Episode Book

EPISODE #5528

What Your eSIM Manager Is Actually Doing

PUBLISHED	24 September 2026 at 14:56 UTC
DURATION	29:03
LISTEN	myweirdprompts.com/episode/esim-provisioning-explained/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:37 UTC

SYNOPSIS

When a physical-to-eSIM adapter shows you certificate downloads and LPA strings, you're watching a standardized provisioning process that normally happens invisibly. This episode walks through GSMA SGP.22 — the eUICC, the SM-DP+, the LPA, and the ES9+ interface — and explains why the QR code isn't the security: a certificate chain rooted in the GSMA Certificate Issuer is. We also cover what a profile actually contains, why an eSIM is not "a SIM that became software," and where provisioning commonly breaks so you can debug it yourself.

TRANSCRIPT



Corn

Okay, before we get into any of this, I need to say something. I read Daniel's message and my first honest reaction was, and I want to be clear about this, dread.



Herman

Dread.



Corn

Dread. Because there is a specific kind of person who sends you a message that begins "I picked up a physical to eSIM adapter" and ends with "what are the key things to understand about the failure modes," and that person is about to make you learn something.



Herman

That person is our friend, and he's right, and we're going to do it.



Corn

Here's what he wrote in. He got one of these 9ESIM adapters, the physical card that pretends to be an eSIM, and he says it was transformative when he was travelling. It's part of a bigger plan he has, which is to run a local SIM primarily for data and push all the actual telephony through Twilio, which is a whole other episode and we're not doing it today. What he wants is the plumbing.



Herman

The plumbing.



Corn

When he provisioned his first profile, he noticed he was seeing more of it than he'd ever seen on a phone. He imports the QR code into the manager, an LPA string shows up, and then the manager starts doing something, and if it works, he's got a connection. He watched certificate files come down from the carrier. When it didn't work and he went digging in the APN settings, he found the APN had already been filled in for him, the way it would be if he'd shoved a physical card into the slot.



Herman

Which is the thing that made him write in, I think.



Corn

It's the thing that made him write in. So he's got three questions. One, is provisioning an eSIM basically the same process as inserting a physical SIM card? Two, are the messages the manager shows us just what was happening in the background all along, invisibly? And three, what should someone using one of these managers actually understand about the provisioning process and where it breaks, so they can debug it themselves?



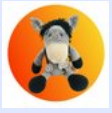
Herman

Good questions. All three of them are good questions.



Corn

So let's pull back the curtain on what's happening in those few seconds. Because the thing I want to land early is that seeing it might be the best debugging tool he didn't know he had.



Herman

Right, so the whole thing is specified. This is not a vendor invention, it's not something 9ESIM made up in a garage. Consumer eSIM is defined by a GSMA specification called SGP.22, and it's currently sitting at version 2.7, published back in April.



Corn

Two point seven. They're still revising it.



Herman

They're still revising it, and there's a whole separate track for IoT called SGP.32 that we might get to at the end. But for the consumer case, SGP.22 defines three actors. The first is the eUICC. That's the secure chip that holds the profiles. On a phone with a built-in eSIM, that's a chip soldered to the board. On Daniel's setup, it's the little programmable card he bought, which is physically SIM-shaped but is really an eUICC with a card edge on it.



Corn

So the adapter is a chip wearing a costume.



Herman

The adapter is a chip wearing a costume. Second actor is the SM-DP+. That's the carrier-side server. Subscription Manager Data Preparation, plus. It's the thing that holds prepared profiles and hands them out. Third is the LPA, the Local Profile Assistant. That's the device-side client. It talks to the SM-DP+, it verifies certificates, it downloads the profile and installs it.



Corn

And the LPA is the piece Daniel is actually looking at when he uses the manager.



Herman

The LPA is the manager. That's the key to his second question and we'll come back to it. There's also an optional fourth piece, the SM-DS, the Discovery Server. That's a directory for push activations, where the carrier triggers a download without you scanning anything. If you've ever had a carrier say "we've pushed an eSIM to your phone," that's the SM-DS doing the introduction.



Corn

And the interface between the LPA and the SM-DP+ has a name?



Herman

It's called ES9+. That's the formal interface. Everything that happens in those few seconds is happening over ES9+.



Corn

So when Daniel scans a QR code and pastes it into the 9ESIM app, which I gather is a build of OpenEUICC, he's triggering a specified sequence of cryptographic operations.



Herman

He's triggering a specified sequence. The manager isn't inventing anything. It's exposing the LPA's work. Which is the whole reason this episode is interesting, because on a normal phone you never see any of it.



Corn

Let's start with the pipeline itself. The sequence of events that starts the moment you scan that code.



Herman

Step one, and this happens before you're involved at all, the carrier prepares a profile on the SM-DP+. That profile is bound to something called a matching ID. Sometimes it's also bound to a specific EID, the identifier of a specific chip, but not always.



Corn

And the EID is the chip's own serial number?



Herman

It's a thirty-two digit number assigned to the chip. And unlike the ICCID, which belongs to the subscription, the EID stays with the eUICC. That distinction matters later, so hold onto it.



Corn

Held.



Herman

Step two, you get an activation code. That's the QR code, or the string underneath it. Step three, the LPA parses that code. Step four, it opens a TLS connection to the SM-DP+ and the two sides run mutual authentication using certificates. Step five, the SM-DP+ creates what's called a Bound Profile Package, protected for the target eUICC. Step six, that package is delivered and installed inside the eUICC. Step seven, you enable the profile. Step eight, the phone authenticates to the mobile network using the network access application that came in the profile.



Corn

Eight steps, a few seconds, and Daniel watched maybe four of them scroll past in a log window.



Herman

He watched the interesting four.



Corn

Let's take the string apart, because that's the piece he can actually see. The LPA string.



Herman

So the format is LPA colon one, dollar sign, the SM-DP+ address, dollar sign, the matching ID. Three fields separated by dollar signs. The LPA colon one is the format and protocol version. The middle field is the address of the server you're about to talk to. The



Herman (continued)

last field is the matching ID.



Corn

And the matching ID is what, exactly?



Herman

The matching ID is a claim ticket. It identifies the profile the carrier has already prepared for you. It is not a secret. That's the part people get wrong. You could read it off a screenshot and it wouldn't help you, because it's only meaningful to the SM-DP+ that issued it, and it's consumed once.



Corn

Neil Brown wrote up his own setup with one of these cards, and he captured a real string from a LycaMobile eSIM. It reads LPA colon one, dollar sign, dp-plus-par07-01 dot oasis-smartsim dot com, dollar sign, then a matching ID starting XHY48, and then a fourth field with a confirmation code.



Herman

Right, and that fourth field is optional. There's a confirmation code that acts as an extra password, and an optional OID after that. And one small detail, if you're typing the code in by hand rather than scanning it, you drop the LPA prefix. The LPA colon is a QR-only convention. The underlying code starts at the one.



Corn

So the QR code is just a text string with a claim ticket in it.



Herman

The QR code is just a text string with a claim ticket in it. There is no secret key in the QR code. And this is where I want to spend some time, because Daniel specifically noticed certificate files coming down from the carrier, and that observation is the whole answer to



Herman (continued)

"is this safe."



Corn

Go.



Herman

The security comes from mutual TLS and certificate chains rooted in something called the GSMA Certificate Issuer, the CI. Legitimate eUICCs hold certificates that chain back to the GSMA CI. Legitimate SM-DP+ servers hold certificates that chain back to the same root. During provisioning, both sides check each other's certificates.



Corn

Both directions.



Herman

Both directions. That's what mutual means. A fake server can't impersonate an accredited SM-DP+, because it can't produce a certificate that chains to the GSMA root. And a random chip can't pretend to be a certified eUICC, for the same reason.



Corn

So the certificate files Daniel watched download are that chain being exchanged.



Herman

That's exactly what they are. The CI-rooted certificate chain and the ECASD certificates, exchanged during mutual authentication. And you can actually see the eUICC's side of it if you query the card. Neil Brown ran a chip info command against his and got back fields called euiccCiPKIdListForVerification and euiccCiPKIdListForSigning, both starting 8137, and a SAS accreditation number, KN-DN-UP-0924, and a default DP address pointing at a Kigen server with a root discovery server at lpa dot ds dot gsma dot com.



Corn

That's a lot of identifiers for a card that costs thirty quid.



Herman

It's a lot of identifiers because it's a real PKI. And then there's the encryption layer on top. The SM-DP+ encrypts each profile with keys bound to the target eUICC's EID. So even if you captured the entire download off the wire, you couldn't install it anywhere else. Only that specific secure element can decrypt it.



Corn

So the answer to "is it safe to download a SIM over the internet" is really "do you trust the certificate chain."



Herman

That's the reframe. The QR code isn't the security. The matching ID isn't the security. The security is a public key infrastructure that both ends have to satisfy before a single byte of profile moves.



Corn

Which is a much better answer than "yes it's encrypted."



Herman

It's a much better answer, and it's the one that tells you what to look at when it fails. Because if the certificate check is where the trust lives, the certificate check is also where a whole class of failures lives.



Corn

So that's the mechanism. But Daniel asked a deeper question, and it's the one I actually find more interesting. Is this essentially the same as inserting a physical SIM? And what happens when it goes wrong?



Herman

The end state is the same. This is the part that surprises people. The profile contains the network access application, the IMSI, the authentication keys, the operator files. That is exactly what a physical SIM holds. When your phone attaches to the network with an eSIM profile, it's authenticating the same way it would with a card.



Corn

So the network doesn't know the difference.



Herman

The network doesn't care. From the network's point of view, a subscriber is a subscriber. The difference is when and how the credentials arrive. On a physical SIM, they're written before you ever get the card. On an eUICC, they're downloaded later over the internet into a tamper-resistant secure element.



Corn

And this is where I want to push on the popular framing, because the popular framing is "the SIM became software."



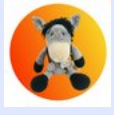
Herman

It didn't. The credentials still live in a secure element. The eUICC is a tamper-resistant chip. What changed is that profiles can be downloaded, switched, disabled, and removed without swapping hardware. That's the actual change. Not the security model, the logistics.



Corn

The card stopped being disposable.



Herman

The card stopped being disposable, and the subscription stopped being welded to it. Which is a much bigger deal than "software SIM."



Corn

Okay, so APN settings. Daniel was surprised they'd been provisioned for the carrier automatically.



Herman

Most profiles configure the APN automatically, because the profile carries operator files. So the APNs populate much the way they would with a physical SIM. It feels like magic the first time, and then you notice it's just the profile doing its job.



Corn

But not always.



Herman

Not always. Neil Brown still had to set the APN manually on Linux for his LycaMobile profile. So if you're debugging and the APN didn't auto-populate, that is not necessarily a provisioning failure. It might just be a profile that doesn't carry the settings, or an OS that doesn't read them the way the profile expects.



Corn

That's a good debugging note on its own. The absence of auto-configuration isn't evidence of a broken download.



Herman

It isn't. It's a separate axis. Which brings us to Daniel's second question, and I think the answer is the most satisfying one in the whole episode.



Corn

Are the messages the manager shows us just what was happening in the background all along?



Herman

Largely yes, with one important correction that I'll let our producer make in a minute.



Corn

That's a tease.



Herman

It's a tease and I'm standing by it. But the short answer is yes. The steps the manager exposes, parse the activation code, open TLS to the SM-DP+, check certificates, download the Bound Profile Package, install, enable, those are the specified SGP.22 flow. On a phone with a built-in eSIM, the operating system's LPA does all of this silently. You see a spinner and then a checkmark.



Corn

And on Daniel's setup, the manager is the LPA.



Herman

The manager is the LPA. That's the whole trick. 9ESIM's app is a version of OpenEUICC, and there are others, lpac, EasyLPAC, MiniLPA. They're all LPA implementations. So when you use one, you're not seeing a special debug mode. You're seeing the actual protocol, because you're looking at the actual client that runs it.



Corn

So it's not that the manager adds visibility. It's that the phone's OS removes it.



Herman

That's the right way to put it. Nobody bolted a logging layer onto the eSIM stack. The phone just hides the LPA behind a settings screen, and the manager doesn't.



Corn

Which means the visibility is free. You get it by using the tool.



Herman

You get it by using the tool, and it is a debugging superpower. This is the thing I'd want Daniel to take away. Every time an eSIM download fails on a normal phone, you get one of two outcomes, a spinner that never resolves, or a generic error. On a manager, you get the actual step it died on. That's the difference between guessing and knowing.



Corn

So let's do the failure pattern. This is the part he actually asked for.



Herman

The single biggest one, and I want to say this clearly because it accounts for an enormous share of support volume, is that delivery and enablement are two independent things. Downloading and installing a profile does not enable it.



Corn

Say that again, because I think people will nod and then do it wrong anyway.



Herman

Downloading and installing a profile does not enable it. A profile on an eUICC is in one of three states. Enabled, disabled, or deleted. You can have a profile that's fully installed, visible in your settings, listed by name, and switched off. And if it's off, you consume zero data.



Corn

Zero data consumed.



Herman

Zero data consumed with an installed profile is nearly diagnostic on its own. If you're looking at a profile that exists and a data counter that hasn't moved, you don't have a network problem. You have a switch that's off.



Corn

That's the one I'd frame on the wall.



Herman

It's the one I'd frame on the wall. And there's a second layer of confusion on top of it, which is that the commercial statuses your carrier shows you are not the spec statuses. You'll see things like GOT_RESOURCE, which means the profile was allocated but nothing reached a device. RELEASED, which means the QR code is valid and waiting. IN_USE, which means it's been downloaded and installed. And ENABLED or DISABLED, which is whether the line is on.



Corn

And IN_USE sounds like it means everything is fine.



Herman

IN_USE sounds like it means everything is fine, and it does not. IN_USE is not a health indicator. It means the profile reached a device. It says nothing about whether that device is online, whether the line is enabled, or whether the subscriber can actually attach. You can be IN_USE and completely offline, and the dashboard will look perfectly healthy.



Corn

So the carrier dashboard and the device are telling you two different stories, and the device is the one that's true.



Herman

The device is the one that's true. That's why the data counter is the field to watch. Everything else can be true while the device is offline.



Corn

What else trips people up?



Herman

Data-only profiles showing no bars. This one gets people constantly. A data-only profile has no voice and no SMS registration. So the status bar may show no operator name and no bars at all, while data is working perfectly. People look at that and conclude the SIM is dead, and they start deleting profiles and re-downloading, which is the worst possible response.



Corn

Because deleting the profile consumes the activation code.



Herman

Deleting the profile consumes the activation code. This is a rule worth saying out loud. One activation code, one install. Once the profile is downloaded, that matching ID is spent. If you delete the profile, you don't get to re-download it with the same code. You need a new eSIM from the carrier.



Corn

So the instinct to nuke it and start over is actively destructive here.



Herman

It's actively destructive. And it's the instinct everybody has, because with a physical SIM, reseating the card is free and harmless. With an eSIM, deleting the profile is a one-way door.



Corn

Okay. Android error codes. You said the manager gives you the actual step it died on. What does that look like?



Herman

On Android, from version eleven onward, there's a detailed code attached to the failure. So instead of "couldn't activate," you get something specific.
ERROR_CERTIFICATE_ERROR means the TLS connection hit a keystore or certificate problem. ERROR_CONNECTION_ERROR means it couldn't reach the server at all. ERROR_ADDRESS_MISSING means there was no SM-DP+ address in the code you gave it.



Corn

Which points straight back at the LPA string.



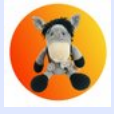
Herman

Straight back at the string. ERROR_INVALID_ACTIVATION_CODE, self-explanatory. ERROR_NO_PROFILES_AVAILABLE, the server has nothing for you. ERROR_CARRIER_LOCKED, the device is locked to a carrier. ERROR_DISALLOWED_BY_PPR, blocked by a profile policy rule, which is the carrier saying no. ERROR_INCOMPATIBLE_CARRIER, the carrier is blacklisted on that device. ERROR_EUICC_INSUFFICIENT_MEMORY, which is real, these cards have finite space. ERROR_INSTALL_PROFILE, which usually means the profile already exists or the ICCID doesn't match. And ERROR_EUICC_MISSING, which means the device can't see the chip at all.



Corn

And that last one, on an adapter, is a physical problem.



Herman

On an adapter, that's a seating problem. That's the one failure pattern where the answer is "take it out and put it back in."



Corn

There's a second error family though, from the server side.



Herman

There is. The SM-DP+ and the SM-DS return SubjectCode and ReasonCode pairs, defined in the spec. So you'll see something like SubjectCode eight point one point one with ReasonCode three point eight, and that specific pair means the EID is bound to another device.



Corn

Meaning someone already installed that profile on a different chip.



Herman

Meaning the profile is locked to a chip that isn't yours. Which is the system working correctly, incidentally. That binding is the security feature we talked about. It's just that when it fires, it looks like a failure.



Corn

Give me the ordinary download failures too, the ones that aren't error codes.



Herman

QR code not scanning, which is usually bad print, wrong format, or camera permissions. "Unable to activate," which is usually the SM-DP+ being unreachable, a wrong address, a firewall, or, and this one catches people, a device clock that's wrong.



Corn

A wrong clock breaks TLS.



Herman

A wrong clock breaks certificate validation, because certificates have validity windows. If your device thinks it's 2019, the certificate looks expired or not yet valid, and the handshake fails. Downloads dying at fifty to seventy percent, which is usually a weak signal or a Wi-Fi drop or a server timeout. SIM lock, where the profile downloads fine but the device can't attach. And no signal after activation, which is usually incomplete provisioning or an APN that didn't configure.



Corn

And the shell trap.



Herman

And the shell trap, which is my favourite failure pattern in this entire episode because it isn't a protocol failure at all. If you use lpac from a command line and you pass the activation code without quoting it, the dollar signs in the string get interpreted as variable expansion by the shell. So the command fails, and it looks like the tool rejected your code.



Corn

But the code was fine.



Herman

The code was fine. The shell ate it. You have to single-quote the whole string. And this is a perfect illustration of the debugging principle, which is that the error you see is not always the layer the error is in.



Corn

That's the whole episode in one sentence.



Herman

It's the whole episode in one sentence. And I want to put the counterweight in here, because we've been fairly positive about this and Daniel's experience was good, but there's a real cost to this architecture when it goes wrong.



Corn

The Ars Technica piece.



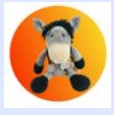
Herman

Ryan Whitwam wrote up switching to eSIM and regretting it. His transfer went sideways, and he ended up locked out of his own phone number. And the reason it was so bad is that the carrier's account recovery ran through SMS to that number. So the verification channel and the broken thing were the same thing.



Corn

That's a grim failure pattern. You can't receive the code that would fix the thing that's stopping you receiving codes.



Herman

And with a physical SIM, that's much harder to do to yourself, because the credential is a piece of plastic in your hand. You can move it to another device. With an eSIM, if the profile is gone and the number is your recovery channel, you're in a loop that only the carrier can break. His line was that a physical SIM is essentially foolproof and eSIM is not.



Corn

Which is worth sitting with, because the elegance of the provisioning pipeline and the elegance of the recovery path are two different things. The pipeline is beautiful. The recovery path is a text message.



Herman

The pipeline is beautiful and the recovery path is a text message. That's exactly the tension.



Corn

There's a detail I want to name before we hand over, because it's the one Daniel noticed and it's a good instinct. He said he saw certificate files downloading from the mobile network operator. And that's the right observation, but the phrasing is slightly off in a way that matters for debugging.



Herman

The certificates aren't downloaded from the operator in the sense of the operator's web server. They're exchanged over the ES9+ interface between the LPA and the SM-DP+, and both sides present chains rooted in the GSMA CI. The operator runs the SM-DP+, so from the outside it looks like the operator sent you a file. But the trust anchor isn't the operator. It's the GSMA root.



Corn

Which is why a certificate error is a different kind of problem from a connection error.



Herman

It's a completely different problem. A connection error means you couldn't get to the server. A certificate error means you got there and didn't believe it. Those two failures look similar in a log and they have nothing in common.



Corn

There's someone here who's been waiting to weigh in on this. And he's got a correction about what the physical SIM world was actually doing behind the scenes.



Hilbert

The OTA updates.



Corn

Sorry?



Hilbert

The over-the-air updates. Carriers pushed them to SIM cards constantly. Roaming tables, bug fixes, new network entries. Silent. The user never saw anything. That was the invisible machinery in the physical world and it was just as invisible as the provisioning steps are now.



Herman

So the correction is that the physical SIM world had its own hidden layer.



Hilbert

It had its own hidden layer. The difference isn't that eSIM has more machinery. It's that the machinery moved off the carrier's servers and onto the device, and the device shows you the receipts.



Corn

And you know this from where, exactly?



Hilbert

I worked a stint at an MVNO. SIM provisioning team. Back when provisioning meant writing files to physical cards in a machine the size of a small fridge. Card trays going in, cards coming out. It had a particular smell. And when it rejected a batch, you heard it. There was a sound.



Herman

The updates you mentioned, the OTA ones. How did those actually reach the card?



Hilbert

Binary SMS. Silent texts the user never saw. The card itself had a little interpreter to process them. That's the part people forget. The SIM was always a small computer running code. It had a processor and a filesystem and it executed commands.



Corn

So the card was never passive.



Hilbert

The card was never passive. It just didn't have a network connection of its own. eSIM gave it one and made it bigger.



Herman

That's the thing I'd been circling all episode. The eUICC isn't a new kind of object. It's the old object with more room and its own link to the server.



Hilbert

Mostly. At the MVNO, the most common support call was "my SIM stopped working." And the fix was almost always the same. An OTA update had failed and left the card in a weird state. It was still there. It just wasn't doing anything.



Corn

Which is the installed but disabled trap.



Hilbert

Same shape. Different decade.



Herman

So the failure pattern we spent twenty minutes on is older than eSIM.



Hilbert

Much older. The delivery path changed. The way it breaks didn't.



Corn

Do you still have one? The machine?



Hilbert

I've got one in the garage. For parts.



Corn

For parts.



Hilbert

For parts. Anyway. Someone's waiting for me in the car park and has been for a while.



Herman

So the machinery was always there. It just moved.



Corn

It moved, and the receipts came with it. Which leaves a few things open, and I don't have answers to any of them. The first is what SGP.32 does to this. That's the IoT track, and it changes the provisioning model in ways that will eventually wash back into the consumer side, and I don't think anyone knows yet what that looks like for a person holding a phone.



Herman

The second is whether this visibility survives. Right now you get it because you chose a manager. If the industry decides that raw protocol steps are a support burden, that window closes and we're back to spinners.



Corn

The third is the one that actually worries me. What happens when the certificate chain breaks. If the GSMA root is ever the problem, there's no user-side recovery. You can't reseat a certificate.



Herman

You can't. And as more devices go eSIM-only, that stops being an edge case. The Ars Technica scenario gets more likely, not less, because the recovery channel and the failed thing keep being the same thing.



Corn

But for anyone willing to look under the hood, the visibility is a gift. That's the part I'd keep.



Herman

That's the part I'd keep too. Thanks as always to Hilbert Flumingtop for producing.



Corn

This has been My Weird Prompts. If this deep dive was useful, leave us a review, tell a friend, and check the show notes for the sources we mentioned.



Herman

We'll be back soon.