to build transmitters and receivers — people won't adopt it. People didn't rush up and adopt Internet protocols; they adopted the ability to do work with these protocols. So, now we're giving away the receivers and writing generic transmitters to give away in the next release from our web site. And people can use it. We've had inquiries from the other sites, but without making it so easy that they wouldn't do anything else, they're not going to adopt it beyond their specialty protocols. We were involved with Etak, Seiko, and other companies in the previous SWIFT project. At that time, it was simply much easier for them to describe a protocol that would work that year for that data and be done with it, than to think about what might happen over the next 5 years.

ICDN: Is Etak using it now?

Dailey: I don't know. They've expressed an interest in it, but I don't know if they're actually using it right now. We built a specialty protocol for them for the SWIFT project, which I believe they are still using.

ICDN: I understand that your self-describing data approach is an alternative to having a central data server, such as the one used in Atlanta for the Traveler Information Showcase. Is that a fair statement?

Dailey: Yes. There's no requirement to geographically centralize the data. Instead, the SDD protocols allow you to get data from a number of sources. You can imagine those sources would live in a number of different places geographically. What's common is the way that you get it. In a

simple example, SDD is used in the same sense that you use the HTML language for web pages: you share a language between your browser and the people who are sending information. You can go to a lot of different places on the web and get that data into your browser. SDD involves a much more complex language to describe data, but it's essentially that same concept.

ICDN: Is the transmitter the piece of software that gets the data onto the network via a TCP/IP protocol?

Dailey: The transmitter is a piece of software that takes real-time legacy data, which all of the agencies produce — usually in some binary format for their own management purposes, and packages it into the SDD format. The SDD format is a combination of intelligent "meta-data" that says what the data is; the content that says what type of sensors we have, where they are located, and what they measure; and finally binary data. We talk about those three pieces on technical papers that are on-line *(http://www.its.washington.edu/its_pubs.html)*. The combination of not only the binary data but also an understanding of what it is, is what SDD gives you in completeness.

ICDN: Does the description of the data, what you call the "meta data," take up much overhead in the overall scheme of things?

Dailey: Anything that provides additional information is information rather than overhead. You can view it as information or as overhead in the same sense that if you use TCP/IP (Internet) protocol and type one character on your keyboard, you sent 43 bytes to get that character from there and back. But those 43 bytes guarantee your character will get to there and back correctly. The important thing here is that that each time the data structure changes — say, for instance, that the folks at WSDOT add new sensors on the highway — somehow you need to communicate that dynamically to everyone downstream of you so that they can use the new data. In that case, you would need to send a new data dictionary because without a data dictionary, the new data would be meaningless.

ICDN: As I understand it, you just send the data dictionary once, you don't send it periodically every so many seconds?

Dailey: You send it once when you connect because you don't know what the data is until you get the data dictionary. Then whenever the source of a data changes it, a new data dictionary is transmitted so you can understand the following data. In our case, we were working with the WSDOT people who could sit at their console and change the data once a day, once a minute, or once a month without telling us. We had no control over that process. Realistically, operators don't change things a lot. But they couldn't tell us they would do it once a day, once a month, or whatever, so the "overhead" is that you need to understand what the data is and that "overhead" takes place anytime the meaning of the data changes.

ICDN: So, what would be involved if another DOT — say Illinois DOT, because they have a policy of making data freely available to anybody who can use it — is interested in using self-describing data? How big a process is it for them to understand the SDD concept and to put their data out on the Internet and to work with ISP's that want to take it? Have you done most of the legwork already? Is SDD "plug-and-play" at this stage?

Dailey: Transmitters are the keys to that, because transmitters have to turn legacy data into generic data. We have a transmitter under testing right now; we have 2 more kinds of data coming to us, and we're going to build specific transmitters with our generic transmitter and tell folks how hard it is to do. Instructions are available right now to do that, so someone who was modestly clever could take what we have right now, hook it up to their data, and make it work.

ICDN: The transmitter needs to be customized for each site?

Dailey: Of course. Think about it. Each site has custom data; we're trying to make the transmitters generic.

ICDN: How about on the receiving end?

Dailey: The receiver we're giving away now would work for any data anywhere. So once Illinois customized our transmitters to work with their data, they could use the same receiver we're using in Seattle to receive data in Illinois.

ICDN: In reading some of your documentation, it's clear that your receiver outputs data in a raw tabular and ASCII format, but obviously a lot of post-processing would be required to do anything with that data. Is that a fair statement?

Dailey: Of course. The intent of the receiver was never actually to be "the application" The intent of the receiver was to be a framework in which you could build an application to use the data. Once users discovered that they could get these tables of data and understand what these tables meant, they didn't bother using the model to build new applications, they just used the data in a couple of cases. That's one model. Another model that Fastline uses involves taking both the data and our hooks in the databases and actually pulling the data out and putting it in their own database. Downstream, they can then do whatever they want with their own database. So, there are two models that people follow — the "quick-and-dirty model" and the "long-term investment model."

ICDN: You make the software, instructions, and documentation for self-describing data freely available, is that right?

Dailey: Yes.

ICDN: Is licensing involved?

Dailey: Yes, the whole package is Copyrighted by the University of Washington, and any transmitters we give away right now will say "For Noncommercial Purposes Only." If a company or consulting firm wants to use it in a product they sell to someone else, then we'd have to address that licensing issue. The commercial use details haven't been worked out; there's no reason to worry about that until there's money on the table.

ICDN: So no money from SDD licenses is flowing into the university at this stage?

Dailey: Our objective is to give it away in hopes that if it is adopted by public agencies it will provide a way of sharing data across the country.

ICDN: Will SDD receivers require a license or will they be royalty free?

Dailey: Well, I"d get chewed out by the intellectual property people of the university if I said they were royalty free. They are copyrighted by the University of Washington and available for use for noncommercial purposes.

ICDN: But aren"t you trying to encourage use by commercial ISPs (Note: the term ISP in this context means "information service provider") that are sprouting up around the country?

Dailey: Right. And we would happily cooperate with them. For right now, we are giving it away. We know who's using it, but we haven't stopped anyone from using it.

ICDN: Okay, but there could be some royalties involved down the road?

Dailey: There could be, but it hasn"t come up.

ICDN: Is the SDD compatible with the "National ITS System Architecture?" I know they've looked at it many times.

Dailey: Yes.

ICDN: Did it help define portions of the architecture, do you think?

Dailey: Absolutely not.

ICDN: What is its relationship with National Architecture?

Dailey: The National ITS System Architecture envisions named flows between certain types of agencies, includingTransit Management and Transportation Management Agencies. The SDD is an

implementation of those flows. We're considering overlaying the naming convention that they use for the National Architecture onto the SDD scheme, so it becomes clearer where the SDD names map into the National Architecture names.

ICDN: That's probably a good idea.

Dailey: Yes, but it takes time.

ICDN: I know it does, but it might reduce some potential concern and it might make it clearer how the SDD concept fits into the national architecture.

Dailey: We're going down that path. We can't have that task take priority over what we're doing, however. We're doing the Seattle MDI project first and my national vision second because of the way the funding works out. There's funding for the MDI project and not for my national vision. But, no, the person who's working building the transmitter right now has attended the architecture classes; we have the national architecture CDs, and are working on architecture compliance. It's clearly mapped into the architecture. It will be clearer when we change our naming structure to match the architecture names. We had long discussions about that in January; however, we had a change of personnel because right now it's a hot job market out there. We are now starting those discussions over again.

ICDN: How does the SDD approach relate to NTCIP? I know you're doing a data dictionary and that NTCIP also has an ongoing data dictionary effort.

Dailey: NTCIP is a set of protocols. The ones that are implemented right now address communication between traffic control masters on the street and the agency, and changeable message signs — all what might be called "intra-agency applications." This fall, we'll be publishing a TRB paper that discusses exactly how NTCIP and SDD fit together.

ICDN: Is that paper already written?

Dailey: Yes, it's being reviewed right now. In essence, what it comes down to is that NTCIP has a number of types of defined kinds of data, and they have room for defining more kinds of data. So NTCIP can have simple data types and complex types as described by a language called "Abstract Syntax Notation.1." It's an ISO standard. Some implementations allow you to transmit that data. We actually transmit using the ASN.1 encoding of the data. NTCIP and ISO have a tree of numbers and names. So, if they'd assign us a number at the bottom of their tree saying this is one of the data types, we would then map into their data notions. The difference between the larger NTCIP efforts, which don't involve just data communication but involve dictionaries and so forth, is that there are 2 different notions that you can hold pertaining to dictionaries: One is we all go out and we buy the same dictionary. I can say 42 to you because you can look in your dictionary and know what 42 means. The other notion is that you may know in advance that your dictionary may change dynamically, that is the model we are using, and it requires transmitting a new dictionary when the data meaning changes.

ICDN: You have the ability to change, enhance, or expand the dictionary yourself at anytime, don't you?

Dailey: Right. Or contract it — whatever is necessary to make it work. We're sort of a super-set of the dictionary that everyone is holding, because if everyone hooked up and got SDD at this moment, we'd all hold the same dictionary. The models would match exactly. Until that dictionary changed, we'd all be doing NTCIP. When the dictionary changed then, the NTCIP model doesn't have any mechanism except for going through a committee process of changing the dictionary, which makes it a little harder.

ICDN: So, do you anticipate more and more discussions about the data dictionary with the NTCIP folks?

Dailey: I attended a number of meetings with NTCIP and presented our SDD ideas early on in the development of what they're doing. There's no conflict and they're well aware of our work. Nothing they're going to do will conflict with this work.

ICDN: Do they see the SDD concept as "competition" as far as you know?

Dailey: I never asked them that question. I asked them 'Are we compatible?' and 'Can we work within your domain?' We shouldn't be competing — we should be collaborating. I should stress that — this should be something that fits underneath their rubric if we're all doing our jobs.

ICDN: Is it fair to characterize you as the "granddaddy" of SDD?

Dailey: "Proud, proud papa of young baby" is a better characterization.

ICDN: Do you believe the whole concept has been proven or are there still some things to prove?

Dailey: It's a set of protocols. Proof is in the usage, in a sense. An analogy is how TCP/IP was chosen. Some years ago there existed a bunch of protocols, TCP/IP, DECNET, AppleTalk, IPX, and to this day, many of those still exist. Most folks picked TCP/IP because people from MIT gave away some implementations and eventually Microsoft sold it and then everyone used it. So, there was nothing wrong with the other protocols. They just weren't adopted as thoroughly. They are still in use but only in little corners.

ICDN: And so you'd like to see yours in that same TCP/IP model?

Dailey: We're giving it away with that same notion, that if you give something away as a first implementation of it, people will try it out and if it does what they need, they'll use it. And, of course, if it doesn't, they won't.

ICDN: What are your future plans for SDD? Where do you see it going? You mentioned your primary involvement so far has been with the WSDOT projects, right?

Dailey: Right. I've had discussions with people from other universities who are doing data communication in other science areas, such as forestry. The notion applies to any sort of dynamic data; it's not specific to transportation data. We just happened to come across it in that domain.

ICDN: Did you invent the notion itself that you combine the description with the data?

Dailey: In terms of data flow across the network, yes. In terms of earlier work, there was earlier work for self-describing databases by Leo Mark, who is now a faculty member in Maryland, I believe. I don't think he called it "self-describing data" because he had no notion of transferring data. He had the notion of creating databases that had this property. We think we invented it but you know, there's nothing new under the sun. I know that we've sent it out for publication and no one has said 'Oh, yeah, Joe did that before.'

ICDN: Have you patented it, to the extent you can patent software?

Dailey: No. It is copyrighted through the UW.

ICDN: The SDD concept theoretically means that ITS information service providers, "ISPs," could essentially provide new services in new cities very easily if they've already hooked into the data in another city, is that correct?

Dailey: That was our intent.

ICDN: So that's the allure, the attractiveness of the overall concept?

Dailey: That's our intent, but only if we can get other cities to adopt it, so that's why we're giving it away.

ICDN: As far as you know, do you think that any other cities are on the brink of adopting it, or is it still too early to tell?

Dailey: I don't have an answer to that question. We've had some discussions with New York. We've had discussions with Arizona. What it comes down to is we haven't given away a robust transmitter yet. Until we give the transmitter away and they can see that it's easy to give their data away that way rather than some other way, there's no way they can evaluate it. The transmitter is done, but it's still being tested. We'll be giving away a transmitter this fall. Call me in December — I'll be in a better position to answer that question then.

ICDN: Is the transmitter also written in Java?

Dailey: Yes, it's also in Java. We had problems doing it in other languages because certain system calls are necessary if you use C or other languages that aren't portable across platforms.

ICDN: Is there anything else germane to SDDs you can think of that would be important to people deploying systems?

Dailey: The SDD concept was originally implanted in C, later C++, then Java. So, we're not tied to a language implementation. Java was the popular language that gave us platform independence just now, but there's no reason you would have to adopt Java for your agency to be able to use this notion. I would like to separate those two things. Because we did a Java implementation, we put an object-oriented overlay on the concept of self-describing data, so that we'll be in line with the National Architecture as it moves to a more object-oriented view of the world. There's nothing inherent that has to be object-oriented about this idea — this is a concept for data transfer.

ICDN: We've talked about the fact that SDDs potentially let ISPs grab information that the public sector is putting on the Net and use it however they want. Can a number of sources combine data into one data stream?

Dailey: Yes. That is a layer above SDD that involves operating on the data and putting it back together. If you look at our SDD distribution site, you'll see a model for how data comes in and goes out. Our ITS Journal article *(http://www.its.washington.edu/pubs/its_jour.pdf)* [Note:

requires an Adobe Acrobat Reader *(http://www.adobe.com/products/acrobat/readstep.html)*] talks about an architecture for building ITS applications with reusable components. It shows a model where these components would plug together and might include something that we'd call a "data fusion element." A data fusion element might put 2 data streams together and output the sum of the streams. SDD would handle the data flow through the components that operate on that data.

ICDN: Before we close, I'd like to get back to your comments about how the private sector might view the SDD concept. You were saying that the private sector might not fully embrace the concept?

Dailey: The SDD concept is very attractive to the public sector because it provides a way of giving data away to everyone on an equal footing. However, it's a mixed bag from the private sector's standpoint. On the positive side, they could conceptually go into any new market and gain access to data quickly. On the negative side, not having control of the data in any market could be a disadvantage.

ICDN: That is a very interesting quandary.

Dailey: My understanding is that the business model for several private sector companies is to control the data in each of the market's they go into. They're going to standardize by getting a license to be the sole distributor for data. That's a very different market model than one that would create a level playing field for lots of players. However, people in the private sector have told me that they are convinced that the market won't bear open competition with a large number of players.

This web page created by the National Associations Working Group for ITS (NAWG) *(http://www.nawgits.com/)*, a cooperative effort of organizations whose members are spearheading ITS deployment in the U.S. The NAWG makes every effort to ensure the accuracy of these pages, although errors can an do occur. Report any errors or omissions to the NAWG webmaster (nawgwm@nawgits.com). Each participating member of the National Associations Working Group for ITS is responsible only for the information it provides.

DESCRIPTION: The National Associations Working Group for ITS' (NAWG) ITS Cooperative Deployment Network (ICDN) is a cooperative effort to share and exchange information on Intelligent Transportation Systems (ITS) through a shared Internet resource for the benefit of NAWG members, produced by the Institute of Transportation Engineers (ITE) and its agent, Communication Alchemy, Inc, and funded by the U.S. Department of Transportation.

DISCLAIMER OF OPINIONS AND VIEWS: The views and opinions of document authors do not necessarily state or reflect those of the U.S. Government or any agency thereof or the Institute of Transportation Engineers, or Communication Alchemy, Inc., or members of the National Associations Working Group for ITS, or any members of the National Associations Working Group for ITS' ITS Cooperative Deployment Network, or any of their directors, officers, employees, agents, and members. The information provided and the views and opinions of opinions expressed in material submitted to the ICDN are those of the participant organization attributed to the material posted on the ICDN's shared Internet resource.

DISCLAIMER OF LIABILITY: Neither the U.S. Government nor any agency thereof, nor the Institute of Transportation Engineers, nor Communication Alchemy, Inc., nor members of the National Associations Working Group for ITS, nor any members of the National Associations Working Group for ITS' ITS Cooperative Deployment Network, nor any of their directors, officers, employees, agents, and members makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

DISCLAIMER OF ENDORSEMENT: Reference to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government nor any agency thereof, or the Institute of Transportation Engineers, or Communication Alchemy, Inc., or members

of the National Associations Working Group for ITS, or any members of the National Associations Working Group for ITS' ITS Cooperative Deployment Network, or any of their directors, officers, employees, agents, and members.

This web page created by the National Associations Working Group for ITS (NAWG), a cooperative effort of organizations whose members are spearheading ITS deployment in the U.S. The NAWG makes every effort to ensure the accuracy of these pages, although errors can an do occur. Report any errors or omissions to the NAWG webmaster. Each participating member of the National Associations

Working Group for ITS is responsible only for the information it provides.

Self-Describing Data (SDD) for Dummies

Smart Trek

For Those who Would like to Understand More about SDD, but were Afraid to Ask

Version 1.4 November 17, 1997

Mark A Whiting
Smart Trek MMDI
1.509.375.2237
Mark.A.Whiting@pnl.gov

The Smart Trek SDD Development team is led by Dr. Daniel Dailey at the Univeristy of Washington
See: http://www.its.washington.edu/bbone/ for more information.

Self-Describing Data for Dummies

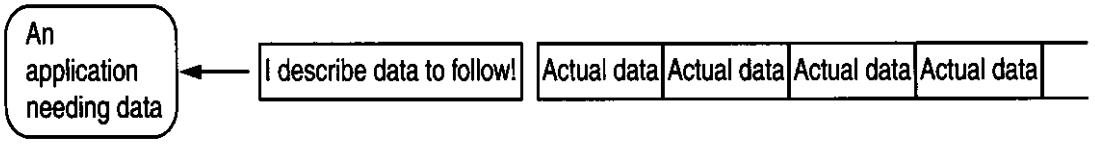
Introduction

Self-Describing Data (SDD) is an approach to transmitting and delivering data, such as transportation data, where a stream of actual data is prefixed with a set of data that "describes" it. This descriptive data can be anything. Yet the intent is to provide descriptive data that helps users, and applications, to understand the actual data that follows. An example data stream is the traffic loop sensor data collected by the Washington State Department of Transportation (WSDOT), where over two thousand sensors provide new data every twenty seconds. The descriptive data for this stream consists of descriptions of the sensors, their cabinets, their locations, and so on. This preface plus the continuous actual data stream create an SDD stream.

SDD is often difficult to understand as it is a new approach to data packaging and transmission and because the approach is still evolving. This document attempts to provide an introduction to SDD for those of us trying to keep up.

What is Self-Describing Data?

Self-Describing Data (SDD) is a data stream that consists of descriptive data, followed by a continuous stream of data. Figuratively, an SDD stream would appear as:



In terms of traffic data, this stream might appear (using English prose):

> "You'll be getting traffic loop sensor data; that belongs to these cabinets X, Y, Z; from locations A, B, and C. The format is "loop sensor name": 8 bytes, "occupancy": 16 bytes, and "volume": 8 bytes."
>
> "XXX-4583" 50 3
> "XXX-4587" 43 2
> "XXX-3848" 48 2
> ...

More on the SDD format shortly...

What Kind of Actual Data Should be Made Self-Describing?

Any kind of data can be delivered as self-describing. But if you have an unchanging (static) set of data, such as a set of reports, or a very slowly changing set of data, such as the King County Metro Transit schedule data that is updated only three times a year, it may not be worth encoding and delivering this data using the SDD approach.

It may be that the best kind of data to be delivered in SDD format is real-time (or near-real time) data streams such as the WSDOT traffic loop sensor data or the King County Metro Transit bus location data. The data elements of these streams are constantly changing in content (remember loop sensor data is updated every 20 seconds), and they have a rich set of descriptive data that can be used by applications in providing traffic management information, traveler information, and so on.

About the Description

The real-time or actual data may be very simple in structure. Even if the actual data is simple, the descriptive data component of an SDD stream may be quite extensive and complex. This complexity is a confusing aspect of SDD. Elements of a simple data stream may have a lot of hidden or implicit meaning (We'll see an example of this below.). If users do not know this, data can be misinterpreted or misused. SDD tries to make more of this implicit meaning available through the descriptive data. Because of this potential complexity, descriptive data is encoded as a**database**, using database structures and language. This was designed so that applications could use relational database technology to manage large and complex data descriptions, as opposed to having to derive ad-hoc management schemes. The database is sent in two parts: a specification for the structure (or "schema") and the contents. So, changing the English prose we used above around a little bit, the stream would more closely look like:

> "Create one data base table that contains descriptions of the traffic loop sensors. Include a field for loop_id (16 bytes), a cabinet_id (7 bytes), etc. Create another database table that contains descriptions of the loop data. Include a field for sensor_id (15 bytes), volume (int), occupancy (int), etc...."
>
> "Put the following data in the Sensor table:
> "XXX-3848", "YY1", ...
> "XXX-4959", "YY2", ...
> Put the following data in the Loop table:
> ...
> ..."
>
> "XXX-4583" 50 3
> "XXX-4587" 43 2
> "XXX-3848" 48 2
> ...

This gives us a specification of the data base tables that describes the traffic loop sensors. This is followed by the contents to insert into those tables. Finally, the data stream is sent.

The descriptive data, that the SDD development team calls a **data dictionary**, is not actually encoded in English, but in a database language called SQL-92. SQL-92 is a standard database language that will allow developers to pass these commands directly to a database and have it understand and perform the appropriate actions (hopefully, create the database and store the values). In SQL-92, our English prose above looks more like:

```
CREATE SCHEMA   <<<this is the structure or schema definition>>>

CREATE TABLE SENSORS
(LOOP_ID CHAR (16),
CABINET_ID CHAR (7),
...
)

CREATE TABLE LOOP_DATA
(SENSOR_ID CHAR (15),
VOLUME SMALLINT,
OCCUPANCY SMALLINT,
...
)

TABLE SENSORS
COLUMN (LOOP_ID, CABINET_ID...)  <<<this fills the tables we created >>>
"XXX-3848", "YY1"...
"XXX-4959", "YY2"...

TABLE LOOP_DATA
...

"XXX-4583"  50  3    <<<then finally the data...>>>
"XXX-4587"  43  2
"XXX-3848"  48  2
...
```

This specification is the same as the previous one, only written using SQL-92 language[1]. The CREATE instructions would allow a database to create the specified tables. The TABLE specification with its COLUMN dependents allows the database to fill the tables with the included data. Finally, the data stream trails as before.

---

1 Actually, the section that describes the contents of the data dictionary is written in a SQL-like language that the SDD development team calls their "Content Language." The Content Language specifications are translated into SQL-92 by the SDD software.

More Wrapping for SDD

The story of the SDD format doesn't end with SQL-92. SDD streams are packaged much like Christmas presents with many layers of wrapping paper. To be compliant with a computing standard known as **ASN.1**, the data dictionary and the actual data are wrapped using Basic Encoding Rules (**BER**). Why ASN.1? (Which, by the way, is pronounced "ay-ess-enn-dot-1.") It's much too long a story for this paper, but it is a standard the National Architecture and other ITS standards groups prefer. Using BER means give each of the three components of SDD a type, a length and a value. The SDD development team has created software that includes a BER encoder on the SDD transmission side, and a BER decoder on the SDD receiver side. The SDD development team has created software that does this automagically, so we really don't have to worry about it too much. But if anyone asks if SDD is ASN.1 compliant, we can say "Yes!"

The BER-encoded SDD streams are then also encoded into "ITS Frames." An ITS Frame is a package developed by the SDD development team to make the network transfer easier. There is an ITS "Framer" at the SDD transmission side and an ITS "Deframer" on the SDD receiver side. The SDD code performs this work automagically as well, so we really don't have to worry about it either. Below is a diagram of the flow of SDD from a source generating data to a sink that would use it.

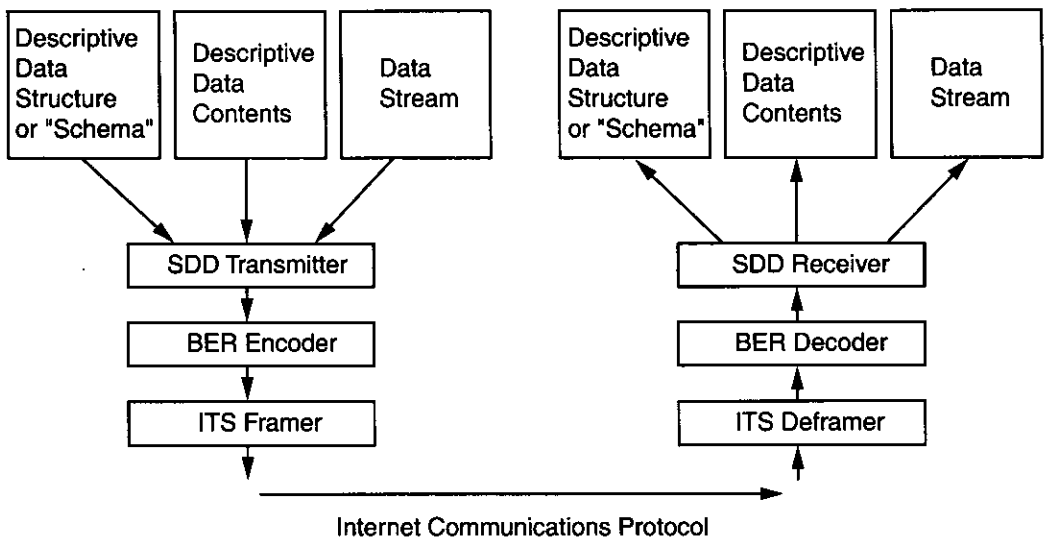


Figure 1 The flow of an SDD stream

What does an SDD stream look like that is BER-encoded and ITS-framed? You really don't want to see it. It's best to just think about SDD in terms of the original three components: the Data Dictionary Schema, the Data Dictionary Contents, and the actual Data.

What goes into a Data Dictionary?

Remember: *There are no rules as to what must be in an SDD data dictionary.*

We gave some examples above as to what might go into a data dictionary. They are not far from what has actually been encoded into one. The SDD development team has created extensive data dictionaries for WSDOT traffic loop sensors and an SDD stream is available with this data for ISPs and other interested parties to use. Let's look at that dictionary—it has 8 tables.

Here are the tables to be defined as part of the database:

COORDINATES - This table describes coordinate data types, such as "geodetic"; and their measurements, such as "longitude" and "latitude"; and their units of measure, such as "degrees".

MEASURES - This table provides additional data on coordinate data types, such as the fact that WSDOT uses NAD23 coordinate referencing, while UW uses NAD89.

STATION_FLAGS - This table provides flag values as to whether or not data is usable.

INCIDENT_DETECT - This table provides flag values as to whether an incident has occurred or not.

CABINETS - This table provides cabinet IDs, freeway names, text descriptions, and whether there is a ramp or not.

CABINET_LOCATION - This table provides the location of cabinets. It includes the cabinet ID, the coordinate type used (see above), and whether the data type is defined using the WSDOT or the UW methods, and the location.

LOOPS - This table describes the loop sensors. It contains the loop ID, the cabinet ID, whether or not it's metered, the road type, the direction of the traffic, the lane type, the lane number, and the sensor type code (a number).

ALG_DESCRIPT - This table provides a complete listing of Java code that will extract the loop sensor data!

As you can see, there is a lot of data here about the data. Heck, there's even the code to use it. If you didn't have access to this descriptive data, you would not automatically be aware of information such as locations and sensor types and so on. SDD makes that implicit data explicit and available. The user of SDD is not forced to use any of this descriptive information, but since it's being provided...why not?

When Do You Get a Data Dictionary and When Do You Just Get Data?

The data dictionary is sent upon connection to an SDD transmitter. An SDD transmission rule is that the data description remains valid until something changes in the data. So the data dictionary is sent once and is not sent again until there is a data change. When a data change does occur, a new data dictionary is sent through the stream. Again, if a data stream were being processed, and a change occurred in the data, the following might appear (going back to the English prose):

"XXX-4583" 50 3
"XXX-4587" 43 2
"XXX-3848" 48 2
"XXX-4589" 43 2
"XXX-3850" 48 2

"You'll now be getting traffic loop sensor data; that belong to these cabinets X, Y, Z; from locations A, B, and C. The format is 'loop sensor name': 8 bytes, 'occupancy': 16 bytes, 'volume': 8 bytes, and 'validity flag': 1 byte."

"XXX-4583" 50 3 1
"XXX-4587" 43 2 0
"XXX-3848" 48 2 1
...

The change that occurred was that a new data element was added to the stream called "validity flag." The reason for the new description is to allow users and smart applications that use SDD to handle the data change gracefully and perhaps even adapt to the change and continue functioning normally and without significant loss of service.

How Do You Know Whether or Not You Received a New Data Dictionary?

That is, without doing a lot of searches, data comparisons, and so on. It wasn't mentioned before, but the SDD stream also includes a time-stamp that is updated when the data dictionary changes. If an application sees a new time stamp, it knows something has changed. And yes, the SDD development team remembered to use four digits to represent the year portion of the time-stamp.

Software for SDD

We've mentioned an SDD transmitter and an SDD receiver, in terms almost like a television station transmitter and a home TV receiver, and this is just about how they work. Most ISPs are concerned with the "TV receiver" part of this setup, as they want to get traffic data. They are both pieces of software that projects can use to send and receive SDD. The receiver is the more mature piece of software at this writing, so we will review this.

The SDD receiver is a Java software program that can receive and understand the Data Dictionary and receive and extract the actual data from the SDD stream. The receiver has some quality checks in it that make sure the SQL-92 language is in the right format and the schema part is in sync with the contents part. The SDD receiver's primary job is to generate two files:

> An ASCII file that contains the SQL-92 code that can create the database for all of the descriptive data. This file can be given to a database system, such as ACCESS or SYBASE, to import the data. An ASCII file that contains the actual data.

So what do you do with this, if you're an ISP? Write software that can dump the Data Dictionary SQL-92 code into a data base, and write other software that can do something interesting with the data stream. That's the basic stuff. More advanced efforts will work with the newly formed database to provide very cool information to traffic managers or travelers.

How Do I Tap Into an SDD Data Stream?

SDD is delivered over the ITS INFORMATION BACKBONE (a.k.a. the "I2B", pronounced "eye-too-bee"). You plug into it using an Internet (TCP/IP) connection. Programmers know how to do these things. The address for the loop sensor data stream is "sdd.its.washington.edu" at port 9033. This provides the loops data on a 20-second cycle.

Want to see if this connection is for real (and you're not a programmer)? Open up Netscape on your PC and enter the URL: http://sdd.its.washington.edu: 9033/. Quickly, you will see a lot of data dictionary filling your screen. You won't be able to see the actual data decoded correctly, but you'll be able to tell if the port is active or not.