



## Hansel minutes

Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

### Text transcript of show # 41

November 13, 2006

#### WS-\*

This week Scott and Carl discuss the various WS specifications and how they relate to WCF, with a strong emphasis on security.

(Transcription services provided by [PWOP Productions](#))



#### Our Sponsors



<http://www.nsoftware.com/>



<http://dotnet.con-sys.com>



<http://www.codesmithtools.com/>





(Music)

**Lawrence Ryan:** From [hanselminutes.com](http://hanselminutes.com), it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman, hosted by Carl Franklin. This is Lawrence Ryan announcing Show #41, recorded Wednesday, November 15<sup>th</sup>, 2006.

Support for Hanselminutes is provided by CodeSmith Tools, makers of CodeSmith – an extensible template-based Code Generator for .NET. Hanselminutes listeners get \$100 off CodeSmith Professional with Coupon Code HM100, online at [codesmithtools.com](http://codesmithtools.com), and by /n software Red Carpet Subscriptions - the most comprehensive solution for adding connectivity to your .NET and ASP.NET applications with components for every major Internet Protocol, online at [www.nsoftware.com](http://www.nsoftware.com).

Support is also provided by .NET Developer's Journal – “The World's leading .NET developer magazine”, online at [www.sys-con.com](http://www.sys-con.com). In this episode, Scott and Carl discuss the specifications of WS-STAR.

**Carl Franklin:** Hi, this is Carl Franklin and you're listening to Hanselminutes. From Hanselminutes.com, I am here with Scott - as we try to be every week - we try.

**Scott Hanselman:** We do; we try, in fact.

**Carl Franklin:** Sometimes we go away; it's hard, but we're here now.

**Scott Hanselman:** It's hard producing a show; you put a lot of work into this.

**Carl Franklin:** It's true, we do. Speaking to -- I don't mean to waste the listener's time; but Rory has sent me some emails lately. He went looking through iTunes for the first time at all the other podcasts out there.

**Scott Hanselman:** Okay.

**Carl Franklin:** And he says they basically suck.

**Scott Hanselman:** Huh!

**Carl Franklin:** Hah! Okay, so what's the topic today?

**Scott Hanselman:** I wanted to talk a little bit about the WS-\* specifications. I think a lot of people have worked with Web Services, but most people are doing Web Services in a File, New, Web Service kind of a way from Visual Studio; and while a number of people have messed with

WS-\*  
November 13, 2006

WSE, the Web Services Enhancements, and more and more people are getting into WCF, I thought that it would be an interesting thing to spend a few double-speed 20 minutes of the listener's time to talk about these different specifications that people keep referring to as WS-\*.

**Carl Franklin:** Hey, you know, Scott, the first thing that I hear from people when we are talking about Indigo and all of this stuff is, “Isn't WS-\* like old technology?” And I think there's a misconception out there that just because we had toolkits for WS-\* before and now we have Indigo that somehow we don't need to pay attention to these. So let's clarify that point.

**Scott Hanselman:** So WS-\* refers to all of the WS-draft specifications, most of which are associated with WS-Security. And these are all things that make SOAP work better, but you kind of have to back up quite a bit to the fundamentals. So there's XML; but more specifically, there is what's called the Post Schema Validated Infoset, or the PSVI. We all know that with XML and angle brackets if you decide to go and say, “<fo/>”, what does that mean? That means an empty element called “fo”. And that is semantically equivalent to having a “fo” element and then an “endFo” element, right? That's kind of the obvious example of ‘these two things mean the same’.

**Carl Franklin:** Right.

**Scott Hanselman:** If you did a diff between two documents that each had these two kinds of elements, if you did a byte-by-byte diff, you would see that they are different; but if you did a semantic diff with like an “s smart XML diff”, you'd see that they are semantically the same. When you apply Schema to an XML document and stop thinking about angle brackets and think about the -- after it's been Schema validated, the PSVI, the Post Schema Validated Infoset, the information set that is associated with that XML, regardless of the angle brackets in the serialization format and how it actually hits the wire, then you start having something interesting. Now, that's XML, and that's the DOM and some of the basics of XML. But what had to happen before we got into the WS-\* specifications was, of course, SOAP. But before that, we've got XML Signature. XML Signature, or XML-DSig Digital Signature, was a specification that described how one could mark an element with an XML with a cryptograph of the significant digital signature to prevent the tampering of that particular element. So I could give you an XML document that maybe had a ‘person’ element and underneath ‘person’ we had ‘first name’ and we had ‘last name’, and I could



then apply an ID to -- like maybe an ID with a GUID to the person element; and then later on somewhere else in the document, I could have a digital signature that refers back via what's called an IDREF, an ID and IDREF, or kind of the primary key and key reference of the XML set. And I could say, "Here is a digital signature associated with the information set represented by 'person'." So that if somebody tampered with 'person', I could tell, because it was signed and I could see that someone had messed around.

**Carl Franklin:** Sort of like a hash or equivalent.

**Scott Hanselman:** Right. Then on top of XML's signature and the techniques for applying signatures to elements within documents was 'XML Encryption', which would let you apply a cipher value and say, "This information set contained in this element and from this element and down is now represented by this XML block."

**Carl Franklin:** And this is all way before SOAP, right?

**Scott Hanselman:** Right. So WS-Security, among other things, takes things like XML Signature and XML Encryption and then integrates it with SOAP. So when I was talking a moment ago about signing an ID -- that ID would be in a different namespace, because you don't want to actually change the information set represented by the 'person' element. One of the things that WS-Security would do is it would say, "I'm going to put this information about the digital signature up in the header." So it calls out the difference between the Body of the message and the Header of the message and indicates a way to potentially encrypt or sign chunks of a document arbitrarily. So this means that I could potentially create, let's say, a document that represents an invoice or a purchase order, and let's say that it included payment information. Right now, most of our listeners could sit on Notepad and kind of bang out a document like that and come up with a general sense of what they think that it will look like, or maybe their companies already have an XML Schema to do that. But what something like WS-Security allows you to do is that if you were passing that combination payment information and invoice via SOAP, you might want to route that document such that the invoice was digitally signed to prevent it from being tampered with, and then you might want to actually encrypt the payment information with a different key; and this information about what was going on would go in the header, and then you might, in the essence of the encryption aspect, remove that whole payment bit and you'd insert in a cipher value. So the information set is still there; it's just been

encrypted. What this would mean is that I could route that document through multiple departments and wouldn't have to be concerned about information that one department might want to see. I don't necessarily want to allow the payment information to be seen by the department that manages the inventory.

**Carl Franklin:** Encryption is all about protection in transit.

**Scott Hanselman:** Exactly right. So these XML Digital Signature and XML Encryption techniques and Draft Specifications, when applied the way that WS-Security describes them, allow you to pass these documents around - these kind of composite documents, and use it in very, very flexible ways that really match the way your business works.

A good physical analogy to this would be like, in Oregon, we do vote-by-mail, so like we just had our big midterm elections in the States, and everyone said, "When did you vote? You were out of town, how did you do that?" Well, in Oregon, we can vote by mail.

**Carl Franklin:** Absentee Ballot, we do. I did the same thing before that.

**Scott Hanselman:** Absentee Ballot, it's more of a -- everyone votes by mail; you don't actually go to the...

**Carl Franklin:** Well, that's cool.

**Scott Hanselman:** So they send us a ballot ahead of time, and you fill it out and you send in your information. And there's a secrecy envelope; so there's an envelope and there's also another envelope. So I can have multiple things in that envelope, and if you tamper with one, you can tell; so there's actually a separate envelope inside the main envelope. The main envelope is addressed; the one inside is done for secrecy purposes. So in case someone opens the outer envelope, they won't accidentally open up the inner envelope.

So what I can do with things like WS-Security is create a large envelope full of other tiny envelopes, some of which have like a wax stamp -- remember when people used to...

**Carl Franklin:** Sure, sealing wax.

**Scott Hanselman:** ...seal off a message and it had like a stamp done in a wax symbol from the king, and they would indicate that this particular envelope had been tampered with. That's what you're doing.



**Carl Franklin:** Sealing wax, they call it.

**Scott Hanselman:** Sealing wax, exactly right, thank you for that. So WS-Security is the 'how to do sealed wax envelopes and documents' within the context of SOAP using XML Signature and XML Encryption. So we've all seen WSDL. WSDL, of course, is the Web Services Description Language that describes what a particular service can do. WS-Policy is a specification that is kind of like WSDL, but it describes policy of a particular web services' endpoint. So it would say, "Well, if you're going to be talking to us, our security policy is going to indicate that you need to use triple DES and you require a client certificate if you're using HTTPS, so the trans word has to be a certain way"; it's kind of WSDL policy.

**Carl Franklin:** Yeah.

**Scott Hanselman:** So WSDL is the 'here is how to call it', and the WS-Policy document is a separate document that I suppose could come along for the ride inside of WSDL that describes all of the different policies around how we want things handled if you're going to be doing this - all the security policies.

**Carl Franklin:** Just about now in the speeches when I hear people clicking the Stop button on their iPods and saying, "Ahhhhh" like Lurch on the Addams Family, right, because when we think of WSDL and WS-Policy, we've all seen those files, and they are cryptic and difficult to understand at first glance. And we might want to just reassure the listener that nobody expects you to write this stuff by hand.

**Scott Hanselman:** Yeah. For the most part, if you're using a decent toolkit, just like a good Web Services toolkit, we'll hide the WSDL from you. Indigo is doing its best, now WCF is doing it's best to hide these things from you.

**Carl Franklin:** And it does a really good job, I mean very high...

**Scott Hanselman:** I still think that an awareness of this is important.

**Carl Franklin:** That's right.

**Scott Hanselman:** And I know that people will probably disagree with me, but I just think it's so important for someone to go up to [w3.org](http://w3.org) and actually read the submissions. I mean, there are submissions, there are Draft Specifications; a lot of people put a lot of good work into these things. These are joint specifications written by, like,

Microsoft, IBM, SAP, BEA. This is not just Microsoft saying how the things ought to be; they wrestle, and they're not particularly hard to read; they really aren't.

**Carl Franklin:** Yeah.

**Scott Hanselman:** And like I am looking at the WS-Policy right now, and it's fairly well organized: it's got examples; it explains what we're trying to accomplish. Yeah, there are people like Tim Bray, who kind of invented XML, who say that this is way too complicated and people complain; but it's not that complicated, right? There is a difference between being difficult and being complex.

**Carl Franklin:** Well, and hopefully that's why you're listening to this show so that you can explain it to us.

**Scott Hanselman:** Yeah, and not just because I can explain all the WS-\* things in one 20-minute podcast that's being played in double speed while someone races down the Autobahn or wherever they are listening to this. But rather than -- what I am trying to accomplish by talking about it in this particular podcast is, rather than just saying "WS-\*" and not really knowing what that means, understanding that these are layers right? They are layers that came very quickly, though, and I think that's what's confusing, because we -- for the longest time there is XML, and then there is Schema that's built on top of things like DTTs and DCDs, and then everyone pretty much understood that. And then we started talking about 'document literal', passing around documents. Then you start thinking about Message Passing, request/response messages and that in the context of SOAP - SOAP being just an enveloping technology and a way to define Headers; that's pretty much all SOAP is; it's like, 'Here is what an envelope looks like' -- and then suddenly we started getting WS-Security, WS-Secure conversation da, da, da, and it got very confusing. Yeah, there are things you probably won't use for a while, like WS-Atomic Transaction and things like that; but the things that you are going to want to understand are things like WS-Security, you are going to want to understand what's called WS-Metadata Exchange.

**Carl Franklin:** Let's talk about that one for a sec.

**Scott Hanselman:** So WS-Metadata Exchange is what they call 'WS-MEX', and this is a way to describe what endpoints need to know when you want to interact. So WS-Policy that we just talked about is the 'describing a characteristic of a web service' like what that web service -- what its



capabilities are, what its requirements are. Well, WSDL describes things like the endpoint address, like where you can find that web service; it describes the messages and the operations that are supported and some of the network protocols that you would potentially want to use - 'this web service uses HTTP' - that will go in WSDL. WS-Policy might say, 'And you really need to be using this kind of security and these client certificates' when you do that.

**Carl Franklin:** So WS-Policy - this isn't really about business rules, is it?

**Scott Hanselman:** Well, it's a good question; it kind of depends. If the business rule says this ought to be encrypted and here is how, WS-Policy is a decent way to describe that in a programmatic sense.

**Carl Franklin:** So where does it end?

**Scott Hanselman:** It's all about discovery, right? I want to be able to ask the Web Service, "So what exactly is it going to take for me to talk to you?"

**Carl Franklin:** OK. So the policy says you must be using this certificate and that certificate, and you must encrypt this payload and that payload.

**Scott Hanselman:** Right, right. So WS-Metadata Exchange is a specification that describes how we're going to work these -- kind of these dialects out; how we're going to say, "Well, I speak WS-Policy, the one that came out in September of 2004, and here is how I want to express that bit of Metadata." It's not something you ordinarily need to worry about, now that WCF kind of handles this for you; but it's a way of describing a bit of Metadata. It's a way of collecting units of Metadata, like WSDL Policy Schema, and saying, 'All of this applies to my endpoint.'

**Carl Franklin:** Okay.

**Scott Hanselman:** So you've got signatures, you've got encryption, you've got things like Metadata Exchange to describe things like policy and kind of tie it all together. So I have some endpoint out there that is going to speak Web Services; and now I want to communicate back and forth using Private keys - Public/Private keys. So Carl Franklin is a service, and I will -- and then I am Scott, I am going to talk to the Carl Franklin service. I am going to take my message and I'll encrypt it using your Public key and my Private key, right? -- so we're going to maintain whether or not it's been tampered with -- and I am using information about you to make sure that things are decryptable by you.

Now, WS-Trust is a specification that describes the interface for a security token service. We talked about STSs in our talk about InfoCard. So Microsoft's InfoCard or Cardspaces is the client side of a WS-Trust relationship, okay? And the trust that you have between yourself and another service is represented by that exchange of tokens. So I am going to go and say, I have some claims about myself; I claim that my first name is Scott and my last name is Hanselman, and I have a website here. That claim can be represented by a URI like a unique identifier. Most people are using URLs for this, but it can be -- it's just a Uniform Resource Indicator or Identifier.

**Carl Franklin:** Identifier, I think.

**Scott Hanselman:** Identifier yeah. And I am going to talk to a particular security token service that we might share. So, like, let's say Scott calls the Carl Franklin Service and the trust relationship has to be set up ahead of time; what a security token service is, it's a third-party trusted thing with a known interface that we can use; you trust him, I trust him, we trust each other. So then I could also call the Richard Campbell Service; so there is the Carl Service and the Richard Campbell Service and there is some security token service that is going to be handing out tokens.

So what WS-Trust is, it is the thing that lets the applications construct these message exchanges in a trusted way. So I can go and say, 'Hey, security token service, I have this claim I want to make about my name or maybe I want to' -- ordinarily you make a claim about either your identity or that there is an aspect about yourself that you feel is important. And it will return back a token; that token is a collection of those claims and if you sign that token then you've got some authority that is going to cryptographically endorse that token and say, "Well this certificate has been applied to this token." So now I've got a security token that I can make sure is trusted and it's not been tampered with. And the security token service, that third party service there might be handled by you, it doesn't necessarily need to be a third-party. It's that web service that gives tokens, okay? So, it is the thing that will -- based on evidence that it has about trust, you provide it with some proof, like, I might give it a username and a password, or username and a password and something from a Smart Card, and then it will return a token; and it would generate those tokens. So, in the context of Cardspaces, like we were talking about last week -- or a couple of weeks ago, when you self-issue a CardSpace card, there is a Security Token Service; it's just



on your machine. So you're basically telling yourself you trust yourself; and that's where a self issued token comes from; is a little, tiny, local Security Token Service running.

**Carl Franklin:** That's fine for you, but somebody --and I don't know big wide world gets that, they're not going to know if they going to trust it or not.

**Scott Hanselman:** Exactly. And that's why we're hoping that like an Amazon or Visa will start being a Security Token Service. So, a Security Token Service is a web service that speaks WS-Trust.

**Carl Franklin:** Have VeriSign or any of these certificate vendors gotten on board with web service based tokens?

**Scott Hanselman:** Everyone's on board, but I don't -- no one has made a public Security Token Service out there for the world yet.

**Carl Franklin:** Dude! Scott's Security Token Service; I can see it now. You are in the wrong business man.

**Scott Hanselman:** SSTS - I trust everybody. So there are these different ways that one could say, "I am in a brokered trust." You could have like Direct Trust where I am going to be the relaying party and I am going to accept all of your claims, or maybe I'll only accept some of those claims, or maybe I'll be vouching for someone. I could -- you trust me and then we could kind of apply the transitive property of trust. You trust me, I trust Richard Campbell and he trusts you by a direct brokered trust.

**Carl Franklin:** Right.

**Scott Hanselman:** You only want to make sure that only one shows up. So, the requester - that would be like me who is going to call the Carl Franklin Web Service, would go the Security Token Web Service and make some claims and say, "I claim these things and do you trust me?" And based on their policy and their tokens they'll return back a token to me that then I could hand to you; and then you could then talk to the Security Token Web Service and find out whether or not you buy it, whether you believe this is the case. And then these trust relationships can be set up ahead of time, or we could have 'Dynamic Trust' where you would trust me because of information that you received from that token service. So, Indigo, WCF understands these things out of the box. Ordinarily, you want to see any of these angle brackets, but you will spend a

lot of time, not in the DLL Hell, but you'll spend it in config file hell.

**Carl Franklin:** So in WCF, do we have configurators? Do we have wizards? Do we have tools to make it, so that we don't have to touch the GUI, or do you actually have to get in there and do it?

**Scott Hanselman:** Well, yeah, so yes or no, that's a great question. There are some - 99% of things, there's a config file for. We bumped into a few things where we have to use the object model that comes with Indigo in order to make some changes. 99 times out of 100, there's some aspect of things in Indigo and WCF that you can get through the config file, you can replace everything. If you don't like the way it does something, you get out and say, "I'll do that."

**Carl Franklin:** Yeah, I do like that about WCF; it's modular, and if you want to do something better, you can just plug it in.

**Scott Hanselman:** But you do have to spend a lot of time in the config files. There are some WinForms applications that will do that, I've seen a couple. Chey Cowen's shown me some. I think more and more you'll see tools to do that. When you plug in some tools into Visual Studio, that will let you get to those things. But I haven't seen kind of, the uber-configurator for all things Indigo that let's me never touch an angle bracket in sort of a config file yet. But maybe a listener will educate us to that.

**Carl Franklin:** Yeah, that's what we need; all right tool vendors, are you listening out there? That's it.

**Scott Hanselman:** Yeah. So that's basically the WS-\* that you need to know about in 20 short minutes.

**Carl Franklin:** Yeah, that was good I think.

**Scott Hanselman:** There's a lot more, but don't be afraid to read the specs, and it is sometimes hard to get the big picture, but if you look at Michele Bustamante's blog, "That Indigo Girl" - I look at her blog, look at the guys like Ingo Rammer, watch the videos on Channel 9 about WCF - they're really good.

**Carl Franklin:** Yeah, and get to one of the iDesign training sessions too, those are great.

**Scott Hanselman:** Oh, yeah. And actually he just released -- that's a great point -- Juval Lowy at [Design.net](http://Design.net) just released his WCF coding standard - just go to the homepage, it's right



WS-  
November 13, 2006

there; and it's very good, it's a description of how he says you should write your WCF services for simplicity, ease of calling, easy of configuration, ease of use.

**Carl Franklin:** He spent a long time thinking about this and working with the team too. Also, he's got his big mammoth example WCF sample code collection.

**Scott Hanselman:** There's so much stuff up there; if you go up to -- just go to [idesign.net](http://idesign.net) and click on downloads; he's got whole huge sections on WCF - example after example after example. There's really more examples there than there are really anywhere else including Microsoft's own site.

**Carl Franklin:** And Juval is so modest, too.

**Scott Hanselman:** Yeah, and he'll tell you that.

**Carl Franklin:** Yeah. All right, Scott, that's a show.

**Scott Hanselman:** Fantastic.

**Carl Franklin:** Thanks a lot. I know you're not feeling all that great today, I am sort of coming down with a cold, too; and we apologize for the lack of links, but I think it was a good primer anyway.

**Scott Hanselman:** I think so.

**Carl Franklin:** Thank you, Scott. And we'll see you next week on Hanselminutes.

(Music)