Exhibit 15

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	) Investigation No.
CERTAIN PERSONAL DATA AND	) 337-TA-710
MOBILE COMMUNICATIONS DEVICES	)
AND RELATED SOFTWARE.	)

Hearing Room A

United States

International Trade Commission

500 E Street, Southwest

Washington, D.C.

Monday, April 25, 2011

VOLUME VI

The parties met, pursuant to the notice of the Judge, at 9:03 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

pieces of the system.
And the wrappers were one of the first pieces to come online. And so we needed the runtime loading at the same time that we were implementing the first piece.
Q. Who else was involved in coming up with the ideas for the selective loading feature of the runtime loader?
A. It was a collaboration. Certainly David Goldsmith, Chris Moeller, probably Dan Chernikoff. We were all working together. And it is difficult to go back for 20 years and say did this person say something or whatever. To the best of my recollection, we were the group of the people that were doing that.
Q. So earlier today you talked about your interpretation of certain words that appear in the claims, for example, loading.
A. Yes.
Q. Did you carefully analyze the prosecution history of the '983 patent in coming up with your interpretation of what the "loading" term means in the context of the '983 patent?
A. I'm sorry, I am not clear what you

mean by the prosecution history, if you can explain.
Q. Sure. That refers to the back and forth correspondence between your patent lawyers and the United States Patent Office.
Did you analyze that history to determine how the word "loading" should be interpreted?
A. I have seen some of that. At the time when we were working on the patent, I was involved with addressing the issues that were raised by the Patent Office. If you have something specific, I would be happy to tell you what I remember.
Q. I would like to -- I am asking you right now sitting here today when you were talking about loading and linking, did you have the prosecution history in mind when you were making that conclusion?
A. I understand your question now. "Linking" and "loading" are terms that are used interchangeably in the field of computer science. And there is a history of that where at some point in the past, probably quite a while ago now, where they were possibly

separate things.
But at the point where we were working on this patent, the words were used interchangeably.
Q. If Ms. Goldsmith were to show up at trial and testify that linking and loading are different concepts, you would have to disagree with her at the time of the invention, right?
A. You have to be careful because some of it depends on the context and who you are talking to and about what systems. If I was talking to somebody who was not well versed in the field of computer science, linking and loading is the same thing.
If I am talking to somebody who is an expert in one small piece of runtime compiler technology, they may make some small distinction between the two, but in my mind, linking and loading are the same thing.
Q. But, again, if one of your coinventors, Ms. Goldsmith, were to testify that linking and loading are different things in the context of this patent, you would have to disagree with her?
A. I would have to disagree with her.

Q. Now, if Apple's expert, Ms. Spielman, were to testify that linking and loading are different concepts, again, you would have to disagree with her?
A. Yes.
Q. Okay. I want to talk a little about this figure that you drew. This is CDX-8000 figure. Do you recall that?
A. Yes.
Q. The "TAS" in this figure stands for task address space, correct?
A. Yes.
Q. If I heard your testimony correctly earlier this morning, what you said was before you worked on the '983 patent, a bunch of people were taking the code library from the storage area and dumping it into the task address space of the application?
A. I'm afraid you misunderstood me. I meant to say that at the time, people in the computer industry, a very common technique for getting code from a data storage medium into the task address space was to just dump the whole library in. That was a common technique used at the time.

Q. That's a common technique in the prior art, right?
A. I don't know that for certain. I couldn't answer yes or no.
Q. That's what you just said, it was a common technique before you came up with the invention of the '983 patent, correct?
A. Yes.
Q. That's not your invention, your invention is not taking the entire library from storage and putting it into the task address space of an application, correct?
A. Taking the entire library and putting it into -- yeah, we were not taking it, the entire library and putting it into the task address space.
Q. That's not your invention?
A. No.
Q. Okay. Your invention, you claim now, is taking the code library from the storage area and putting it into another portion of RAM that is not the task address space of the application?
A. Yeah, preferably. That was what we wanted to have happen.

Q. But, again, if I were to take aspects of the code library that I needed and load it into, directly into the task address space before I ran the application, that would not be your invention, right?
A. I think at the time that we envisioned it, we weren't trying to limit ourselves to just loading from RAM. As I mentioned in my deposition, it would be possible to load from some other storage device, from a flash card or from something along those lines, but we would prefer to load it from the RAM.
Q. Ms. Orton, that wasn't my question. My question is if I took the library that I needed for an application, all the classes and methods that I needed for that application, it was sitting in storage on a data storage medium, before I ran the application, if I took that, loaded that entire set of things I needed into the task address space of the application, that's not your invention, right?
A. Oh, it could be. The Opus wrappers were a very low piece of the system. And it was very common to need tasking and IPC, so certainly there were some applications that

would use the whole code library because they needed the whole code library.
Q. But if I loaded that entire code library that I needed for an application before the runtime of the application, before I launched and ran the application, that would not be your invention, right?
A. So you are saying you are statically using the library not at runtime? I am not sure I understand your question.
Q. Do you have an understanding of what runtime means?
A. I do.
Q. What is your understanding of runtime?
A. Runtime is a very general term, but it is distinct from what other people would call, say, compile time, which happens at the point when you are actually compiling the code. And runtime means when things are running.
Q. Did you consider the agreed-upon definition of "during runtime" between Apple and HTC in considering your contributions to this invention? Do you know that there is such a thing?
A. Yeah, I'm sorry, I didn't really

understand your question.
Q. Do you know that there is an agreed-upon definition of "during runtime" in this case?
A. No, I am not familiar with that.
Q. Were you shown that definition?
A. I don't recall.
Q. So when you were earlier testifying about runtime loader and selective loading during runtime, you didn't have the specific definition of "during runtime" that the parties agreed to in this case in mind, did you?
A. No.
Q. Now, I want to go to another aspect of your testimony. Ms. Spielman, would you consider yourself to be familiar with how operating systems work?
A. Ms. Orton.
Q. I'm sorry, Ms. Orton, sorry. I apologize.
Do you have an opinion -- or do you have experience with operating systems?
A. I do have some experience with operating systems.
Q. And based on your work with operating

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	) Investigation No.
CERTAIN PERSONAL DATA AND	) 337-TA-710
MOBILE COMMUNICATIONS DEVICES	)
AND RELATED SOFTWARE.	)

Hearing Room A

United States

International Trade Commission

500 E Street, Southwest

Washington, D.C.

Tuesday, April 26, 2011

VOLUME VII

The parties met, pursuant to the notice of the Judge, at 9:04 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

would need to be loaded into RAM so that the program can then execute.
Q. So this hard drive here, even if you had Word on it, which is an executable program, right?
A. Yes.
Q. So say I had Word on this, I couldn't execute Word out of this hard drive, could I?
A. No. The Word program, which is an executable program, would need to be loaded into RAM in order for it to execute.
Q. That's because a hard drive is not program executable memory, is it?
A. No.
Q. And RAM, random access memory, or I think it's sometimes called, is it volatile memory, did I get that right?
A. Yes.
Q. So RAM is volatile memory, and that is a type of memory in which you can execute program code in, right?
A. Yes. You can -- it is a storage medium that can be used to execute programs.
Q. And a hard disk is not?
A. A hard disk is a storage medium that

typically does not execute programs. You need to have the program that you would want to execute in RAM in order to do that.
Q. So if you wanted to execute this Word program, you couldn't use this hard disk memory to do that, could you?
A. The hard disk memory is not memory. The hard disk is a file system. So you would need to load the file into the RAM.
Q. You don't think this is a storage device?
A. It is a storage device. It is not memory.
Q. If you put something in here, it remembers it and it comes back, right?
A. It is a type of storage device, yes, but it is not volatile memory.
Q. It is non-volatile memory, right?
A. Correct.
Q. Okay. And in order, if I want to execute a Word program, I have got to transfer that code from this form of memory into random access memory, don't I?
A. You would transfer the program itself, yes.

Q. And once it is in RAM memory, then you can execute it, right?
A. If it is an executable program, once it is in RAM, you can execute it.
Q. I am talking about Word.
A. If we're talking about the Word executable program, then Word -- win.exe would be able to execute, or word.exe.
Q. So going back to your expert report and looking together at paragraphs 27 and 28 that we looked at, what you are drawing a distinction here is between a system with static loading, where you load for a given application, you load everything before runtime, versus runtime loading, where you load only the code that will actually be executed during runtime, correct?
A. That's the distinction, yes.
Q. And the benefit of that is you save memory space, right?
A. Well, that's one of the benefits, yes.
Q. Okay. Now briefly I would like to address Apple's proposed claim construction of the selectively loading element, the last element of claims 1 and 7. Could we go to

SSS-RDX-5.
All I have done here, Ms. Spielman, is put up those two elements again. Hopefully they are accurate. The last elements of claim 1 and 7. Do you see those?
A. Yes, they look correct.
Q. Okay. And then I have simply put Apple's proposed claim construction for those elements below. Do you see that?
A. Yes.
Q. Now, claim 1 says "to selectively load required object methods," and it goes on. Do you see that?
A. Required object-oriented methods. On claim 1?
Q. Yes.
A. Yes.
Q. You see it says to selectively load?
A. Yes.
Q. And it says selectively load required object-oriented methods, right?
A. Yes.
Q. And claim 7 also says selectively loading, right?
A. Yes.

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	)	Investigation No.
CERTAIN PERSONAL DATA AND MOBILE COMMUNICATIONS DEVICES AND RELATED SOFTWARE.	)	337-TA-710

Hearing Room A

United States
International Trade Commission
500 E Street, Southwest
Washington, D.C.

Thursday, April 28, 2011

VOLUME IX

The parties met, pursuant to the notice of the Judge, at 9:00 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

that that reference that was used, that it does not teach a system-dependent form of the message. In fact, it would teach away from a system-dependent form of the message.

And the reason for that is that the McCullough reference was talking about an Ethernet package and the response stated that an Ethernet packet is not system dependent. Ethernet is a standardized communication protocol.

It is, therefore, not system-dependent. At that point, the examiner issued the office action where the rejection over McCullough has been overcome by the applicants' amendments. So the examiner understood that Ethernet was a standard non-system dependent message.

Q. Okay. And just to be clear, since I think there is a typo in the office action, the examiner says the rejection over McCullough has been overcome by applicants' amendments. Just to be clear, it was an argument, not an amendment, right?

A. That's correct.

Q. Now let's look at a couple of

miscellaneous issues related to claim construction, which I don't believe are in dispute, but I want to make sure we cover as well.

Slide 56, CDX-532, the word "said" and "a" appears in these claims like most patent claims. Can you please describe the claim construction issue here?

A. So it is my understanding that the first time a term is used in the patent, it will be referred to as "a." And then further references to that will use the "said," referring back to that first, the first time it is used.

When reading through the various claims, it is clear to someone of ordinary skill that, for example, in claim 1, the use of a second process, the second time is referring to the first time it was used.

So for all of these examples, basically, someone of ordinary skill would understand the reference was to the first time that the term was used, not an additional time.

Q. And so is that true also for an object-oriented programming language based

query and an object-oriented programming language based reply in claims 5 and 6?

A. Yes.

Q. So it is the same situation that the second instance used "a"? Yes, the second instance used "a" instead of "said"?

A. Yes.

Q. In each of those cases, did you read through the entire claim to make sure there was correspondence that the second use was in fact "said," sort of said term?

A. Yes, I did. I read through it as someone of ordinary skill who would read it to understand what would need to be implemented and that's how it would be read.

Q. Okay. Next in slide 57, CDX-533, there are two constructions, one for object-oriented programming language based message and transmitting shown.

Do you see those?

A. Yes.

Q. And, first, do these constructions have any impact, as far as you know, on any of the technical disputes or analyses in this case?

A. No, they don't.

Q. And did you consider in both cases what the meaning of the two terms should be?

A. Yes, I did.

Q. And what are they?

A. That there is no technical difference between the terms. So it would not affect my opinions from my analysis.

Q. Okay. Taking, for example, the first object-oriented programming language based message, why is it that you say this should be plain meaning?

A. Say the question one more time?

Q. Looking at the top one, object-oriented programming language based message, why is it you say that should be given its plain meaning?

A. It says everything in the term of what it is. So one of ordinary skill would understand what an object-oriented programming language based message is. There is no ambiguity there.

Q. The term transmitting as well, is there any meaningful difference between transmitting and sending?

looking at the claim itself, nothing else, that the claim itself recites sufficient structure; is that right?
A. That's my opinion, yes.
Q. Okay. And, Ryan, can we go back to claim 1, please. So using this transmitting -- the first element, I will read it into the record, "transmitting, using a first processing means, said object-oriented programming language-based message to a first proxy in said first process."
Do you see that?
A. Yes.
Q. Where is the structure?
A. It is my opinion that someone of ordinary skill would understand that transmitting using a first processing means means transmitting on a CPU, processing.
Q. So transmitting, that's a verb, right? Sorry to get into grammar again.
A. This is why I became a computer scientist. Yes, it is a verb.
Q. Transmitting is not a structure, right? Agreed?
A. Transmitting is a verb. I think

someone of ordinary skill understands what transmitting means.
Q. But I am asking you, the question is transmitting is not a structure, right? You agree with that?
A. I don't agree with that. I think that the function of transmitting is understood by someone of ordinary skill.
Q. Okay. So you just said the function of transmitting. So would you agree with me that transmitting is a function?
A. Yes.
Q. Okay. And a function is not a structure under the law, right?
A. I'm not a lawyer, so I don't think I can answer that accurately.
Q. You understand when we're talking about structure, we're talking about -- well, let me ask you. What is your understanding of what you are supposed to be looking for when you are looking for structure?
A. My understanding is that the structure would be what someone of ordinary skill would need to understand in order to perform the function.

Q. Okay. Anything else?
A. Not off the top of my head.
Q. Okay. So you agree, though, transmitting is a function, right?
A. Yes.
Q. Processing is a function, isn't it?
A. Processing could be a function, but in the context of this claim, it is my opinion that it is telling you what the processor, it is processing. It is telling you what the transmitting is doing.
Q. This doesn't say computer processor, does it?
A. That does not have those words, no.
Q. And sorry to get into the grammar again, but that's a verb, processing? It is an action word?
A. I would agree with that.
Q. It is a function, right?
A. I think if we're using the term "function" as it's applying in this context, it's my opinion that someone of ordinary skill understands that it's talking about a processor.
Q. Well, it is your opinion to this Court

that just looking at this claim element, this claim element discloses sufficient structure so that the means-plus-function analysis doesn't need to apply, correct?
A. That's correct.
Q. Okay. Where is the structure, what word is the structure?
A. It is implied by someone of ordinary skill reading this that they would understand what that is. I don't think that -- that is the structure.
Q. Well, can you tell me using any of these words in this, can you point to any word in this element that you claim is the structure?
A. That would be what's highlighted, is using a first processing means, is the structure.
Q. Okay. So "means," what structure does that connote?
A. I don't understand the question.
Q. Do you know what this word "means" means?
A. It means means. It is a first processing means. It is the way that you are

doing something. It is processing.
Q. So means itself is not a structure, right? It doesn't mean anything in the abstract, does it?
A. Someone of ordinary skill reading this would be reading it in the context of these claims. And it is my opinion that they would understand that transmitting using a first processing means would be understood to be a processor.
Q. You understand that when someone uses the word "means" -- I understand you are not a lawyer. So I am just trying to get your understanding.
A. Okay.
Q. Is it your understanding that when a person or a patent applicant claiming an invention uses the word "means" in a claim element, that usually what they are saying to the reader is you need to look to the specification to see the structure of this function? That's why there is a presumption? Did you know that?
A. I understand that if the word is used and it is not clear for what it would be of

someone of ordinary skill, that you would look to the spec for that structure. But it is my opinion that someone of ordinary skill reading this would understand what this is.
Q. And one exception to that general rule, which you cite in your report we just looked at, is if the element itself recites sufficient structure in the element itself, that can be an exception to looking at the specification? Do you understand that?
A. Yes, I do.
Q. Okay. And you think that this first processing means, that that phrase itself is what recites structure?
A. Yes.
Q. Okay. Now, I know you aren't a lawyer, Ms. Spielman. I have a couple more foundational questions to ask you about the framework under which you applied your technical expertise.
Were you aware that in cases involving computer-implemented inventions, in which the patentee uses the phrase "means," that under the law you are required -- it is required that the structure that's disclosed is more than

simply a general purpose computer or microprocessor? Are you aware of that?
MR. AROVAS: Your Honor, I am going to object to the extent I think it is a mischaracterization of the law. This is obviously not a legal expert. I mean, we can certainly cross-examine about her -- the fact that she is not applying other case law, which I think is directly contrary to what Mr. Verhoeven is saying.
If he wants to ask her have you been informed or did you apply this standard, that standard, I think that that's certainly something he can do, but, you know, legal cross-examination seems to be --
MR. VERHOEVEN: I have a way around this, Your Honor, if I may.
JUDGE CHARNESKI: Yes, go ahead.
BY MR. VERHOEVEN:
Q. Can we put up demonstrative SSS-RDX-54, please.
Now, this is a quote from a case by the Federal Circuit. And I don't want to ask you about any legal expertise. I just want to understand -- have your understanding of the

legal rules of the road you applied when you came up with your analysis. This case is the Aristocrat versus IGT case. For the record it is 521 F.3d 1328, and I have pulled out an excerpt from pin cite 1333. "In cases involving a computer-implemented invention in which the inventor has invoked means-plus-function claiming, this court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor."
MR. AROVAS: Your Honor -- sorry.
BY MR. VERHOEVEN:
Q. It goes on, "A computer-implemented means-plus-function term is limited to the corresponding structure disclosed in the specification and equivalents thereof, and the corresponding structure is the algorithm." The corresponding structure for a Section 112, 6 claim for computer-implemented function is the algorithm disclosed in the specification.
Do you see that?
A. I see what you just read.
Q. Now, when you apply --

JUDGE CHARNESKI: I think you have an objection here.
MR. AROVAS: I am going to object to the line of questioning, examining this technical expert on a series of Federal Circuit cases. As I said, I don't believe it is a correct statement of the law, and I don't believe it is appropriate for cross-examination.
JUDGE CHARNESKI: I am going to sustain the objection. Mr. Verhoeven, you can bring out any questions regarding the witness' opinion, and then in your post-hearing brief if you want to weigh it against the legal background, that's a different thing.
MR. VERHOEVEN: Thank you, Your Honor.
BY MR. VERHOEVEN:
Q. Ms. Spielman -- can we go back to claim 1, please, Ryan. This is claim 1 of the '721 patent. When you were looking at this element here to assess whether this element disclosed sufficient structure, were you looking for an algorithm?
A. I was looking to see if someone of ordinary skill understands what transmitting

means, and in my opinion they would understand what that means for using a first processing means.
Q. So you weren't looking for an algorithm?
A. I read this as someone of ordinary skill, if they would understand the claims as it is written.
Q. So my question, though, is you weren't looking for an algorithm when you were testing whether this is sufficient structure; is that correct?
A. The algorithm would be, anyone of ordinary skill would be able to implement what transmitting is. So it is my opinion that someone of ordinary skill reading this would understand what the algorithm would be necessary in order to transmit with a processor.
Q. Let me just ask you this: Looking only at this first element that has the first processing means in it --
A. Okay.
Q. -- can you tell His Honor, where is the algorithm?

A. Your Honor, someone of ordinary skill would understand that transmitting could be implemented however the developer sees fit. Someone of ordinary skill would understand what that is.
Q. Would you agree with me there is no specific algorithm in this element?
A. No, I would not agree with that. Again, I think someone of ordinary skill understands that.
Q. Okay. And where is the algorithm? Can you point me to the words?
A. The words of "transmitting using a first processing means," someone of ordinary skill understands what that algorithm would be.
Q. You are a computer scientist, you know what the word "algorithm" means, right?
A. Yes, I do.
Q. Tell His Honor what it means.
A. An algorithm is a number of steps in order to be able to accomplish something that you are doing. So it would be a series of steps.
Q. Where is the series of steps in that first element?

A. It is my opinion that someone of ordinary skill would understand the series of steps in transmitting using a first processing means, the steps are included in how to transmit something.
Q. So is it your opinion that someone could infer it, but it is not in there? Or do you think it is actually in the words here?
A. I think it is in the -- I think it is in the words by inferring from someone of ordinary skill.
Q. Which words?
A. What I just stated, "transmitting using a first processing means," somebody of ordinary skill would understand that they would be able to implement transmitting from that using a processor.
Q. Would you agree with me that this claim doesn't say "processor"?
A. I would agree with you that that word is not in there.
Q. Would you agree with me that this element does not disclose expressly any set of process steps?
A. I would not agree with that, the way

being able to use the processor, so the processor is a first processing means.
Q. So the processor is the structure, it is a piece of hardware, right? And the hardware does something. Wouldn't a person of ordinary skill reading this saying what does the hardware do, what's the function that the hardware is performing, conclude that it is performing the function of transmitting said object-oriented programming language-based message to a first proxy in said first process?
A. No, I would not agree with that.
Q. A person of ordinary skill wouldn't understand that?
A. A person of ordinary skill would understand that the CPU is a general processor. It is not the function for containing the software specific to this element. So someone of ordinary skill understands that it would -- the processor is processing.
Q. And nothing else?
A. Correct. That's what the processor does, processes.
Q. It doesn't transmit said object-oriented programming language-based

message?
A. No, that's the software that's doing that.
Q. And it is your belief that the software can't be structure?
A. In a general sense, I suppose software could be structure. But as we're talking about this element, this element doesn't require any additional structure for someone of ordinary skill to understand what this means.
Q. Let's go to another demonstrative. This is SSS-RDX-57, please.
Ms. Spielman, this is another excerpt of your report of January 26th, 2011 on the '721 patent. And I have pulled out paragraph 111.
Do you see that?
A. I do.
Q. And here you say, "Indeed, a person of skill in the art at the relevant time would not understand the processing means terms to have a function of transmitting, but would instead interpret the processing means" -- that's supposed to be "means," there, right?
A. Yes.

Q. -- "but instead would interpret the processing means to describe the location of the transmitting step of the relevant claims (and likewise for the decoding and encoding limitations."
And you probably need a close paren there too, don't you?
A. It looks like -- it looks that way.
Q. Okay. You see that sentence, right?
A. Yes.
Q. So it's your opinion that a person of ordinary skill in the art looking at the transmitting step we just looked at would not understand the processing means to have the function of transmitting, but would instead, looking at that, interpret processing means to simply be describing a location; is that right?
A. That's correct. That would be the CPU.
Q. Let's go to claim 1 again. Where is the description of a location in this step, please?
A. Someone of ordinary skill reading this would understand that the location would be the CPU as a first processing means.

Q. Okay. It doesn't say "location" anywhere in the claim, does it?
A. No.
Q. All right. Ryan, could we go to JX-1, the '721 patent, to column 73, which is claim 14.
All right. We have the patent in your binder, if you would like to pull it out, but I have just pulled out an excerpt of dependent claim 14. And I will read it into the record.
A. Do you know where it is in the binder?
MR. VERHOEVEN: If I may approach the witness.
JUDGE CHARNESKI: Yes.
BY MR. VERHOEVEN:
Q. Tell me when you are ready.
A. 74, so column 73, 14? Okay. I got it.
Q. Yeah, this is dependent claim 14 of the '721 patent. Correct?
A. It looks like a method of claim 11, so that would be a dependent claim.
Q. Is 14, right?
A. Yes, it is labeled 14.
Q. For the record, claim 14 says, "the

method of claim 11 wherein said first process and said second process are located on first and second computers respectively." Do you see that?
A. Yes, I see those words.
Q. Now, claim 1, the one we looked at, didn't say "are located on first and second computers," did it?
A. Well, this dependent claim is talking about the processes, of where the processes are located. The method step we're talking about is an element of what it's doing.
Q. Claim 1 doesn't use the words "are located on," does it?
A. I do not believe so.
Q. So you would agree with me that when the patentee wanted to say where something is located, the patentee could expressly say it, in a method claim?
A. They use the words in this claim 14.
Q. But they didn't in claim 1, a different method claim?
A. Those words are not in method claim 1.
Q. Okay. But you nevertheless believe that a person of ordinary skill reading through

this patent, noticing that some of the claims specify a location and claim 1 doesn't, would nevertheless still conclude that the transmitting step of claim 1 is specifying a location and not a function?
A. Someone of ordinary skill reading claim 1 would understand that the transmitting is using a processor to process. And in the context of claim 14, we're talking about a first and second process, not a processor. So, yes, it is my opinion someone of ordinary skill would understand that.
Q. Ms. Spielman, you have reviewed the file history for the '721 patent, correct?
A. That's correct.
Q. The second binder that I brought up for you is in its entirety the prosecution history. It is a thick document. It has been marked as JX-7.
A. Yes.
Q. I have put in some tabs I am going to reference to make it easier to find things.
A. Okay.
Q. All right? If you turn to tab 1, which hopefully is JX-7 at Control No. 8073,

and I have it on the screen as well.
May I approach, Your Honor, to make sure?
A. Okay. This tab wasn't labeled. I got it. Yeah.
Q. This is the September 10th, 1993 office action issued by the Patent Office on the '721 patent, correct?
A. Yes.
Q. And the Patent Office rejected all the claims, right?
A. Yes.
Q. Okay. You cite this office action in your report, right?
Can we put up, Ryan, Ms. Spielman's January 26th, 2011 opening report on the '721 patent at paragraph 73.
Does that refresh your recollection? Do you see the cite to the September 10th, 1993 office action?
A. Yes.
Q. Okay. So you cite this in your report, right?
A. That's correct.
Q. And now if we could go to the bottom

of Control No. 8074 and carry over to 8075 of the office action, Ryan. And that whole first paragraph there. Thank you.
So in paragraph 4 of the office action, the Patent Office examiner cites to Section 112 and the requirements of Section 112. And then says, "the specification is objected to under Section 112 as failing to adequately teach how to make and/or use the invention, i.e., failing to provide an enabling disclosure. It is unclear how a receiver object determines whether it has been given all of the information it needs to execute a message, in order to determine if a query must be generated and sent back to the sender object." Then it goes on.
Do you see that?
A. Yes.
Q. So the Patent Office is saying even when I am reading the specification of this application, it is unclear to me how the receiver object determines whether it has been given all the information it needs to execute a message. Do you see that?
A. Well, I see that in the context of

the information it needs to execute a message in order to determine if a query must be generated and sent back to the sender object.

And the applicant is -- contends that this feature is fully disclosed and is described on pages 22, 23, 24 of the originally filed spec. So they are talking about that particular piece of functionality.

Q. Correct. So let's go to 22, 23 -- let's go back then to 22. This is your tab 3.

A. I got it.

Q. So this is 22 of the original spec. Are you with me?

A. Yes.

Q. Okay. And part of what's being referenced here on page 22 is this paragraph starting on line 16, right?

A. That is the paragraph.

Q. And it says, "The local process includes a receiver proxy that accepts the message. The receiver proxy is an object that executes a forward:: method." Do you see that?

A. Yes.

Q. How would a computer scientist refer to that forward::?

A. You would refer to it as forward::. It is a method.

Q. Okay. And then it continues. "The receiver proxy encodes the message and transmits it across the process boundary to the remote object."

Do you see that?

A. Yes.

Q. Then it goes on. "In the preferred embodiment of the present invention, the proxy encodes the message" -- and I am going to skip the parenthetical -- "as an operating system message, such as a Mach message, and transmits it to the receiver object in the remote process." And it continues on.

Do you see that?

A. Yes.

Q. So the forward:: method, that's part of the way the invention processes messages, right?

A. I would -- it is my opinion that that's an example in the spec for how an object can make use of a forward method.

Q. In any event, that's what the patentee referred the patent examiner to, at least in part, to show the examiner that this patent is fully enabled and this is how it functions, right?

A. As it relates to the particular information that it needs to execute a message in order to determine if a query must be generated and sent back to the sender object.

Q. So that's yes? With that qualification?

A. This would be an example in the specification. I wouldn't say that that's the only way to do it, but they are -- the applicants are stating that it is disclosed and described on that portion of the spec.

Q. Let's go back to tab 2, please. This is Control No. 8189. So back up to --

A. Okay.

Q. Just one second while we get it up on the screen. All I need, Ryan, is to pull up that paragraph we just looked at.

So this is the paragraph we just looked at, the response of the patentee to the examiner. And we just looked at the -- we just looked at the first sentence. The second sentence in response to the examiner says,

"Applicant also refers examiner to Appendix A of the original filed specification, page 41, lines 25 through 40."

Do you see that?

A. Yes.

Q. So let's go look at that. That's at tab 4 for your ease of reference.

A. Okay. Got it.

Q. And for the record this is Control No. 8035. Do you see at the bottom it says 41?

A. Yes.

Q. And that's, I will represent, this is page 41 of the original application.

A. Okay.

Q. And this is the appendix -- remember, Appendix A was the one with all the source code in it, right?

A. Yes, this is the Objective-C source code.

Q. Now, you testified at your deposition that, at least as of the time of your deposition, in March 15th, 2011, that you didn't go through this code line by line. You hadn't done that at the time of your deposition, right?

A. That's correct. I reviewed it, but I didn't have to go through line by line.
Q. You said you had a general understanding of how it works, but you didn't go through it line by line, right?
A. Yes.
Q. Well, let's look at around lines 25 through 40, right about there, down to right about here (indicating), and pull it out.
It says "remote method," and then some characters. What is that describing?
A. This is the definition for the remote method, method info, taking a parameter.
Q. What would a person of ordinary skill understand in reading this code here? What is this showing?
A. This is showing us if the selector, if it finds the selector, return a known method interface to avoid recursion. It is checking if it found a value for that selector. It looks like it is generally getting the instance of that method. It is doing the required arguments.
If it doesn't know, on the line where it specifies Res equals paren id, that's a

generic object. If it doesn't understand it, it is forwarding the selector message to a forward, it is sending the selector message to the forward, which is of itself, and it's contained in an id object, which is a generic object in Objective-C.
Q. So this is this forward method, this thing here I am circling, that we saw in the spec, but it is actually in the code here in detail, right?
A. Yes. It is an Objective-C generic forward on the id object.
Q. And going back to tab 2, 8189, that is the specific reference, page 41, lines 25 through 40, that the applicant refers the examiner to in the appendix for the specific code to show the patent examiner that this process, this invention is enabled. Right?
A. As it relates to the need to execute a message in order to determine if a query must be generated and sent back to the sender object. That's my understanding from reading the paragraph it's referencing.
Q. Okay. Now, I may have an error. Can I check with my counsel very quickly?

If we can go back to tab 2 again.
A. Okay.
Q. This time go a few pages further to page 8188 through 8189.
A. That's exactly where we were.
Q. Let's go to 8189 then, please. We just went through the response to the enablement rejection by the examiner. Now I want to look at the response to the second paragraph in the rejection that we looked at. So that would be starting right here, Ryan (indicating). I think I am losing my battery. Anyone have a light pen I can use? There it goes.
Ryan, also the response to this paragraph.
A. There is a battery here if you need it.
MR. AROVAS: We have one here too if you need an extra.
MR. VERHOEVEN: Thank you.
BY MR. VERHOEVEN:
Q. That's fine. All right. So, again, take your time to look at this, but I think what you will see is this A here is again sort

of citing back to the examiner what the examiner said in rejecting claims 1, 2, 5, 6 and 11. So I will read just it.
It says, "As per claims 1, 2, 5, 6 and 11, it is unclear who or what is executing these steps. If they are executed by a computer, this must be explicitly stated within the context of the claims, and the claims involving providing must be clarified" -- can we highlight that, Ryan -- "must be clarified in relation to a computer actually implementing these steps."
Do you see that?
A. Yes, I do.
Q. And you remember we looked at that in the rejection in tab 1, right?
A. Yes.
Q. Okay. So this is their response to that. Do you understand that?
A. This -- yes, yes. It is.
Q. The next sentence is a one-sentence response. Do you see it?
A. Yes.
Q. "Applicant has amended these claims accordingly."

repeatedly to structure that involved the execute forward method?
A. I don't know if I would agree with the way that that question is phrased. I think they are specifically talking about the query feature here, which would be claim 5. So --
Q. Would you agree with me that they repeatedly pointed the examiner to the forward:: method, at least?
A. Yeah, they pointed the examiner to the example of the preferred embodiment that uses that feature of Objective-C as a way to do that.
Q. Okay. And they did that in response to the examiner's Section 112 issues, right?
A. That's correct.
Q. Okay. Now, let's switch subjects a little bit now and let me -- we were talking about means-plus-function in the processing means and whether those elements should be considered means-plus-function at all. Right?
A. Yes.
Q. The second issue that the parties dispute on this term is, assuming that these elements should be interpreted as

means-plus-function elements, and, therefore, you have to look to the specification for corresponding structure, there is a dispute between the parties as to what that corresponding structure should be. Are you with me?
A. Yes.
Q. So I want to ask you a few questions about that dispute, what would be the appropriate corresponding structure if HTC is right and these should be considered means-plus-function claims and you have to look to the specification.
A. Okay.
Q. Okay? Now, in your -- you have done that analysis yourself in your expert report, right?
A. Yes, I did. It is in my expert report.
Q. So let's go to that. This is demonstrative SSS-RDX-60. For the record, this is out of your -- Ms. Spielman, out of your January 26, 2011 expert report of the '721 patent, paragraph 120.
A. Yes.

Q. You agree with that, right?
A. Yes.
Q. Okay. And you say, "I am of the opinion that, if the Court were to find that the processing means claims are in means-plus-function form, which I do not believe they are, then portions of figures 3A, 3B, 3C, 4, and 5 provide corresponding structure for these limitations, together with portions of the specification at column 7, lines 30 through 59, column 10, lines 41 through 47, and column 10, line 64 through column 11, line 65."
Do you see that?
A. Yes.
Q. So you are saying in your report that's where the corresponding structure is, right?
A. That's correct. I believe there might be other places in the report that call this out, but definitely in this paragraph.
Q. Okay. So let's go to the patent and look at part of what you cite as corresponding structure. In particular, let's go to JX-1 at column 11, lines 2 through 12.

A. Is that JX-7? Oh, wait, okay. I got you.
Q. You are on JX-1, correct?
A. JX-1.
Q. Which is the '721 patent, correct?
A. That's correct. What column were you on?
Q. We just saw from your report that you cited column 10, line 64, through column 11, lines 65 as corresponding structure. I would like you to look at a part of that, column 11, lines 2 through 12. Are you with me?
A. Yes.
Q. And so this is a part of what you cited as corresponding structure. And it says, "The local process includes a sender object that sends a message to a receiver object. The object is located in the remote process. However, the sender object can send its message to the remote object as if it were a local object."
"The local process includes a receiver proxy that accepts the message. The receiver proxy is an object that executes a forward:: method. The receiver proxy encodes the message

and transmits it across the process boundary to the remote object."

Do you see that?

A. Yes.

Q. So the corresponding structure for the processing means includes, in part, the receiver proxy is an object that executes a forward:: method; you admit that, right?

A. It includes the -- it would include -- this is specific for Objective-C. So, yes, I'm including that as part of the structure.

Q. Now, let's look at that forward:: method that you included as part of the corresponding structure in a little bit more detail. I direct your attention to column 7, lines 7 through 8. There is a reference to a figure. Do you see that?

A. Yes.

Q. And it says, "Figure 5 is a flow diagram of the forwarding method of the present invention." Do you see that?

A. Yes.

Q. That's the same forwarding method I just highlighted from the spec, right?

A. That is invoking the forward:: method.

So it is the forwarding in figure 5 utilizing the Objective-C specific call for forwarding in the local process.

Q. And the patent itself says that figure 5 is a flow diagram. A flow diagram is similar to an algorithm, right?

A. Yes.

Q. Figure 5 is a flow diagram of the forwarding method, right?

A. It is included as part of the flow diagram, but this particular portion is on the local object, but it is certainly part of figure 5.

Q. Okay. And you notice that this sentence doesn't say preferred embodiment, it says "of the present invention," right?

A. That's what that says.

Q. Okay. And direct your attention to column 7, lines 7 through 8. That just says the same thing.

A. It's the same thing.

Q. We don't need to repeat that. I apologize.

A. Okay.

Q. Let's go to -- I have put together

figure 5, what that reference is, a flow diagram of the present invention, together with some text describing it into a demonstrative. Let's put up SSS-RDX-61.

Do you see on the right we have got figure 5 from JX-1, the '721 patent?

A. Yes.

Q. Then on the left we have an excerpt from the specification of JX-1, the '721 patent, in particular column 14, line 59, through column 15, line 18. Do you see that?

A. Yes.

Q. And if you look at the first sentence in the excerpt I have pulled out, it says, "The operation of the present invention is illustrated in the flow diagrams of figure 5."

Do you see that?

A. Yes.

Q. So that's referring to this figure here (indicating), right?

A. That's correct.

Q. And it is characterizing it as an illustration of the operation of the present invention, right?

A. That's what that sentence says.

Q. Okay. And you see the second paragraph, "The forward:: method then performs the first of its two tasks at step 505." Do you see that?

A. Yes.

Q. So the forward method, that's right there at 504, right, invoke forward::?

A. Yes.

Q. So that's what the text on the left is referring to, right? In other words -- let me back up. I'm sorry. Let me back up to make it a little more clear.

A. Okay.

Q. Let's walk through it. It says, "In step 501, object A sends an A selector message to object B. At decision block 502, the argument 'implementation exists in object B?' is made. If the argument is true, the method of the A selector message can be executed by object B. If that is the case, the system proceeds to step 503 and the method is executed. If the argument is not true, the system proceeds to step 504 and invokes the forward:: method."

Do you see that?

A. Yes.
Q. And we saw that in the source code we looked at that was cited by the applicant to the examiner, right?
A. Yes, that's what we looked at.
Q. You walked us right through that, right?
A. Yes.
Q. Okay. And so if it is not true, then you go to this forward:: method. Then the next paragraph says, "The forward:: method then performs the first of its two tasks at step 505. Namely, it attempts to locate an object to respond to A selector message." And it continues on.
Do you see that?
A. Yes.
Q. So this is a more detailed description of the forward:: method that we looked at from column 11 that you cited as corresponding structure, right?
A. It is the description specifically using the forward from the Objective-C language. The algorithm itself is talking about whether an object actually implements a

method. So if it doesn't implement it, there is other ways to algorithmically do that in other languages. But this is specific for Objective-C, since forward:: does not exist in other languages.
Q. But you know we're in the part, I think I tried to clarify, I am asking you to assume that means-plus-function is required, and then the question is, if we have to look for corresponding structure, what is it? Are you with me?
A. Okay.
Q. And this is the corresponding structure, isn't it?
A. This is part of what I had in mind, if that's what you are asking me.
MR. AROVAS: Just one objection, Your Honor. If Mr. Verhoeven wants to pose a hypothetical, that's fine. But we all know as lawyers that it is not just assuming means-plus-function, it is also assuming what particular function is being used. And he should specify that as well.
JUDGE CHARNESKI: I am going to overrule the objection. You can pick that up

on cross.
MR. VERHOEVEN: Thank you, Your Honor.
BY MR. VERHOEVEN:
Q. Okay. Let's switch subjects.
A. Okay.
Q. Let's turn to the issue of whether or not the processing means element, whether or not you found the processing means element in the accused technology.
I don't think I need to go on the confidential for this, but we will play it by ear depending on the answer, Your Honor.
JUDGE CHARNESKI: Sure.
BY MR. VERHOEVEN:
Q. Now, you have provided an opinion on direct examination and in your expert reports that you think the processing means element is found in the accused technology, right?
A. That's correct.
Q. But your opinion -- withdraw the question.
But you haven't done any analysis or provided any opinion, Ms. Spielman, as to whether processing means is infringed by the accused devices if we assume that the forward::

method is part of the corresponding structure, have you?
A. Can you restate that one more time so I make sure I understand.
Q. You haven't provided an opinion to this Court that there is infringement by the accused devices if we assume that HTC and the Staff are right, that the forward:: method is part of the corresponding structure? Isn't that true?
A. My opinion would be that it would still infringe under the doctrine of equivalents.
Q. My question is very specific. You have not provided an opinion in your expert reports that this processing means element is met if you assume that we're using HTC and the Staff's construction of processing means, which incorporates the corresponding structure of the forward:: method, correct?
A. I think I am answering your question accurately in that my opinion is that it would be under the doctrine of equivalents. So if not, maybe you can ask it a different way and I can make sure.

Q. Well, let's go to, back to SSS-RDX-61. Remember, we were just looking at this invoke forward?
A. Yes.
Q. And then the implementation -- the question does the -- the question in box 502, "implementation exist in object B?" Yes or no.
And then 504, if the answer is no, invoke forward. Do you see that?
A. Yes.
Q. And we talked about that here in the associated text on the left (indicating), right?
A. Yes.
Q. And we talked about that in the Appendix A source code that was cited to the examiner, right?
A. Yes.
Q. Now, you don't have a view on whether Android performs steps 502 or 504, do you?
A. From the code I have examined, it does not perform those because the interfaces are predefined.
Q. So it is your opinion that the Android system does not perform these two steps?

A. Well, the implementation exist in an object is the dynamic binding for 502. So it definitely performs those steps. If you are talking about the invoke forward::, that's not a method that is implemented in Java.
Q. So, in other words, the Android system can't even perform step 502 or 504, can it?
A. I disagree with that. I think it performs 502. If you are asking me if it invokes the C, Objective-C forward::, no, it doesn't, because it is not written in Objective-C.
Q. And just for the record, I would like to play you what you said at your deposition when you were asked this question. This is from March 15th, 2011. This is at your deposition taken after your expert reports were served, page 494, lines 12 through 18.
(Videotape played and transcribed as follows:)
"Question: Do you have a view on whether Android performs 502 and 504? If not, you can just tell me, I don't have a view on that.
"Answer: I don't think I have an

opinion one way or the other of whether the entire Android system performs those particular portions."
(End of videotape portion played.)
BY MR. VERHOEVEN:
Q. That was accurate testimony, right?
A. Yes.
Q. All right. Let's switch subjects. Another element that exists in all of the claims that are being asserted in this '721 patent is this element called proxy, right?
A. Yes.
Q. And you understand that the parties dispute the meaning of the claim term "proxy," right?
A. Yes.
Q. You understand that HTC argues, or HTC's position is that in a local process, under the correct construction of this term, there is one proxy that acts as the local receiver for all objects in the local process, right?
A. That's my understanding that that's their construction.
Q. And you understand that, in addition,

it is HTC's proposed construction that there is a one-to-one correspondence between a local proxy object and the remote object, right?
A. It is my understanding that the unicity that's being proposed by HTC is that there only exists one instance of the proxy on the local -- in the local process; that they were not arguing that there was one proxy to -- their argument is that there is one and only one proxy to a remote object.
Q. Well, let's -- I am going to put up a demonstrative and try to help understand these abstract concepts. Let's put up SSS-RDX-63.
So here I have just created an illustration.
A. Okay.
Q. Here is your line denoting the local process and the remote process. Are you with me?
A. Yes.
Q. And this is an interprocess communication I am trying to illustrate. Do you understand that?
A. Okay. On a very high level, but yes.
Q. Yes, very, very high level. And so

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	)	Investigation No.
CERTAIN PERSONAL DATA AND MOBILE COMMUNICATIONS DEVICES AND RELATED SOFTWARE.	)	337-TA-710

Hearing Room A

United States
International Trade Commission
500 E Street, Southwest
Washington, D.C.

Friday, April 29, 2011

VOLUME X

The parties met, pursuant to the notice of the Judge, at 9:00 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

the whole thing in one package?
JUDGE CHARNESKI: Right, that's right.
MR. VAN NEST: Okay. We will see what shape Mr. Bennett is in and try to get him over here sooner. I don't think there is any other option for us.
JUDGE CHARNESKI: Right. But you suggest that somehow we're hauling him off the plane. Where did he fly in from?
MR. VAN NEST: I think from Denver. He is one of the deans at the University of Colorado.
JUDGE CHARNESKI: That's not that far away. It is not a bad flight. Okay?
MR. VAN NEST: We will make that arrangement, Your Honor, thank you.
JUDGE CHARNESKI: Okay. Mr. Krupka, where do we stand? Does Apple rest in its case-in-chief?
MR. KRUPKA: Yes, but, Your Honor. The but is you may recall that we worked through with the Respondents, and I think we advised Your Honor about it at the beginning of the trial, that with certain witnesses who were going to be called by one party or the other,

that rather than have them called by both parties, that once they are called by one party, the other party can use that.
So subject to that caveat, I think the answer is yes, we rest our case.
JUDGE CHARNESKI: All right. Mr. Donovan does owe me some exhibits at some point that you marked new exhibits, I think when Dr. Mowry was on the stand. You changed them from the CDXs to Complainants exhibits.
MR. DONOVAN: I'm sorry to interrupt, Your Honor.
JUDGE CHARNESKI: Go ahead.
MR. DONOVAN: You will have them today, as well as copies for Respondents and Staff. Thank you, Your Honor.
JUDGE CHARNESKI: Okay, great.
MR. KRUPKA: Thank you, Your Honor. I think that's all we had. Thank you very much.
JUDGE CHARNESKI: Okay. Mr. Pak, good morning.
MR. PAK: Good morning. Your Honor, we're going to be calling Mr. Bornstein.
JUDGE CHARNESKI: Okay.
//

Whereupon--
DANIEL BORNSTEIN,
having been first duly sworn, was examined and testified as follows:
DIRECT EXAMINATION
BY MR. PAK:
Q. Good morning, Mr. Bornstein.
A. Good morning.
Q. Please state your name for the record.
A. Daniel Rubin Bornstein.
Q. You can always adjust your mic, so bring that closer. Who is your current employer?
A. Google.
Q. And how long have you worked at Google?
A. About five and a half years.
Q. Okay. And what are your current responsibilities at Google?
A. I am the tech lead and manager of the Dalvik team within the Android project.
Q. Did you say Dalvik?
A. I did.
Q. Can you spell that on the record?
A. Yes. D-a-l-v-i-k.

Q. And, Mr. Bornstein, when did you start working on Dalvik?
A. Soon after joining Google.
Q. And throughout your tenure at Google, has your responsibility changed?
A. Yes. So at first I was kind of a team of one working on Dalvik, and now I lead a team of several other people. We all work in the same area.
Q. So throughout the course of this investigation, we have heard a lot about this Dalvik project. So can you please tell His Honor, what is Dalvik?
A. Sure. So Dalvik is a piece of software, and in particular it is a computer program, a virtual machine is a computer program that runs other computer programs.
Q. And who came up with the name Dalvik?
A. I did.
Q. And why did you decide to name your virtual machine Dalvik?
A. Well, at the time that the project was starting, it needed a name, and I had actually just finished reading a book of translated Icelandic fiction, so I had Iceland on the

mind, and I thought maybe I can find a place name within Iceland that was pronounceable, at least by me, and was reasonably short and memorable. And I looked on a map and found Dalvik and read up a little bit about it on the Internet and sounded like a cool place, so I thought I had a name.

Q. Wonderful.

Now, can you explain to us why does Android need a virtual machine?

A. So virtual machines are useful, kind of for many different purposes, in general. For Android specifically, we wanted to provide benefits to software developers and hardware developers. And using a virtual machine actually enables that in several ways.

Q. Okay. So, Ryan, can you put up RX-1047.

Mr. Bornstein, you can see that on the screen in front of you?

A. Um-hum.

Q. Are you familiar with this document?

A. Yes. So that looks like the cover page for -- or cover slide for our presentation that I gave at Google IO in 2008.

Q. Great. And there is actually a binder of materials in front of you as well that will have copies, so you can put that in front of you now.

A. Okay.

Q. I will refer to that binder from time to time. You just said Google IO; is that correct?

A. I did.

Q. And --

A. Okay, I got it.

Q. You have got it? Great.

A. Yes. Sorry, Google IO.

Q. So, Mr. Bornstein, what is Google IO?

A. Oh, I'm sorry. Google IO is a conference that Google hosts or Google puts on every year. It is meant as developer outreach.

Q. And what do you mean by developer outreach?

A. So Google has a lot of technologies that it is interested in having other computer professionals in the world use, and so Google hosts, you know, puts on this event as a way to present a lot of these technologies and as kind of a social structure for this development

community.

Q. And, Mr. Bornstein, did you prepare these slides?

A. I did.

Q. And did you, in fact, present these slides at that conference?

A. I did.

Q. And why were you giving this presentation?

A. So, you know, in general, the conference is about developer outreach. At the time that I gave this presentation, Android had not been out for that long, and it was actually a topic of a lot of interest.

And so there were several other people who were actually giving talks on various aspects of Dalvik. And so, you know, my talk was sort of part of that, you know, we were trying to sort of kind of tell the world about, you know, what all Dalvik or what Android was about. And so I was trying to say what my part of Android was all about.

Q. Now, Mr. Bornstein, if you take a look at the binder in front of you, there is a tab titled CX-908C.

A. CX-908C.

MR. PAK: And, Your Honor, this is another version of the same presentation that was already introduced into evidence, but I would like to also introduce this version because this is in color.

JUDGE CHARNESKI: Okay.

BY MR. PAK:

Q. So, Mr. Bornstein, if you could just confirm for us that, in fact, what we're seeing on this screen, which is RX-1047, is another version of the same presentation that had been admitted into evidence as CX-908C?

A. So, yeah, they look like they are the same one, just the first one looks like a better version to me.

Q. Great. Why don't you set the binder aside now.

A. Okay.

Q. Now, Ryan, if we could jump to slide 4 in this exhibit. I think we have seen this picture a number of times. And it is titled the big picture.

Mr. Bornstein, I would like for you to identify in this diagram where the Dalvik

virtual machinery resides?
A. So if you look at the right, there is a big yellow box with a small yellow box inside. That smaller yellow box says Dalvik virtual machine. And actually, if you go to the next slide, it is circled, because this is what the talk was all about.
Q. Great. So it is the yellow box with the red line circled around it?
A. That's correct.
Q. And this is RX-1047.005. Okay.
Now, at the bottom of this diagram, you see the red box titled Linux kernel?
A. Yes.
Q. And how is that used in the Android system?
A. So the Linux kernel is, I would say it is the lower layer of the Android operating system.
Q. So, Mr. Bornstein, you talked earlier about the benefits that the virtual machine provides to both software developers and hardware manufacturers.
Can you further elaborate on the benefits that the virtual machine provides for

software developers?
A. Sure. So for software developers, what Dalvik does is reduces the amount that a developer has to care about the particulars of the hardware their software is running on.
Q. And can you please give His Honor an example, real-world example of what you mean by that?
A. Sure. So there is a metaphor I like to use for this, which is the steering wheel of a car. So as a driver, you can pretty much step into any car and you know how to operate that car using the steering wheel. You turn to the right, turn the wheel to the right and the car is going to go to the right. Turn the wheel to the left, the car is going to go to the left. And you, as a driver, you don't have to care about kind of what is going on under the hood. And, in fact, there is actually a lot of different ways that cars can be put together and you don't have to care.
So, similarly, for software developers, Dalvik is kind of like that steering wheel. So a software developer that knows how to use what Dalvik is providing, they

can use that, and then Dalvik can operate the different cars and, in fact, the different pieces of hardware. And the software developer doesn't have to really care about what's going on sort of under the hood of Dalvik.
Q. Let's focus on the hardware manufacturers. What are some of the benefits that the Dalvik virtual machine provides to hardware manufacturers?
A. So for hardware manufacturers, it is sort of the other side of the coin. So hardware manufacturers are basically freed up to have a greater diversity of hardware because they know that there is this virtual machine there that's shielding software developers from the choices, the details of the choices that the hardware manufacturers are making.
Q. Now, Mr. Bornstein, before your work on this particular virtual machine called Dalvik, did you work on other virtual machines?
A. I did.
Q. And can you tell us some examples of other virtual machines that you worked on?
A. Sure. So I will give you two examples. At my immediately previous job, I

worked at a company called Danger. Danger was also in the Smartphone space and had a virtual machine, and I was a developer on that. And actually, very early in my career, I worked on a virtual machine for a language called ScriptX.
Q. Now, earlier in our investigation, we heard testimony from Mr. Andy Rubin, who I believe also worked at Danger. Did you work with Mr. Rubin?
A. I did.
Q. Now, when was the first time you came across the concept of virtual machines?
A. Probably that would have been in like the early '80s.
Q. And what system was that?
A. So on my Apple II, there was a Pascal development system. Pascal is a computer programming language. And this Pascal system compiled into a virtual -- into a virtual machine referred to as a p-code system.
Q. Mr. Bornstein, if I look at the top of this big picture diagram, I see applications. Do you see that?
A. I do.

Q. So do these applications in Android make use of the Dalvik virtual machine in yellow?
A. They do.
Q. And as far as you know, are you aware of any Android applications that do not use the Dalvik virtual machine?
A. No, I am not.
Q. I want to switch topics and talk about programming languages. Does Android have a default programming language for programmers to write applications?
A. Yes, it does. It is the Java programming language.
Q. And why was the Java programming language selected as the default programming language for Android?
A. So at the time Android was starting out, we knew we needed to have a language that we wanted to support, and we wanted to find a language that had, I would say, sort of like the right shape of functionality for the kinds of developers and the kinds of development that we wanted.
And we wanted to have language that

already had good support in the open source community. And the Java programming language fit the bill in both regards.
Q. Based on your work with the Java programming language, how would you characterize that language?
A. So I would say Java is an object-oriented class-based programming language.
Q. And what do you mean when you say object-oriented programming language?
A. By object-oriented, I mean that the language embodies this concept of objects, where objects are combinations of behavior and information.
Q. And when you say class-based programming language, what do you mean by class?
A. So a class-based programming language is an object-oriented programming language that also has this concept of classes, where classes are ways to group related sets of objects.
Q. And can you give all of us an example of a class?
A. Oh, sure. So if you can think of, you

know, anything, any application you might be running like on your computer, you might have buttons that you click on, so there might be a class called button that would be used to help represent those buttons on the screen.
Q. And how do classes relate to each other in systems that you have worked on?
A. So one of the primary ways that classes relate to each other is in this relationship called a superclass, subclass relationship. And that's where you have one class called a parent class, and it has subclasses where the subclasses, they inherit behavior and state from the parent class and then specialize that behavior in some way.
Q. So using this button example on the screen, can you give us an example of a subclass or subclasses of a button class?
A. Sure. So two subclasses, two possible subclasses of that button class I was talking about might be a text button and an icon button.
Q. And in your example, what additional functionality might those subclasses of the button class provide?

A. So I said that the icon button class might be a button where it has like a little picture in it, so like on the phone if you make a phone call, there might be a little picture of a telephone in a button. And so you can think of an icon button as being a button that draws a little picture.
Similarly, a text button might be a button that draws a little piece of text inside, so like in your e-mail client, there might be a little button that says reply, for example.
Q. And in a class-based programming language, how do classes implement the types of functionality that you just mentioned?
A. So in a class-based programming language, each class will typically define a set of things called methods. And a method is simply a named piece of code or a named piece of behavior.
Q. And can you give us an example of a method?
A. So, sure. So this button class we have been talking about might define a draw method.

Q. And how would a draw method work in that example?
A. So let's say that in this case, the draw method might simply draw kind of the outline of a button or the frame of a button. Think of it as like making an empty button.
Q. And then for the subclasses that you mentioned, the icon class and the text class, how would the same draw method work for those subclasses?
A. So in the -- as I said, a subclass inherits some behavior from its parent class. So the icon button class, say, would inherit the behavior of drawing that frame, but then in addition, would draw the icon inside.
And the text button class would inherit that same frame drawing behavior but then, in addition, would draw that piece of text inside.
Q. And based on your experience, Mr. Bornstein, can some classes define methods that are not implemented by other classes?
A. Absolutely. That's quite common.
Q. And can you give us an example of that?

A. Sure. So let's say -- so for a text button, because it is drawing a piece of text, you might want to control like the style of the text, so like to make it italic or bold or something like that. So the text button class might -- that class specifically might define a set style method.
Q. How would you contrast that, for example, with the other type of button subclass?
A. So this set style method would only be defined on text button, because the base button class doesn't have any concept of text, nor does the icon button subclass have any concept of text. There would be no point in defining a set style method on either of those. And so you simply wouldn't have done that.
Q. Also based on your experience, Mr. Bornstein, can a class contain methods that are not actually executed in a given application?
A. Oh, yeah, absolutely.
Q. We talked about classes and subclasses. Are you aware of something called objects?

A. Yes.
Q. And how do objects relate to classes?
A. So in a class-based system, so you have this set of objects and each object is associated with one particular class.
Q. And can you give us an example of an object?
A. Sure. So in this button example I have been talking about, so you can imagine that in some application that you are running, it might have a window and inside that window, you would have, say, three different buttons, let's say, two of the buttons had icons inside of them and one of them had a piece of text inside.
So in this application, you might have each of those buttons represented by a different object. So there would be three objects, two of which would be, say, icon buttons and one of which would be a text button.
Q. And would all three of those objects belong to the parent class button?
A. Yes.
Q. Now, in a class-based system, is there

something called a library?
A. Yes.
Q. And what is a library?
A. So in a class-based system, a library is simply a set of classes.
Q. And, again, can you give us an example of a library in a class-based system?
A. Sure. So in this example we have been talking about, so button, icon button, text button, these are all what are referred to as user interface elements or user interface classes. You can imagine that there are other kinds of user interface classes to represent things like scroll bars or text entry fields and the like.
All of those classes might be put together into a user interface library.
Q. And why would somebody do that? Why would somebody take all of these different classes and put them together in one library file?
A. So one way to think about it is as a convenience to programmers. And so if you as a programmer are going to use one class out of a library in a particular area of functionality,

then there is a good chance you are going to be using other classes in that same area of functionality.

And furthermore, those classes in that area of functionality will end up using each other.

Q. We talked a little bit about Java. Are you familiar with traditional Java-based systems?

A. I am.

Q. And how are you familiar with Java-based systems?

A. So I have been using the Java programming language for many years now, and I have helped implement Java systems in various capacities over the years.

Q. And how do traditional Java-based systems implement libraries, classes, and methods that we have been discussing this morning?

A. So in terms of, say, how a library gets represented in storage, in a traditional Java system, a library is stored in a kind of file called a JAR file.

Q. Is that JAR?

A. That's JAR, yes.

Q. Just like a jar?

A. Like a jar. It is actually kind of a pun.

Q. And what does JAR stand for?

A. So JAR stands for Java archive.

Q. And what is an archive file in Java?

A. So an archive file in general is a file which contains a hierarchy of directories and files. Directories are like folders like you might see on your computer.

Q. And what types of files are typically contained in a JAR archive file in Java?

A. So in a traditional Java JAR file, the files that are contained -- the files contained in it are the individual files representing each of these classes that we have been talking about.

Q. And would it be accurate to call them class files?

A. Yes. In fact, they are commonly called class files.

Q. Is it possible in a standard Java implementation to select and load into memory a particular class file?

A. Yes, it is actually designed for that.

Q. So it is possible to select just one class file out of a JAR file in Java and load just that component into memory?

A. Yes.

MR. PAK: Your Honor, may the witness approach the board? He has a few drawings.

JUDGE CHARNESKI: Yes.

MR. PAK: Thank you.

BY MR. PAK:

Q. So, Mr. Bornstein, using the drawing board here, can you please illustrate for us how a JAR file in Java would hold these different class files?

A. Sure thing. So I was just talking about a user interface library.

JUDGE CHARNESKI: If you can speak loud enough for the court reporter.

THE WITNESS: I was just talking about a user interface library. And so a common abbreviation for that is UI. So you might have this file called UI library.JAR.

And as I said, it contains -- it would contain within it a hierarchy of directories, that would then contain the individual class

files, so there might be a directory called widget. So user interface elements are often referred to as widgets. So there might be a widget directory.

Under the widget directory, you might have a directory called button. And this one would be the directory that contains all of these button classes we have been talking about. So I will go ahead and draw those.

BY MR. PAK:

Q. Sure.

A. So there would be this base button class and, in addition, there would be the icon button class and the text button class.

So in addition, you can imagine there are other sub-directories of widget and there is other sub-directories in general. In general, there is going to be other class files. And actually, in addition, there are things called resources. And these are things like images or sound, large pieces of text, that sort of thing.

But in terms of the code, this is kind of the interesting part here where there is one file for each of the classes.

Q. So, Mr. Bornstein, earlier today when you mentioned that you could select a single class file and load it separately into memory, is that what you were discussing?
A. Yes, exactly. So each of these -- each of these classes here is in the structure of the JAR archive file, it is set up so that it can be pulled out individually.
Q. Now, Mr. Bornstein, you talked earlier about the fact that Android supports the Java programming language. In terms of how Android implements library files, how would you compare that, the Android approach versus what you have just drawn here on the Java approach?
A. So the Android approach is actually a bit different. So in Android, instead of having a separate file per class in this directory hierarchy, in Dalvik we use a single file, so it is still -- sorry, the outer JAR file, we actually still use a file in the JAR format, but inside that JAR file we have just a single file called classes.dex that represents all of the classes in the library.
Q. And would you mind drawing a line and then underneath that line illustrating for us

this Dalvik.dex format?
A. Sure. So this is just traditional Java. And so this will be for Android. So, again, you might still have the same name for the JAR file, but as far as you crack it open, you will see that instead of all this widget, button, et cetera, .class files, there is just one file called classes.dex. And you will also still have the resource files, just for completeness.
And so in comparing these guys to these guys, it is really in this case one file for all the classes.
Q. So if I understand you correctly, Mr. Bornstein, is it true that all of the methods, classes that I would need for a particular application would be stored in that single file?
A. Yes.
Q. And is it possible to go into that single class file in DEX and select just a portion of that file and load it into memory?
A. No.
Q. Now, we have been talking about DEX. Is that D-E-X?

A. Yes, that's right, it is right here.
Q. And what does DEX stand for?
A. It is stands for Dalvik executable.
Q. Can you write that on the bottom?
A. Oh, sure.
Q. Mr. Bornstein, if you could return to your witness chair, I have a few more questions for you, but I think we have the drawing.
MR. PAK: Your Honor, I would like to mark this as a demonstrative.
JUDGE CHARNESKI: Sure.
MR. PAK: Just for the record, I have marked the drawing by Mr. Bornstein as DB-RDX-1.
(Respondent Exhibit Number DB-RDX-1 was marked for identification.)
BY MR. PAK:
Q. Mr. Bornstein, with this drawing in front of you, I would like to ask you a few more questions about the DEX file structure in Android. First of all, let me ask, is a DEX file an executable file?
A. Yes.
Q. What makes a DEX file executable?
A. I would say that a DEX file is

executable, specifically because it contains code inside it.
Q. And what is code again?
A. So code is the actual instructions of a computer program.
Q. And in the Android architecture, what is the thing that executes the code in the DEX file?
A. The code in the DEX file is executed by the Dalvik virtual machine.
Q. Now, if someone were to get up here and say a DEX file is only a data file and it is not an executable file, what would you say to that?
A. I would say they are wrong.
Q. And who designed the DEX file format?
A. I did.
Q. Now, going back to these Java systems that we looked at, how do standard Java systems load class files into executable memory?
A. So in a traditional Java system, classes get loaded on demand one by one.
Q. By comparison, how does Dalvik, your system in Android, load the DEX file into executable memory?

A. So in contrast, a DEX file is always loaded as a single unit.
Q. Now, was there a point in your development process when you considered using the standard Java approach that's depicted above in that figure?
A. Yes, we did.
Q. And what was your ultimate decision based on that?
A. Ultimately, we rejected it in favor of doing Dalvik and DEX files as described.
Q. Now, if this approach at the top was popular and well-known in Java, why did you decide to take a different approach?
A. So it boils down to an efficiency argument and the effect on the user experience that we wanted.
Q. So when you say efficiency, can you further elaborate for His Honor what you mean by that?
A. Sure. So any time that you load a class, that's what I would call a heavyweight operation. And if you are in the middle of running a program, what our aim is as designers of the system was to minimize any glitches or

pauses that you would experience.
So loading is, again, this heavyweight thing that we would want to avoid having to do when you're in the middle of running a program. So what we do in Dalvik and with DEX files is we do all of that loading upfront. So that happens either as your phone is booting, your device is booting or just before you start running an application.
Q. And you talked about this heavyweight operation that's associated with the traditional Java loading approach.
What would be the impact on the user experience if you were to adopt the standard Java approach?
A. So it would be that you would experience these occasional pauses or glitches.
Q. And before testifying in Court today, have you explained this difference between the Java system and the Dalvik system to other people?
A. Yeah, many times.
Q. So I am going to switch gears and talk about a different topic. In Android, are you familiar with the term system libraries?

A. I am.
Q. And what are system libraries?
A. System libraries are simply libraries that are provided by the operating system.
Q. And in Android, can one application provide libraries to another application?
A. No, it is not designed to do that.
Q. And why not?
A. So there is actually a lot of trouble being able to get that right. So there is actually a common term called DLL hell, which some people have found themselves in. And basically it is a set of problems that we didn't want to foist upon the users of Android.
Q. Now, in terms of system libraries, are there things called core libraries?
A. There are.
Q. And what are core libraries?
A. So core libraries are simply those libraries that are provided to all applications without them explicitly having to ask for them.
Q. And does Android have different types of core libraries?
A. Yes.
Q. And what are those different types?

A. So there are core Dalvik libraries and core native libraries.
Q. And when are these core libraries loaded into executable memory?
A. They are loaded at boot time.
Q. And when you say boot time, what do you mean by boot time?
A. That's before any applications have started.
Q. And is that true for both Dalvik libraries and these other native libraries?
A. Yes.
Q. And how does the Android system determine which of these core libraries to load into executable memory during the boot process?
A. It loads all of them.
Q. And why did you decide as the designer of Dalvik to load all these core libraries into executable memory during boot?
A. So it is that same efficiency argument that I was making earlier.
Q. And what is that again, remind us?
A. So it is that we didn't want to cause undue pauses when a user is actually using their device actively.

Q. So earlier in this investigation, we heard some testimony about a process called Zygote, Z-y-g-o-t-e. Mr. Bornstein, are you familiar with the Zygote process in Android?
A. I am.
Q. And what is the Zygote process in Android?
A. So the Zygote process is a piece of the Android operating system. And it is responsible for launching applications.
Q. And when is the Zygote created?
A. It is created at boot time.
Q. And why is this process called a Zygote?
A. So it is an analogy to biology. So in biology after an egg is fertilized, it is called a Zygote. And the Zygote is full of these undifferentiated cells. And, you know, over the course of an organism's growth, these undifferentiated cells eventually become differentiated into the different parts of an organism.
Similarly, the Zygote is in a sense an undifferentiated process and it eventually spawns off these other processes that become

the applications and other pieces of the system.
Q. How does the Zygote know when to launch an Android application?
A. So the Zygote launches an application when it receives a command from a different part of the operating system.
Q. After it receives this command, what is the first thing that a Zygote process would do?
A. So when the Zygote receives the commands to launch an application, the first thing it does in response is this thing called forking.
Q. And when you say fork, is it f-o-r-k?
A. Yes, f-o-r-k. It is like a fork in the road.
Q. Can you further explain for us what you mean by forking in this context?
A. Sure. So imagine the process is walking along a road and it hits a fork in the road. So what it does is it clones itself. So in a sense the process gets to walk in both directions, except at this point it is two processes, one called the parent process and

the other is called the child process.
Q. And after it does this forking in the road, what does the Zygote do next?
A. So at this point, there is two Zygotes, so there is the original Zygote, which is now called the parent Zygote, and the parent Zygote goes back to listening for commands. And then the child Zygote is the one that actually goes ahead and launches the application.
Q. How does the child Zygote launch the application?
A. So to actually launch the application, what the -- one of the first things that the child Zygote does is to load the DEX file for the application that's being launched.
Q. And is this the same DEX file that we have been discussing?
A. Yes.
Q. And just to clarify, when does this loading of the DEX file into executable memory take place?
A. It takes place just before the application starts running.
Q. So it is before execution?

A. Yes, before execution.
Q. Are any lines of code actually executed in an application before its DEX file is loaded?
A. No.
Q. And when the DEX file is loaded into executable memory, what type of memory is that generally?
A. So that's the memory of that, of that process.
Q. Are you familiar with the term address space?
A. Yes.
Q. What is your understanding of address space?
A. So address space refers to the memory, you know, in a particular context, so you might say the process address space is the memory available to a particular process.
Q. Are you familiar with the term task address space?
A. Yes.
Q. And how does the term task address space relate to what you just talked about in terms of process address space?

A. So task and process are basically synonyms. I would say that task is kind of an older term. Process is the term that I would use.
Q. When you say that the memory belongs to the task or process space, what do you mean by that?
A. Can you repeat the question?
Q. When you say -- you were talking about this memory belongs to the address space of the child process, what do you mean by that?
A. So what I mean is that the process in a sense is confined to activity only within the memory that it can refer to. So this process's address space is just all the memory that it can refer to.
Q. Now, why is this memory space or region called an address space?
A. You can think of the term address as like building addresses. So what it is, it is the set of addresses that a process is allowed to refer to.
Q. When the Zygote is loading a new application, is launching a new application, into which task address space does the Zygote

load the DEX file?
A. So when launching an application, so remember this is the child Zygote process that's doing this, it loads the application DEX file into its process address space and then it starts running the application. The moment it starts running the application, it is now the application's process and the application's address space.
Q. I would like to switch back to RX-1047, Ryan.
MR. PAK: Your Honor, I have a few more questions on this presentation, but at this time, I would like to move this exhibit into evidence.
JUDGE CHARNESKI: All right. Any objection?
MR. FRIEDMAN: No objection.
MS. JOFFRE: No objection.
JUDGE CHARNESKI: Admitted.
(Respondent Exhibit Number RX-1047 was received into evidence.)
BY MR. PAK:
Q. Ryan, if we can go to slide 9 in this presentation. Mr. Bornstein, if you can take a

look at this presentation slide and can you identify anything in this slide that discusses this concept of address space for processes?
A. Yeah. The last two bullet items do.
Q. Can you read those last two bullet items into the record?
A. Yes. "Multiple independent mutually-suspicious processes, separate address spaces, separate memory."
Q. And that's an interesting term, mutually suspicious. What do you mean by "mutually suspicious" there?
A. So there is a reference to computer security. And in a well-designed, secure system, one of the aspects is that different applications not be able to mess with each other. And so what sometimes you say is that applications are mutually suspicious of each other. So, for example, if you are playing solitaire and you have your e-mail program running, you don't want the solitaire application to be able to like grab your e-mail to surreptitiously send it off to ne'er-do-wells.
And similarly, although maybe less

dangerously, you don't want the e-mail program to be able to grab your solitaire high scores and send it away.
Q. How does this idea of having separate address spaces address the security issue?
A. So separated address spaces, this is basically the enforcement mechanism that disallows that kind of bad behavior.
Q. Can you explain exactly how that happens?
A. So each process only can use the memory in its own address space. And so it can't -- it simply can't reach into another address space to grab any data out.
Q. If we can turn to slide 26, Ryan, in this PowerPoint. We have a slide called enter the Zygote. Is the concept of task address space or process address space depicted in this particular diagram?
A. Yes, it is.
Q. And can you explain to His Honor how they are depicted?
A. Sure. So you can see in this diagram there are four columns, where each of the columns is kind of a grayish box that contains

a few other white boxes inside. Each of those columns is meant to be a separate process and the separate process address space.
Q. So why don't we focus on one of these applications. I want to focus now on the browser application. Do you see that?
A. Yes.
Q. So which portion of the task address space available to the browser application is executable program memory?
A. So among the things listed in the diagram, the things that are executed -- really it is the whole memory.
Q. And when you say whole memory, in this picture, how is that depicted?
A. So for the browser, the address space is, if you look at the browser box, it is that third column. And actually it is the contents of the first column as well. And that's what that green arrow is meant to represent.
Q. Now, I see a reference in this diagram to a browser DEX file. Do you see that?
A. Yes.
Q. And is that the same DEX file that we were discussing earlier?

A. Yes.
Q. And this time is it associated with the browser application?
A. Yes.
Q. In terms of memory space, where is the browser DEX file located at this point for the browser?
A. The browser DEX file is located in the address space of the browser process.
Q. And when was the browser DEX file loaded into the task address space of the browser application?
A. It would have been loaded in its entirety just before the browser started running.
Q. And is that the same process of loading it once that we talked about earlier?
A. Yes.
Q. At that point after it is loaded into memory, is the code inside the DEX file executable?
A. I'm sorry, say again?
Q. At that point, is the code inside the DEX file executable?
A. Oh, yeah, absolutely.

Q. Once the DEX file is loaded into the task address space of the browser application, does it ever move to another part of memory?
A. No.
Q. So it stays in that address space?
A. Yes.
Q. I would like to direct your attention to another portion of your 2008 presentation. If we can go to slide 23, Ryan. In this presentation at the top, do you see it says, four kinds of memory?
A. Yes.
Q. And there is some discussion here between clean versus dirty. Do you see that?
A. I do.
Q. Mr. Bornstein, in the context of this presentation, what did you mean by the term clean?
A. So in the context of this presentation, clean memory is simply memory that hasn't been modified.
Q. If we see the word dirty here, what does that mean in the context of the presentation?
A. In the same context, dirty is really

just the opposite. It is memory that has been modified.
Q. Now, does the distinction between clean and dirty memory in your presentation have anything to do with whether the code is executable or not?
A. No.
Q. Does the Dalvik code itself distinguish between dirty and clean memory?
A. No.
Q. Is there any constraint within the Dalvik code on the ability to execute such code based on whether it is in clean or dirty memory?
A. No.
Q. So then, Mr. Bornstein, why were you talking about clean and dirty memory in this context?
A. So the point of this presentation was to help motivate the design of Dalvik from calling it a system-wide perspective, so clean versus dirty is a distinction that I as a programmer understand. But in the end, it is not a distinction that Dalvik needed to make itself.

Q. Okay. I would like to now switch gears a little bit and talk about something called Bionic, B-i-o-n-i-c.
And what is Bionic in Android?
A. Bionic is the name of one of the core native libraries.
Q. And is this part of the library structure that we talked about earlier?
A. Yes.
Q. Okay. Does the Zygote process that we have been discussing use Bionic?
A. Yes.
Q. Does the Zygote process actually load Bionic into executable memory?
A. Actually, no, it doesn't.
Q. When is the Bionic loaded into memory?
A. So for the Zygote, Bionic gets loaded by the same process that starts Zygote itself.
Q. And are you familiar with something called local server socket?
A. I am.
Q. And what is that?
A. Local server socket is one of the core Dalvik classes.
Q. And does the Zygote process use local

server socket?
A. It does.
Q. And how does it use that?
A. So I mentioned that what the Zygote process does is listen for commands from a different part of the system. Local server socket is a class that helps implement that communication mechanism.
Q. And where is the code for local server socket located when it is being stored?
A. In storage, it is part of the -- it is in one of the core Dalvik libraries. That library is called the framework library. So it is stored as part of the framework library.
Q. And when is the local server socket code actually brought in or loaded into executable memory?
A. At boot time.
Q. Is that the same boot time that we talked about before?
A. Yes.
Q. And how is that loading done?
A. So it is done just like all DEX files. The framework, the DEX file associated with the framework is loaded all at once.

Q. So is the local server socket code loaded in executable memory before or after the Zygote starts executing?
A. Before.
Q. Now I am going to go a little bit further down into the inner workings of the Dalvik machine.
A. Okay.
Q. Are you familiar with something called op new instance, OP_new_instance?
A. I am.
Q. That's a mouthful. But what is op_new_instance?
A. So op_new_instance is a label that's used in the source code for the Dalvik virtual machine to refer to an op code. That's what the op stands for. So it is an op code called new instance. And that's new-instance.
Q. And what is an op code?
A. An op code, you can think of it as kind of a very low level command.
Q. And what is the purpose of this low-level command called new instance?
A. So the op code new instance indicates that a new instance of an object should be

created.
Q. And what actually creates the new instance of classes in Android?
A. That would be the Dalvik virtual machine.
Q. Now, in creating a new instance of a class, does Dalvik load any methods into executable memory?
A. No.
Q. And, again, in creating a new instance of a class, does Dalvik load any classes into executable memory?
A. No.
Q. And how do you know that?
A. So, I mean, there is no mechanism for it. The way Dalvik is designed, the way Dalvik is designed, all of the loading for a given library, for a given application, is always done before any execution of that code. So there is simply nothing in the Dalvik implementation to do that.
Q. And who designed the Dalvik implementation?
A. I did.
Q. And why did you make that particular

design choice?
A. So this really goes back to that same efficiency argument I was making earlier. We really wanted to do as much as possible to avoid these kind of glitches or pauses during when a user is actually using their phone or their device.
Q. So let's focus on the scenario where a class, a particular class is used for the first time in Android application. What happens when an Android application uses a class for the first time?
A. So when an Android application uses a class for the first time, that class undergoes this thing called class resolution.
Q. And what do you mean by class resolution?
A. Class resolution is the act of taking the name of a class and using that name to find a more direct kind of computer-oriented reference to that class.
Q. Are you familiar with something called dynamic linking?
A. I am.
Q. And how is class resolution related to

dynamic linking?
A. Class resolution, I would say, is a particular kind of dynamic linking.
Q. When performing class resolution, where does Dalvik look to find a class to see if it has been resolved or not?
A. So it just looks in the memory of the process that's doing that resolution.
Q. When you say memory, is that the task address space?
A. Yes.
Q. Does Dalvik look anywhere else when performing class resolution?
A. No.
Q. If a class is actually sitting in storage in an Android application somewhere, but it is not in the task address space of a process, can Dalvik go find that missing class and load it into memory?
A. No, not during class resolution.
Q. And why not?
A. Again, it is just -- there is no mechanism for that.
Q. What if a class is missing in my task address space as the application, but is

sitting in some other part of memory, can the Dalvik code go find that in another portion of memory and bring it or load it into executable program memory?
A. No.
Q. And why not?
A. So, again, there is no mechanism for that.
Q. Okay. So what would happen if Dalvik is running along with a particular application and it tries to resolve a class that is not in the current task address space of an application, what would happen?
A. So if there is -- so if there is the name of a class and that class is not in the memory of the application and you try to resolve that name, it would simply be an error.
Q. Is there any mechanism in Dalvik that would automatically load a missing piece of code into the task address space of an application?
A. No.
Q. Are you familiar with the function dvmResolveClass?
A. Yes.

Q. What is the purpose of that class?
A. DvmResolveClass is the function within the implementation of the Dalvik virtual machine that performs class resolution.
Q. Does dvmResolveClass load any classes into executable memory?
A. No, it doesn't.
Q. How about dvmResolveMethod, are you familiar with that function?
A. I am.
Q. And what is that?
A. So similar to dvmResolveClass, dvmResolveMethod is the function inside the Dalvik virtual machine implementation that performs method resolution.
Q. And does dvmResolveMethod itself load any classes into executable memory?
A. No.
Q. Are you familiar with something called resolver cache, c-a-c-h-e?
A. Yes.
Q. And what is that?
A. The resolver cache was this piece of implementation inside the Dalvik VM that was meant to make various forms of resolution more

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	)	Investigation No.
CERTAIN PERSONAL DATA AND MOBILE COMMUNICATIONS DEVICES AND RELATED SOFTWARE.	)	337-TA-710

Hearing Room A

United States
International Trade Commission
500 E Street, Southwest
Washington, D.C.

Wednesday, May 4, 2011

VOLUME XIII

The parties met, pursuant to the notice of the Judge, at 9:01 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

MR. PAK: I will represent on the record that we have no intention of relying on CORBA as prior art. We're explaining CORBA as a predecessor technology to the Android development process that we have currently going on at Google. And, again, if that's the objection, then I will represent on the record that we will not make any prior art arguments based on CORBA.

MR. PIEJA: Your Honor, if counsel is willing to make that representation, we will accept it and I will withdraw the objection subject to that representation.

JUDGE CHARNESKI: Well, it's made. Very good.

BY MR. PAK:

Q. Let's go back to CORBA.

A. Right.

Q. We were talking about the tools that CORBA used to predefine these messages. Was there such a tool in CORBA?

A. Yes. So there is a language you write your interfaces in called interface description language, or IDL for short. And then there is a corresponding tool you use to compile these

descriptions into code that you get built into your application to do the communication.

Q. Now, Ms. Hackborn, is there an analogue to this IDL tool in Android?

A. Yes, there is.

Q. And what is that called?

A. So in Android this is called the Android interface description language, or AIDL.

Q. And how is AIDL used in Android?

A. On Android you use AIDL to -- in a similar way, you predefine all the methods that your application is going to handle, so that when you are compiling your applications, AIDL turns those descriptions into Java language source code that implements the communication and then gets compiled into the application until it can finally run on the device.

Q. Is what you described the same for both AIDL on Android and IDL on CORBA?

A. Yes, it is.

Q. After you worked with CORBA at school, did you have additional experience working with interprocess communications?

A. Yes, I did.

Q. And what was your next experience?

A. My next experience was when I was working at a company called Be, that's B-e.

Q. Great. And what kind of company was Be?

A. Be was a software company that made a desktop operating system that they sold to users.

Q. And when did you start at Be?

A. I started there in 1999.

Q. What was your title?

A. I was a software engineer there.

Q. And did your work at Be involve working on these interprocess communication mechanisms that we talked about?

A. Yes, it did.

Q. And was there a name that was given to that mechanism?

A. Yes. That was called Binder at Be.

Q. And how is that Binder at Be OS related to the Binder we have been discussing for Android?

A. The Android's Binder is a direct descendant of the one from Be.

Q. Now, where was Binder first developed?

A. Binder was first developed at Be.

Q. And were you there at the time?

A. Yes, I was.

Q. So at the time that Binder was created at Be, what was the purpose of Binder?

A. At Be -- Be was a multi-process system. We had things divided into processes. At the time we were starting to do more work across processes, and we needed tools to help us with that. So Binder was created to help.

Q. And was there a particular system that was in use at Be at the time?

A. Yes, this was on -- BeOS was the operating system.

Q. So it's B-e-O-S?

A. B-e-O-S, yes.

Q. And how long did you work at Be?

A. I was there for two years.

Q. And what happened then?

A. At that point Be was acquired by a company called Palm.

Q. And what kind of company was Palm?

A. So Palm was a company that made PDAs, mobile devices. They had their own mobile operating system at the time that was becoming

A. Yes, I have.
Q. And what is this document?
A. This is the front page on my web site where I am hosting the OpenBinder documentation describing the, why that's there and what the documentation is.
Q. Now, that C designation indicates this is a confidential document, but is it, in fact, confidential?
A. No, not at all.
Q. Now, does this document accurately describe the OpenBinder project?
A. Yes, it does.
MR. PAK: Your Honor, I would like to move RX-1715C into evidence with the note that this is not an actual confidential document.
JUDGE CHARNESKI: Okay. Counsel?
MR. PIEJA: No objection, Your Honor.
MS. JOFFRE: No objection.
JUDGE CHARNESKI: Admitted.
(Respondent Exhibit Number RX-1715C was received into evidence.)
BY MR. PAK:
Q. Now, stepping back, Ms. Hackborn, how long has Binder been in use on Android?

A. It has been in use on Android about since I began there five years ago.
Q. And in your assessment has Binder worked well in Android?
A. Yes, it has.
Q. And has Binder worked well in the other systems that we talked about?
A. Yes, it has.
Q. And just to remind us again, what were the other systems that used Binder as the interprocess communication mechanism?
A. They were BeOS, Palm OS Cobalt and Palm's Android-based operating system.
Q. So Ms. Hackborn, in total between Google, PalmSource, and Be, how long have you personally been working on this Binder framework?
A. For about 11 years.
Q. Now, Ms. Hackborn, returning to Binder as implemented on Android, let's walk through a typical scenario in which an application developer would use Binder.
A. Okay.
Q. So if an application developer wants to use the Binder interface, what is the first

thing that the developer needs to think about?
A. So the first thing that a developer needs to do is decide what methods they want to make available to other applications.
Q. And can you give His Honor an example of some methods that an Android developer would want to make available from an application?
A. Yes. So let's say I want to write an application that lets other applications place orders for pizzas. So my application may need to have some methods like -- well, you would have to have like place an order, to have it place an order, and let's say to have a check status method that will check on the status of an order that you placed.
Q. So using your pizza example, Ms. Hackborn, can you give us some example names of these methods that we just talked about?
A. Yeah. They are pretty much what I said. So to place an order, you could have the method called placeOrder to do that. And then for checking a status, it could be called checkStatus.
Q. Now, once a developer knows what

methods he or she wants to make available to other applications, what's the next step in the Android process?
A. The next thing you need to do is use the AIDL language to predefine these methods that they are going to use.
Q. And is this the same AIDL tool we talked about earlier?
A. Yes, it is.
Q. Again, remind us, what is the purpose of using this AIDL tool?
A. The AIDL tool generates all the code that's needed to actually do the interprocess communication, to make the method call go from one process to another. So you need to run this tool as part of building your application to compile the code into the application so it can do that.
Q. Now, going back to your pizza example, can you give us an illustration of how a developer might predefine these methods using the AIDL tool?
A. Yeah, I can. Can I write it?
MR. PAK: Your Honor, may the witness approach the white board?

JUDGE CHARNESKI: Sure.
MR. PAK: Thank you.
THE WITNESS: So I am going to draw up here our AIDL file as a box. So we have two methods that we're going to define. The first one we're doing is placeOrder. So I am going to write the name placeOrder. And then we need to say what argument it is going to take. So first thing we need is an address to tell the application where to deliver this order. So we will call that address. And then let's say we also need to say the toppings we want on our pizza to have delivered. So there is toppings there.
And finally we need to say what this will return. And in this case we want to have this return an integer, that's a number that represents the order that was placed.
So the next thing we need to do is define our checkStatus method. Again, you write the name of it. This method is going to take an integer, which is the order number that we had previously placed, and this one will return a time, which is the time at which the order is going to be delivered.

So that's basically the AIDL file.
BY MR. PAK:
Q. Ms. Hackborn, just to summarize what you told us, let's take the check status method. What would be the method name in that particular entry?
A. The name of that method is checkStatus.
Q. And what would be the argument that corresponded to that particular method?
A. The argument there is an integer type, that is the order number.
Q. And what would be the return value for that particular method?
A. The return value is a time type.
Q. And, again, what is the purpose of placing these types of information in the AIDL tool?
A. This is to allow the tools to create the code so your application can actually send and receive these kinds of method calls.
Q. Now, in your explanation, Ms. Hackborn, you also talked about something called a compiler, AIDL compiler. Do you recall that?

A. Yes.
Q. What is the output of the AIDL compiler?
A. The compiler generates Java language source code, the same kind of code you make when you are writing your own application. And so this code then is compiled into the rest of your application as part of building it into something that can actually run.
Q. And is there a name for that code?
A. Yes. This is called an interface stub.
Q. Is that s-t-u-b?
A. S-t-u-b, yes.
Q. Why is this code called a stub?
A. It is a stub because it actually doesn't do anything interesting. All it is there is to actually be able to do the communication across processes, but by itself it doesn't actually implement any behavior that does something that we want.
Q. So after the developer has gotten this interface stub through the AIDL compiler, what does the developer do next?
A. So the next thing they need to do is

actually implement this interface. So, for example, in our pizza application, this is where we actually finally say, okay, when someone wants to place an order, I will go to my restaurant and generate the order for it and return to them the order number that I created for the restaurant.
And then when they want to check the status on it, I will take the order number that they give me and check on my restaurant and see what the status of that is and return it.
Q. So when you say implement, are you talking about writing code?
A. Yes. This is actually writing Java language code that actually implements the behavior that you want.
Q. So once the developer has implemented these interfaces that you described, how can other applications now make use of these interfaces?
A. So to communicate with the application, Android actually doesn't give any way to automatically do this. Once these have been run through the AIDL compiler, the information about the interface is gone as far

as being able to know what is in it.
So what you need to do is actually manually give another application developer an exact copy of that same interface that you had used to build into your application.
Q. And that interface would look something like this at the top?
A. Yeah. It needs to look just like that.
Q. Does it have to be an exact copy of this file?
A. Yes, it does.
Q. Why is that? Why do I have to hand you an exact copy of my interface file?
A. Because AIDL assumes the interface is -- that's how it works. So it does things like when it sends the communication across processes to do place order, instead of saying the name of the method, it actually just sends a number that it knows is that method.
MR. PAK: Now, Your Honor, if I could have the witness approach the white board again.
JUDGE CHARNESKI: Sure.
BY MR. PAK:

Q. Ms. Hackborn, I would like to have you illustrate using your pizza example what you mean by sending this number, rather than the method name.
A. Okay. So at this point we have two processes that we're using, so I am going to do a little squiggly line down the middle here to indicate the division between them. On the right side we will have our pizza application. I will draw a box here for the pizza application.
And we will name this "pizza app." Inside the application we have compiled into it the interface stub for our AIDL file. So I will draw that here as a dotted box.
The stub, this came from the interface. And this stub knows how to take calls coming from other processes into calls in its process.
So likewise on the other side we have our other application. I will draw a box on the left for it. We will call this the client.
This client has also built into itself the same AIDL file with code that was generated. I will represent that also with a

dotted box. And this has the other side where it knows, if someone wants to place an order, how to send that across processes.
So those are two methods. One, two. So when number 1 is sent across processes, this stub over here knows how to turn that into a place order, and two, check status.
Q. Ms. Hackborn, if you could step a little to the side. I am going to ask you a few questions and then have you draw one more item on the board. And if you could talk a little bit louder for the court reporter.
A. Yes.
Q. So in terms of what is actually being sent from the client application to the pizza application, what is being communicated there?
A. What is being communicated is the number of the method to have called and whatever arguments were supplied to it.
Q. So it is just a number, not the method name?
A. It is just a number, yes.
Q. And how does the receiving side know about these numbers and also what methods they correspond to?

A. It only knows of these because that's just the same generated code from the AIDL file assigned those same numbers to those methods.
Q. And when were all these numbers assigned and correlated to the method names?
A. These were assigned as part of building the application.
Q. And when you say building the application, is that before or during the runtime of the application?
A. That is before.
Q. Now, how does this assignment of numbers and sending the numbers across relate to the notion of predefining methods that we heard about earlier today?
A. Well, these have to, for it to know these numbers, you have to predefine these methods as part of building your application. So if you don't know these at runtime, you will never know them.
Q. So what happens if two applications for whatever reason ends up using two different AIDL interface files and, therefore, they are talking a different language, what would happen?

A. Well, I can show you what will happen on here. Let's take an example of, let's do another stub that the client built into itself. So in this case that stub, we will call it the bad stub. This developer didn't take an exact copy of the AIDL. They made their own and they changed the order of the methods in it, which you wouldn't think would make a difference, but it does.

So here this stub has checkStatus first, so that's method number 1. And placeOrder second, so that was method number 2. And now when the developer tries to call placeOrder on this stub, it says call method number 2, and when this arrives in the original stub of the pizza application, it sees, oh, call method number 2, and it thinks it wants to do a check status.

Q. Would that be a good result?

A. That is not a good result.

Q. Okay. If you can take a seat, I will ask you a few more questions on this topic.

So Ms. Hackborn, the first question I have is why doesn't the system just forward the request for place order, number 2, on the

receive side to the code that actually does it?

A. Well, it is impossible for it to do it at this point because all it knows is that some method, given number 2, is supposed to be called. It has no idea that that should actually go to the placeOrder method.

Q. Is there any type of forwarding mechanism within Android for finding the method code that corresponds to a number?

A. No, there is not.

Q. And is there any type of mechanism in Android that figures out whether the method calls on the left-hand side are being made to the right method calls on the right-hand side?

A. No, there is not.

Q. Is there any mechanism in Android to figure out from the client side what methods are actually available on the receive side?

A. No, there is not.

Q. So, again, you would have to exchange these manual files?

A. Yes, you have to exchange the actual files.

Q. What about this scenario, what if I wanted to, in the middle of the application

running, decided to add a new method, for example, add toppings. If I wanted to add toppings to your application while it is running, is that possible in Android?

A. It is not possible.

Q. Again, why is that not possible?

A. So in an example here, you will see the AIDL file has these two methods. If in my code for my application I take this interface and try to call add toppings on it, when I try to compile my application, it will fail. It cannot generate an application that does that because it doesn't know what this add toppings method is. It doesn't know what number to give it.

MR. PAK: Your Honor, at this point I would like to mark this as a demonstrative, DH-RDX-1.

JUDGE CHARNESKI: Okay.

(Respondent Exhibit Number DH-RDX-1 was marked for identification.)

JUDGE CHARNESKI: You are saying DH-RDX-1?

MR. PAK: RDX-1.

JUDGE CHARNESKI: RDX-1; is that

correct?

MR. PAK: Yes, RDX-1.

JUDGE CHARNESKI: You mean we don't have an RDX-1? I find that hard to believe.

MR. PAK: We're using DH as the name.

JUDGE CHARNESKI: That might be a little bit confusing.

MR. PAK: I see. We have been doing that with the other witnesses.

JUDGE CHARNESKI: I know you have been calling them out. Well, let's go with the DH.

MR. PAK: Okay. Perhaps one thing we could do is after the evidence comes in, we can always go back and consecutively number the RDX exhibits and provide you with that on the record. Would that be helpful, Your Honor?

JUDGE CHARNESKI: As long as the record clearly refers, and I know counsel have been putting initials in front of the RDX. And this is the first time I have seen that, but I didn't want to throw anybody off. So it will work this way.

MR. PAK: Okay, great. Thank you.

BY MR. PAK:

Q. Ms. Hackborn, we talked about actually

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	) Investigation No.
CERTAIN PERSONAL DATA AND	) 337-TA-710
MOBILE COMMUNICATIONS DEVICES	)
AND RELATED SOFTWARE.	)

Hearing Room A

United States

International Trade Commission

500 E Street, Southwest

Washington, D.C.

Thursday, May 5, 2011

VOLUME XIV

The parties met, pursuant to the notice of the Judge, at 9:01 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

Q. Then if we go to 72A, this is another slide you have created, correct?
A. Yes.
Q. And what are you illustrating here?
A. So what I am illustrating here is the separate algorithms for the different messages. In particular, we're looking here at RX-5806, IWindow.aidl at 1. And what we see here are the declarations for two different kinds of minutes that the Android system can send. One is the dispatchPointer message. The other is a dispatchTrackBall message.

If we look at the code that the AIDL compiler generates when it is given this, you will see that there is RX-6454, IWindow.Java at 5 through 6, it has generated the algorithm for encoding the dispatchPointer messages. Okay?

Now, the other box indicates there is a separate and distinct algorithm for encoding dispatchTrackBall messages. Since there has to be a different algorithm for every message in the Android system, you can see the Android system can't learn how to send new messages because it would have to have a new algorithm for sending each message.

MR. VERHOEVEN: I would like to move to admit RX-5806 and RX-6454.
MR. HALES: No objection.
MS. JOFFRE: No objection.
JUDGE CHARNESKI: Admitted.
(Respondent Exhibit Numbers RX-5806 and RX-6454 were received into evidence.)
BY MR. VERHOEVEN:
Q. Let's go to the next slide, please, this is MR-RDX-73A.
A. Um-hum.
Q. Can you finish up your point here?
A. This is a summary of the fact that in Android, a new message requires a new encoding algorithm. And Android does not have the capability of learning this new algorithm at runtime, unlike the '721 patent.
Q. And slide 74, can you walk us through what you are illustrating here?
A. Sure. This is the structure for -- what I am illustrating here is the structure for the transmitting function. So if you look at box 502 --
Q. Just for the record, the thing you said this here that we're looking at, it is

figure 5 of JX-1, the '721 patent?
A. Yes. If you look at box 502 of the figure 5 of the '721 patent, you will see it is asking a question: Does an implementation exist in object B?

Now, the Android system doesn't ask this question. It has no need to ask this question. Because the compiler has already guaranteed that such an implementation will exist.

So what the Android system does is it goes every time directly from 501, where it is sending a message from one object to another to box 503 where it executes the method. It does not have box 502. It does not proceed along and invoke forward 504. And this is because the compiler has verified that all messages will have implementations and objects that they are sent to.
Q. And this picture on the right, you are just illustrating Android is a constrained system like you talked about before?
A. That's exactly what I am illustrating there.
Q. Okay. And then slide 75, you have

pulled out some hearing testimony from Apple testimony, Ms. Spielman . This is from the transcript of April 28th, 2011, page 2929, lines 13 through 23.
A. Actually, that would be 19 through 23.
Q. 19 through 23. Thank you, Dr. Rinard. Please explain why you pulled this out.
A. What I am noting here is that Apple's expert, Ms. Spielman, also agrees that Android does not have box 502 or 504. And here is the question.

"Question: Now, you don't have a view on whether Android performs steps 502 or 504, do you?

"Answer: From the code I have examined, it does not perform those because the interfaces are predefined."
Q. Okay. And do you agree with that?
A. I agree with that.
Q. And for that reason, you think there is no infringement of these elements?
A. That's correct.
Q. Let's move onto the next subject, the proxy element that's in claim 1.

ORIGINAL

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	)	Investigation No.
CERTAIN PERSONAL DATA AND	)	337-TA-710
MOBILE COMMUNICATIONS DEVICES	)	
AND RELATED SOFTWARE.	)	

OPEN SESSION

Pages: 4439 through 4783 (with excerpts)

Place: Washington, D.C.

Date: May 5, 2011

MAY - 6 2011

HERITAGE REPORTING CORPORATION
Official Reporters
1220 L Street, N.W., Suite 600
Washington, D.C. 20005
(202) 628-4888
contracts@hrccourtreporters.com

before the execution of that application, that would be different than loading into task address space during runtime, right?

A. I can't comment if you are giving me a hypothetical. I have examined the source code that's present in this case. I have thoroughly looked at how the calls are being made and how the files are being used.

So my evaluation of the source code is complete and how it is being done. So to say if it was being done a different way, that's a different set of source code that is not part of this case right now.

Q. Ms. Spielman, let me ask you a hypothetical question then. Hypothetically speaking, if it turned out that the DEX file was being loaded into the task address space before runtime execution of an application, that would not be covered by the '983 patent claims, correct?

A. If the DEX file existed inside the task address space in the hypothetical situation, which is not the way the source code works that I have examined in this case, then it would be already Mallocked as part of the

task address space, and it would not infringe in that particular situation, but, again, that is a hypothetical situation that would also take into account that there could be many other places in the code that would need to be adjusted. So I just want to make sure it is very clear on the record that that is not the way that the source code works.

Q. Ms. Spielman, you talked about this Mallock command.

A. Yes.

Q. Are you relying on the Mallock command as part of your infringement theory?

A. I wouldn't say -- I am relying on the source code trace that I stepped through to understand that the structures are allocated for holding the contents of the pointer that are then copied into the task address space.

Q. But part of your infringement analysis as you just described it depends on the use of the Mallock command or not, correct?

A. It depends on the use -- well, no, I would not agree with that because there are other memory allocations that can be done in task address space. That is a memory

BEFORE THE

UNITED STATES INTERNATIONAL TRADE COMMISSION

In the Matter of:	) Investigation No.
CERTAIN PERSONAL DATA AND	) 337-TA-710
MOBILE COMMUNICATIONS DEVICES	)
AND RELATED SOFTWARE.	)

Hearing Room A

United States

International Trade Commission

500 E Street, Southwest

Washington, D.C.

Friday, May 6, 2011

VOLUME XV

The parties met, pursuant to the notice of the Judge, at 9:01 a.m.

BEFORE: THE HONORABLE CARL C. CHARNESKI

OPEN SESSION

BY MR. PAK:

Q. And, Ms. Spielman, when you talked earlier in your examination about task address space, did you have this discussion of tasks, task address space and virtual memory regions in mind?

A. Well, task address space is specific to the program executing, so that has to do with the application heap that we have talked about when an application starts. Virtual address space is a different concept that could be used within a program but there is a distinction to be made between the task address space and the virtual address space.

Q. Now, Ms. Spielman, you have a copy of the '983 patent in front of you, right?

A. Yes.

Q. The '983 patent describes the Mach task and the address space of a Mach task in this column that we're seeing on the screen, correct?

A. In the same line numbers that we're looking at?

Q. Yes.

A. Yes, it is talking about a Mach task. Yes.

Q. So you would agree with me that at least in the '983 patent in this context, the task address space consists or contains a number of virtual memory regions?

A. I would agree that in the context of how this reads, that the address space of a Mach task contains a number of virtual memory regions. That's the context here.

Q. And a Mach task is a preferred embodiment of the '983 patent, correct?

A. I am not sure if I would say a Mach task is the preferred embodiment. It certainly is listed in the description here.

Q. Do you remember the micro-kernel that's actually used?

A. Yes.

Q. In the '983 patent?

A. Yes.

Q. What kind of kernel is that?

A. It is a Mach kernel.

Q. Let's shift gears to your infringement analysis. You would agree with me that when an application is launched, the DEX file is loaded

into memory using what is called an mMap or memory map operation?

A. Yes, it uses the mMap system call, that's correct.

Q. And that's the command that's used at the beginning of an application startup to load the DEX file in its entirety into memory, correct?

A. That's correct.

Q. Now, I would like to show you an exhibit that we have seen before. This is CX-7262.

A. Is it in the binder?

Q. I am not sure if it is in your binder but it is here on the screen.

A. Okay.

Q. You were asked about this during your first cross-examination, Ms. Spielman. This is a copy of a Linux manual page for the mMap operation. Do you see that?

A. Yes.

Q. And you recognize this document?

A. I do. I'm assuming it is the same version, but I certainly recognize it.

Q. And manual pages are available for

Linux developers to understand how certain operations work in the Linux operating system, correct?

A. That's correct.

Q. So why don't we, Ryan, have you focus on the description of what the mMap operation performs.

A. Is it possible to get a hard copy of this? Because there is a number of pages.

Q. Sure. I am happy to hand you one now.

A. Thanks.

MR. PAK: May I approach the witness?

JUDGE CHARNESKI: Yes.

THE WITNESS: Although just to be clear, I believe the version that I was given on my direct -- on my cross-examination only had four pages, but I could be mistaken.

BY MR. PAK:

Q. I will represent to you that this has been marked as CX-7262 by the Complainant.

A. Okay.

Q. So, Ryan, let's focus on the top description starting with mMap creates. Ms. Spielman, if you focus on the section that has just been highlighted, you can see that

according to the Linux kernel documentation or the Linux manual page, memory map or mMap creates a new mapping in the virtual address space of the calling process. Do you see that?
A. Correct, yes.
Q. And you agree with that description, correct?
A. Yes, it does create a mapping in the virtual address space that the kernel allocates.
Q. And that's for the process -- calling process, correct?
A. For the calling process.
Q. Let's move to a different topic, Ms. Spielman. We can take that down, Ryan.
You also talked about DEX files. Do you recall that?
A. We have talked extensively about DEX files, yes.
Q. And just to remind everybody, DEX stands for Dalvik executable, correct?
A. That's my understanding that the acronym stands for.
Q. And you would agree with me, Ms. Spielman, that DEX files contain code, correct?

A. I wouldn't agree with that. DEX files contain data that happens to be op code for Java, so the DEX file is a container data file. And that data can be used by the Dalvik virtual machine.
Q. You would agree with me that DEX files contain byte codes?
A. I would agree that the data is byte code, yes.
Q. And you have worked with byte codes in the context of Java programming languages, correct?
A. I have worked with them in the sense that I have compiled Java programs and they get compiled into byte codes, so I am familiar with them in that sense.
Q. And you also understand that Android applications are written in the Java programming language, correct?
A. That's correct.
Q. And, Ms. Spielman, you have testified here that you have extensive experience with the Java programming language?
A. Yes.
Q. And you have written some books about

that?
A. Yes.
Q. So in your extensive experience with Java, have you known people of skill in the art to characterize byte code as executable?
A. Well, byte code is executable in the context of the virtual machine. So the virtual machine will read the byte code and use that to actually execute the appropriate instructions, that that is the point of the Java programming language that the byte code is portable.
Q. So it is executable, correct?
A. The byte code is executable in the context of the virtual machine.
Q. It is the same byte code we're talking about with respect to what is stored in the DEX files, correct?
A. To be technically precise, Java byte code is not compatible with the Android DEX -- the Dalvik byte code. So there were modifications made, so the byte code that is created from the DEX compiler is not compatible with standard byte code.
Q. But, Ms. Spielman, you would agree with me that DEX byte codes are executable in

the same sense that Java byte codes are executable, correct?
A. I would agree that they are executable in the context of the Dalvik virtual machine running the byte code.
Q. And you would agree with me that when byte codes are transferred from the DEX file to the virtual machine, that the byte codes are not modified in any way, correct?
A. Not to my knowledge. I did not see anything in the code that modifies it.
Q. Let's switch topics again, Ms. Spielman. I want to talk about Vernon and Gautron combination.
A. Okay.
Q. That's one of the combinations that you considered as part of your opinions for the '983 patent, correct?
A. That's correct.
Q. Stepping back, based on your testimony, Ms. Spielman, I believe that you are asserting in this case that the '983 patent produced unexpected results; is that right?
A. That's correct.
Q. In particular, you assert that the

Smith's telephone number.
And then if I double-tap on "Bob Jones" in the appointment, what will happen is a contact entry will pop up that says Amy Smith and it will contain Amy Smith's phone number.
Similarly, if I draw a D over the "Bob Jones" over here (indicating) after modifying this, a dialer window will pop up with Amy Smith's phone number.
So as I said, at no point is the text here actually used in the process of doing, in the scenario that Dr. Olsen described.
Q. The example that you have just provided with the contact information being changed to Amy Smith and Amy Smith's phone number, how do you know that that would be the case?
A. I did this myself on the device, as well as I also analyzed the source code to confirm this.
Q. When you say on the device, are you referring to an EO?
A. Yes, the 440.
MR. MIZZO: Your Honor, at this time I would like to move on the confidential record.

Right now it is going to be third-party confidential, but we're going to quickly make our way over to Apple confidential.
JUDGE CHARNESKI: Okay. Just hold on one second, please.
For the sake of the record, Mr. Mizzo, what I want you to do is take a look at the LiveNote, and where Dr. Mowry, I guess it's the paragraph he says "and then if I double-tap on 'Bob Jones,'" read the next two paragraphs. And he was using the pointer to point to the screen. Just make sure that whoever reads this after me can tie up his testimony with what's on the CDX.
If it is clear to you, that's fine. I just want to make sure that you believe the record is clear. Because I believe when Dr. Mowry mentioned the term "here," he was pointing to information, and I am not so sure his testimony will translate to what he is looking at.
If you think it is clear, fine. It didn't seem clear to me.
MR. MIZZO: Thank you, Your Honor.
BY MR. MIZZO:

Q. Just for the sake of clarity, I want to make sure the record is clear, I am now showing the same exact CDX, Number 794, on the ELMO. And if you could briefly walk me through the two scenarios that you were discussing using the depiction of Perspective on the slide. And I will relabel this as CDX-5006. So that way it is clear that this is a different demonstrative than what we were showing before.
Can you begin by describing the double-tap that you were discussing before?
A. Yes.
MR. VAN NEST: Excuse me, Your Honor, I am confused. I am not sure why we're now relabeling this. It is the same, I think it is the same exhibit we had before.
JUDGE CHARNESKI: Okay. There is no need to relabel it.
MR. MIZZO: I wanted to make the record clear, that what I had shown the witness before was not labeled, and now if I am going to write anything on this, this is a new demonstrative.
JUDGE CHARNESKI: Well, I don't want

you to write anything on it. All I said was when I listened to the witness' testimony, and he was using the laser, when I looked at what he said, it didn't seem clear to me, what my observation was. If it were clear to you, fine. It just didn't seem to match up.
Maybe I am off a second. Maybe I am off a beat. But if you think you were clear, you can just move on.
And I pointed you to two paragraphs in his testimony. And if you thought it was unclear, you can just tighten it up, that's all.
MR. KRUPKA: May I have a minute, Your Honor?
JUDGE CHARNESKI: Yes.
BY MR. MIZZO:
Q. If we can go back to the slide CDX-794. I hope it was clear, Your Honor, but I would like to go through it again just to make sure it is abundantly clear what we were discussing.
So, Dr. Mowry, as you are discussing these scenarios, the double-tap and the D gesture, what I would like you to do is be a