THE WIBU-MAGAZINE

KEYnote52



FALL/WINTER 2026

Rate Limits: Implementing AI-Style Token Models with CodeMeter

Highlights

- Automatic Function Encryption Without Source Code Changes
- Case Study | SICK: Scaling Industrial Software Business Models with CodeMeter in Sensor Automation
- Token-Based Licensing: Consumption-Based Software Licensing with CodeMeter

WIBU SYSTEMS

Content

Case Study | SICK: Scaling Industrial Software Business Models with CodeMeter in Sensor Automation 3

Token-Based Licensing: Consumption-Based Software Licensing with CodeMeter 4

Shared License Pool: Flexible Software Licensing with a Shared Token Pool 7

Rate Limits: Implementing AI-Style Token Models with CodeMeter 11

JSON Web Tokens: Compact Licenses for Small Devices and Offline Applications 14

Automatic Function Encryption Without Source Code Changes 17

Still Shipping Parts, or Printing Them On Demand? 20

India's Industrial Ambitions Meet Software Expertise 22

Three Awards, One Mission: Protecting Digital Innovation 23

When Licensing Becomes Strategic Infrastructure



Software was once treated largely as a feature of a physical product. Today, it increasingly defines what products can do, how securely they operate and evolve, and how manufacturers create value throughout their life cycles. This shift makes licensing a strategic concern.

Licensing initially relied on local entitlements. Network floating licenses allowed organizations to share software efficiently across teams. Named user licenses connected rights more closely to individuals and digital identities. Token-based licensing extends this progression, enabling finely structured consumption models and closer alignment between software use and commercial value.

These models do not necessarily replace one another. Different products, markets, and operating environments require different approaches. The strategic capability lies in supporting these models securely, consistently, and in combinations that can evolve with the business.

"Licensing has become the control layer connecting software innovation with security, commercial value, and responsible operation."

That control layer matters because monetization cannot be separated from trust. Licenses must be protected against manipulation and unauthorized use, while ensuring that only authorized users, devices, and software functions receive the intended rights. Secure licensing does not create safety or compliance on its own, but without it, neither can be governed reliably across software's operational life.

Delivering secure licensing across changing business models also requires a close understanding of the markets in which customers operate. By opening our Indian subsidiary, we are moving closer to one of the world's most dynamic technology and industrial markets. Local presence enables closer cooperation with Indian customers and innovators, supported by the experience and infrastructure of our global organization.

As software becomes more valuable, licensing must become more adaptable; as systems become more connected, it must become more secure. Wibu-Systems brings these requirements together, enabling customers to protect what they create, choose how they bring it to market, and grow with confidence.

Best regards,
Oliver Winzenried
CEO



Scaling Industrial Business Models in Sensor Automation

SICK
Sensor Intelligence

SICK is expanding beyond intelligent sensors and integrated automation solutions with digitally enhanced products, standalone software, and cloud-based services. To support this evolution, the company needed a consistent way to protect software intellectual property, manage licenses at scale, and enable digital business models across manufacturing and logistics. CodeMeter provides the protection, licensing, and delivery foundation for this growing portfolio.

The Challenge

SICK's software-driven offerings span a broad technical landscape, from standalone applications and device-level functionality to cloud services. Licensing therefore has to work in connected environments as well as on shop floors with limited or no connectivity. The approach must support varied delivery scenarios, integrate with enterprise and production processes, and give customers a way to receive, activate, and manage their licenses. The challenge was not a single application, but a scalable foundation that could evolve with multiple product lines and business models.

The Solution

Starting in 2021, SICK introduced CodeMeter first for standalone software, then for digitally enhanced products, and subsequently for cloud services. AxProtector supports software protection, while CodeMeter Runtime and CodeMeter Embedded enable device-level licensing. Depending on the deployment scenario, licenses are held in CmDongles, CmActLicenses, or CmCloudLiteContainers. CodeMeter License Central centralizes license management and distribution across SICK's growing portfolio and varied licensing environment. For digitally enhanced products, licensing is integrated into production and activated during device commissioning. Standalone software and cloud services use ticket-based activation, enabling customers to activate and manage licenses independently. Tickets accompany standard product shipments or are supplied digitally, while integration with e-commerce, CRM, and ERP environments supports automated license fulfillment and management.

The Results

SICK has established a common licensing framework that accommodates offline, device-based, and cloud-based scenarios while supporting the company's transition toward software-driven business models. Adoption of digital licensing is growing strongly across multiple product lines. The continuing rollout gives SICK a scalable basis for delivering additional customer value through software and digital services, with a customer-facing license portal, broader cloud licensing, global expansion, and pay-per-use models planned for future development.

The Company

SICK AG is a global technology company for intelligent sensors and integrated solutions in industrial automation. Founded in 1946 and headquartered in Waldkirch near Freiburg, Germany, the company serves manufacturing and logistics customers around the world. With more than 10,000 employees and EUR 1.85 billion in sales in 2025, SICK combines sensor expertise with a growing portfolio of software and digital services designed to support greater efficiency, safety, and sustainability across industry.



Christoph Mueller, Senior Vice President Integrated Automation, SICK AG
"Our customers expect software and digital services to be delivered with the same reliability and simplicity as our physical products. CodeMeter helps us provide a consistent licensing experience across our portfolio while enabling scalable licensing and entitlement processes across the entire customer lifecycle."

SUCCESS STORY



Token-Based Software Licensing: Consumption-Based Licensing with CodeMeter

Software licensing does not have to be limited to users, devices, or license terms. Consumption-based licensing models allow software vendors to charge customers based on their actual usage. With its Unit Counter, CodeMeter provides proven technology for this purpose supporting both online and offline use.

Tokens as Units of Consumption

In consumption-based licensing, a token represents a defined amount of usage rights. Each action within a software application can be assigned its own token cost. For example, a simple calculation might cost one token, while a complex analysis requires ten or twenty tokens.

The available token balance decreases as the software is used. Once the balance is insufficient for a particular function, that function can no longer be executed. Other functions that require fewer tokens may still be available.

For software vendors, this model opens up new monetization opportunities. Customers no longer need to pay for unlimited usage. Instead, they purchase only the capacity they need.

Prepaid and Postpaid

There are two basic billing models:

- In the **Prepaid Model**, customers purchase a defined number of tokens in advance. These are credited to their license container and consumed as they use the software. Once the balance is depleted, customers can purchase additional tokens.
- In the **Postpaid Model**, billing takes place after usage. Customers initially receive a sufficiently large allocation. Actual consumption is measured regularly and reported to the software vendor, who calculates the number of units consumed based on the difference between the initial and remaining balances.

Both models use the same technical mechanism. The main difference lies in when billing occurs and how the commercial process is managed.

The Unit Counter as the Technical Foundation

CodeMeter has supported consumption-based licensing since its inception. The core feature is the **Unit Counter**, a license option that provides a protected counter within a CodeMeter license container.

Software vendors can initialize the counter with a defined value and later update it. Two options are available:

- **Additive update:** New units are added to the existing balance. For example, if a customer has 27 tokens remaining and purchases another 100, the new balance is 127 tokens.
- **Absolute update:** The counter is set to a specified value, such as 100 tokens, regardless of the previous balance.

This distinction matters because the two methods support different business models. An additive update is particularly useful for replenishing a balance, while an absolute update allows a new allocation to be set explicitly.

The protected application can only decrement the Unit Counter, while the balance can be increased only through a license updated issue by the software vendor.

Security Through Cryptographic Operations

A key advantage over a simple software counter is protection against tampering. Decrementing the Unit Counter can be tied to a cryptographic operation, such as encryption, decryption, or a digital signature.

This allows the consumption of a unit to be linked to a protected action.

One potential application is point-of-sale systems, where the number of invoices generated must be recorded in a traceable manner. For each invoice, a hash value can be generated and digitally signed along with the date and the current counter value.

The resulting logs make it possible to verify later that the recorded transactions remain unchanged and can be checked against the corresponding invoice records. Depending on the implementation, this can help detect tampering and improve usage traceability.

This principle applies beyond point-of-sale systems. Cryptographically linking consumption to processing can also be useful in industrial production, document processing, and other billable operations.

Online and Offline Without Restricting the Licensing Model

Many consumption-based business models require a permanent Internet connection. This is a natural fit for cloud services since consumption can be recorded and billed centrally.

CodeMeter takes a different approach: Tokens are securely stored in a license container with several container technologies available for this purpose.

CmCloudContainer allows licenses to be provided online. **CmActLicense** binds licenses to a specific computer, while **CmDongle** provides a portable, hardware-based license container.

This makes consumption-based licensing models possible even in fully isolated environments. Software can securely use tokens even in air-gapped systems with no connection to the Internet.

This is particularly relevant for industrial applications where production equipment is not permitted to communicate continuously with external systems for security or operational reasons.

Providing and Replenishing Token Packages

A typical use case with three software functions illustrates how easily this model can be implemented. The first function requires ten tokens, the second twenty, and the third just one token per execution.

These functions cannot be used without an available token balance.

The software vendor provides a license package containing 100 tokens through **CodeMeter License Central** and gen-



erates a corresponding ticket. The customer redeems the ticket through **CodeMeter License Portal** or WebDepot and activates the tokens in the license container of their choice.

The 100 tokens are then immediately available in the application. Each time a function is executed, the specified number of tokens is deducted.

Tokens are consumed and are not returned when the function finishes. One implementation detail is when tokens are consumed. This can be configured to occur either when the function starts or when it completes successfully. If the balance is insufficient for a particular function, execution is denied.

Customers can purchase another token package at any time and replenish their balance through the same activation mechanism. With this type of update, the remaining tokens are added to the new allocation. For example, if six tokens remain, adding 100 tokens brings the total balance to 106.

Activation can take place through CodeMeter License Portal, through WebDepot, or directly within the vendor's own application. Updates can also be performed entirely offline by exchanging license files.

New Revenue Opportunities Through Recurring Consumption

Consumption-based licensing creates a direct link between actual software usage and revenue. Instead of relying solely on one-time license sales, software vendors can generate recurring revenue as customers purchase additional units.

Customers benefit from greater flexibility in usage. They do not need to purchase a comprehensive license for functions they rarely use. Instead, they can manage consumption according to their actual needs.

With the Unit Counter, CodeMeter provides the technical foundation for these models. The combination of protected consumption tracking, flexible deployment, and support for offline environments enables software vendors to implement usage-based business models even where conventional cloud-based billing systems are not suitable.

Key Advantage: Consumption-Based Licensing Works Offline, Too

Consumption-based business models are not limited to the cloud. CodeMeter's protected Unit Counter allows usage units to be securely tracked and billed even without a permanent Internet connection. Software vendors can implement flexible prepaid and postpaid models even in fully isolated production environments. The technical infrastructure does not dictate the business model. Instead, the business model determines the most suitable licensing approach.





Shared License Pools: Flexible Software Licensing through a Shared Token Pool

Not every use of software has to permanently consume license units. With a Shared License Pool, software vendors can provide a shared allocation of tokens that applications and features draw on as needed. CodeMeter supports this licensing model on individual computers as well as in local networks and cloud environments.

Using Tokens Rather Than Consuming Them

In token-based software licensing, there is a fundamental difference between consumable units and reusable tokens. In a consumption-based model, tokens are permanently deducted for each action performed. In a Shared License Pool, tokens are reserved for the duration of their use and then become available again.

When an application closes or a feature finishes running, it releases its reserved tokens. These tokens are then available to other applications or features. The total allocation remains unchanged.

A Shared License Pool therefore allows different users to use different features flexibly. Instead of providing a separate license for each feature or user, multiple features can draw on a shared pool. What matters is how many tokens are needed at the same time.

This model is particularly suitable for software products with different sets of features or fluctuating workloads. Higher-value features can reserve more tokens than basic features. As long as enough unallocated tokens are available, the features can be used in parallel.

License Quantity as the Technical Foundation

CodeMeter uses the **License Quantity** option for Shared License Pools. It specifies how many tokens within a license allocation are available for simultaneous use.

As with Unit Counter for consumable units, the software vendor can set the initial License Quantity and change it later through license updates.

The key difference from Unit Counter lies in what happens during use. With Unit Counter, the counter is permanently reduced. With License Quantity, the total allocation is preserved. Only the number of tokens currently available changes according to the active reservations.

This makes it possible to manage reusable tokens without having to issue a new license each time a feature runs.

Dynamic Allocation Across Features

An application with three different features illustrates how this works in practice.

The first feature requires ten tokens, the second twenty tokens, and the third just one token. The customer has a Shared License Pool containing a total of 25 tokens.

When the customer starts the first feature, CodeMeter reserves ten tokens, leaving 15 tokens available. The third feature can run in parallel, reducing the available allocation to 14 tokens.

The second feature, however, requires twenty tokens and therefore cannot start in this situation. Only after the first feature finishes running and releases its ten tokens are 24 tokens available again. The second feature can now run as well.

This example illustrates how a Shared License Pool controls the simultaneous use of different features. With 25 tokens, either the first or the second feature can run alongside the third. Running all three features simultaneously would require at least 31 tokens.

If the customer needs more capacity on an ongoing basis, they can expand their allocation by purchasing additional tokens. This requires no changes to the software itself.

Automatic Release After Application Crashes

An important question with shared license allocations is what happens when an application terminates unexpectedly.

Under normal circumstances, the software reserves the required tokens when a feature starts and releases them when it finishes. If the application crashes, however, this release mechanism can no longer be executed.

CodeMeter handles this situation by monitoring the processes using licenses. The CodeMeter service runs in the background and regularly checks whether the processes that have reserved licenses are still running.

This verification occurs once a minute. If a process has terminated, its reserved tokens are released automatically.

This prevents a crashed application from tying up tokens indefinitely. Once released, the tokens are available to other applications again.

For software vendors, this mechanism simplifies implementation because the application itself does not have to handle every aspect of these crash scenarios.

Linger Time: Keeping Tokens Reserved Longer

Not every business model calls for tokens to become available again immediately after a feature finishes running.

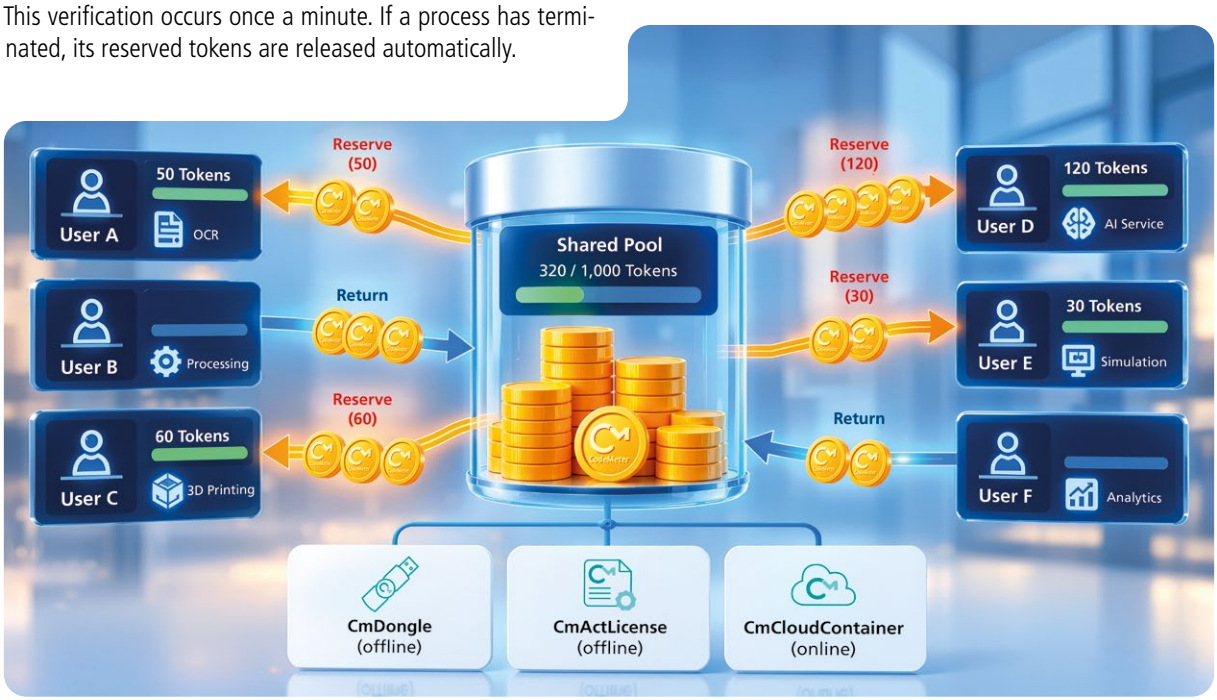
For these cases, CodeMeter offers the **Linger Time** option. It allows a license reservation to remain in effect beyond the actual end of use.

For example, a software vendor can specify that reserved tokens become available to other features again only after an hour or a day.

This is useful when the goal is to prevent users from rapidly switching between features to access a large number of different features with a small token allocation.

Linger Time therefore expands the options for designing a Shared License Pool. It allows vendors to limit the number of features used simultaneously and to influence the time interval before tokens can be reused.

The specific configuration depends on the business model. For some applications, immediate release is ideal, while for others, an extended reservation better suits the intended licensing model.



From Individual Computers to the Cloud

A Shared License Pool is not tied to a particular deployment location. CodeMeter supports this model on individual computers, across local area networks (LANs), and wide area networks (WANs), and in cloud environments.

In a local network, for example, multiple workstations can access a shared license allocation. A VPN connection allows such a pool to be used in a wide area network as well.

Alternatively, tokens can be provided through a CmCloud-Container.

Support for offline environments is particularly valuable for many scenarios. CodeMeter allows license allocations to be used without a connection to an external license server on the Internet. This enables software vendors to deploy Shared License Pools in industrial environments where production systems are disconnected from the Internet for security reasons.

The actual reach of a pool depends on the chosen container technology and the accessible licensing infrastructure. A local pool requires no Internet connection, while a pool spanning multiple networks requires communication between the systems involved.

Providing and Expanding License Capacity

Shared License Pools are provisioned through CodeMeter's established licensing processes. The software vendor defines a corresponding license item in CodeMeter License Central, such as a package of 25 pool tokens.

A ticket is generated for the customer, who can redeem it through CodeMeter License Portal or WebDepot. During activation, the customer selects the desired license container and confirms the license transfer.

Once activation is complete, the tokens are available to the application. The software can then reserve the required number of tokens when individual features start.

To expand the allocation later, the software vendor can offer an upgrade that increases the pool size, for example, from 25 to 50 tokens. This allows software vendors to give customers a smaller pool initially and offer additional capacity as their needs grow.

Visibility Through CodeMeter WebAdmin

Managing a Shared License Pool requires visibility into both the total token allocation and the tokens currently reserved.

CodeMeter provides **CodeMeter WebAdmin** as an administration tool for this purpose.

It displays existing licenses and their respective license options. License Monitoring can also be used to check how many tokens are currently reserved.

This information is displayed through CodeMeter's standard tools. Software vendors therefore do not have to develop their own administration interface to view basic information about license allocations and usage. With Shared License Pools, this visibility helps identify bottlenecks and assess whether the allocation is appropriately sized.

Flexible Licensing Models for Different Requirements

Shared License pools allow software vendors to define the scope of a license independently of a fixed number of individual features. Instead of licensing each feature separately, they can provide a shared allocation from which different features draw according to their respective token requirements.

This makes it possible to cover different combinations of features with the same license package. Within the available allocation, the customer decides which features to use simultaneously.

For software vendors, this creates additional options for structuring their product offerings. Different token packages can represent different capacity levels. Customers can start with a smaller allocation and expand it as their requirements grow.

Unlike consumption-based tokens, repeated use does not automatically require another purchase. Instead, additional revenue is generated when customers need greater license capacity to use more features simultaneously.

The two models therefore address different requirements: Consumption-based tokens (**Unit Counter**) enable billing for individual instances of use, while Shared License Pools (**License Quantity**) control the simultaneous availability of features.

The Key Advantage: One License Pool, Many Ways to Use It

A Shared License Pool makes software licensing more flexible while maintaining control over use. Instead of licensing individual features separately, software vendors can provide a shared token allocation that is distributed dynamically based on current demand. CodeMeter handles secure reservation, automatic release, and monitoring of license capacity locally, across a network, or in the cloud. Rather than licensing every possible use separately, vendors license only the simultaneous capacity customers actually need.



One idea at the right time can change everything.

Subscribe to our blog



READ MORE

Rate Limits: Implementing AI-Style Token Models with CodeMeter

Generative AI has brought token-based business models into the spotlight. Instead of unlimited usage, customers receive defined quotas that can be used within specified periods. These rate limits are not limited to AI services, however. With CodeMeter, software vendors can combine time-based usage limits, automatically replenished quotas, and additional capacity, even without a permanent Internet connection.

When Tokens Are Replenished Regularly

In conventional consumption-based licensing models, a customer purchases a fixed number of tokens. Each use permanently reduces the available balance. Once it is depleted, the customer must purchase more tokens.

Many AI providers use different models. For example, customers receive a certain number of tokens per hour, day, or month. They can use the available tokens within that time window. Once the period ends, the quota is automatically replenished.

This principle is known as **rate limiting**. It limits the use of software or a service within defined time intervals.

For example, a customer might be allowed to use a maximum of 100,000 tokens within four hours and no more than one million tokens per month. Both limits apply simultaneously. Even if the short-term quota has already been replenished, the monthly limit remains in effect.

This allows software vendors to offer different usage tiers. A basic subscription might include lower limits, while higher-tier plans offer larger quotas or shorter waiting times.

Unit Counter and Universal Data

In CodeMeter, these models can be implemented with **Unit Counter** and **Universal Data (UvD)**.

The Unit Counter provides a protected counter that can be used to track or control token consumption. The data fields complement this mechanism with information about time-based usage limits.

Universal Data are protected data fields that can be associated with a license. They can store the number of tokens permitted per period, current consumption, and the time of the next replenishment.

The implementation distinguishes between information that cannot be changed and information that can. Limit definitions are typically stored in read-only data fields. They are set by the software vendor during license activation and cannot be changed by the application.

Other data fields are configured with write permissions for use during operation. The software stores the current status of each quota and its next reset time in these fields.

This separation distinguishes the contractually defined terms of use from consumption data that changes at runtime.

Enforcing Multiple Limits Simultaneously

A key feature of modern AI token models is the combination of different time windows.

Instead of setting a single limit, software vendors can define several independent limits. For example, a maximum of 10,000 tokens could be used within one hour and a maximum of one million within one month.

Every use must be checked against all relevant limits.

If enough tokens remain in the hourly quota but the monthly quota has already been exhausted, further usage is denied. Conversely, a customer may be well below their monthly quota but have to wait temporarily because the hourly limit has been reached.

The requested number of tokens can only be used if all applicable limits permit it.

In practice, combining two or three limits is often sufficient. A short-term session limit can restrict intensive usage over a few hours, while an additional monthly limit defines the subscription's total usage allowance.

This allows both short-term usage spikes and longer-term quotas to be managed.

Rolling Windows and Fixed Reset Times

Not all rate limits have to follow the same timing rules. The CodeMeter reference implementation distinguishes between rolling time windows and fixed, calendar-based intervals.

With a **Rolling Window**, the time window begins when usage starts or according to a predefined time interval. One example is 100 tokens within 60 seconds. Once the window ends, the quota is replenished. Longer intervals, such as one or four hours, are often more appropriate for real-world applications. Short windows, by contrast, can be useful for demonstrations and testing.

With fixed intervals, replenishment takes place at defined times. A five-minute limit can, for example, reset at 00:00, 00:05, 00:10, and 00:15. An hourly limit resets on the hour.

Daily, weekly, and monthly quotas are also possible. By default, these follow Coordinated Universal Time (UTC): A daily limit resets at midnight, a weekly limit on Monday, and a monthly limit on the first day of the month.

The distinction matters: A rolling window has its own individual time window, while fixed intervals follow a shared schedule.

Time Zones and Configurable Offsets

Internally, the reference implementation uses UTC as its time basis. This ensures consistent calculations regardless of where the software is used.

However, quotas do not always need to reset according to UTC. A customer in Germany may expect their daily quota to reset at midnight German time. Another customer may want replenishment to align with local business hours.

The implementation therefore supports configurable time shifts, known as offsets. These offsets adjust the reset times relative to UTC. For example, hourly limits can be aligned with a different hourly schedule, or daily limits can be set to reset at local midnight.

Weekly quotas do not necessarily have to begin on Monday, either. A configurable offset can adjust the start to a Tuesday, for example. Alternative reset times can also be configured



for monthly limits, such as times relative to the beginning or end of the month.

Overflow Tokens for Additional Capacity

What happens when a customer has exhausted their regular quota but urgently needs more tokens?

This is where **Overflow Tokens** come in. They allow additional usage beyond the limits that normally apply. The CodeMeter reference implementation described here supports different strategies.

In the first option, the customer purchases additional units of consumption. These are permanently deducted when used and are no longer available afterward. This follows the principle of conventional consumption-based tokens.

A second option provides additional capacity for a limited time. For example, a customer could receive a higher usage limit for a specified period to accommodate short-term demand spikes.

This additional capacity can have its own reset cycle, such as weekly, monthly, or annually. This supports business models in which customers occasionally need more capacity without having to move permanently to a higher-tier plan.

For software vendors, this also creates an additional revenue opportunity by supplementing regular subscriptions with on-demand extensions.

Controlling the Use of Overflow Tokens

The licensing system does not necessarily need to determine whether additional tokens are used automatically. The application can decide how to handle an exhausted regular quota.

One option is to use overflow tokens automatically as soon as the normal limits are insufficient. Alternatively, the application can first try to perform an action using only the regular quota. If this fails, it can ask the user whether additional tokens should be used.

This distinction is particularly relevant when overflow units incur a charge. Customers can make an informed decision about whether a particular action may incur additional costs or whether they would prefer to wait until the next reset.

The licensing system provides the technical capabilities for this. The software vendor remains responsible for the user experience and for deciding how additional capacity is used.

A Reference Implementation for Software Developers

To simplify the integration of time-based token models, CodeMeter provides a reference implementation with two core functions.

With **GetLimits**, the application can retrieve the usage limits defined for a license. This includes the configured quotas, their current status, and the next reset times.

The second function, **ConsumeCredits**, is used to consume a specified number of tokens.

The reference implementation checks the defined time windows and determines whether the requested usage is permitted. Software vendors therefore do not have to develop all the calculations for different time intervals and their combinations themselves.

The actual integration remains part of the application, however. It determines how many tokens each action requires and how to respond when quotas are exhausted.

Subscriptions Without a Permanent Cloud Connection

Although rate-limit models have become widely known through cloud-based AI services, they do not have to rely on a continuously available billing service.

With CodeMeter, the required license information and current consumption data can be managed securely within a license container. The mechanisms described here can therefore be used in both online and offline environments.

This opens up additional possibilities for software vendors, including locally running AI models, industrial analytics applications, and other software products whose usage needs to be controlled through time-based quotas. Subscription-based licenses can also include time-based usage limits.

The software is thus granted not only permission to run, but also a defined set of rules governing how intensively it may be used within specified periods.

The Key Advantage: AI Business Models Independent of the Cloud

Rate limits turn a simple software license into a usage quota that can be flexibly managed. Combining the Unit Counter, data fields such as Universal Data, and configurable time windows makes it possible to implement AI-style billing models with automatic quota replenishment and additional overflow tokens. CodeMeter's particular advantage is that these models are not limited to cloud services. They can also be used for software running locally and fully isolated environments. This enables modern, usage-based business models even where a permanent Internet connection is unavailable.



JSON Web Tokens: Compact Software Licenses for Small Devices and Offline Applications

Not every token represents a consumable unit or a time-limited usage allowance. JSON Web Tokens (JWTs) take a different approach. Originally JWTs were used to transport rights and identities in a compact format. Today, they are also used to encode entitlements and license information in use cases for tiny devices. CodeMeter enables centralized management of these tokens and supports generating them both online and offline. This allows software vendors to manage licensing for devices that cannot run CodeMeter Runtime.

A Token with a Different Purpose

In software licensing, the term token is used for different concepts. With Consumable Units, it represents a unit that is consumed; with a Shared License Pool, it represents reusable license capacity; and with Rate Limits, it represents a usage allowance tied to a specific time period.

A JSON Web Token, by contrast, is not a business model in its own right, but a technical format for transferring information and entitlements between systems. A JWT can, for example, specify which software features are enabled, which user or device an entitlement applies to, and when it expires.

A digital signature makes it possible to verify the authenticity and integrity of this information without contacting a central license server every time it is used. This makes JWTs particularly useful for applications with limited resources, mobile devices, and scenarios where a continuous network connection cannot be guaranteed.

The Structure of a JSON Web Token

A signed JWT typically consists of three components: a header, a payload, and a signature.

The **header** contains information about the token format and the signing algorithm used. This tells the validating application how to verify the signature.

The **payload** carries the actual information. This can include details about the licensee, permitted software features, administrative privileges, or the license validity period.

For example, the payload can specify that a particular user may use the software and access selected features. It can also define whether that user may create, modify, or delete other users.

The **signature** protects the integrity of the token. It enables the application to verify whether the token was issued by a trusted party and whether its contents have been modified since it was issued.

In most use cases, the information in a JWT is not encrypted. Confidential data should therefore not be stored in the payload without protection. For licensing, the signature is the key element: It allows entitlements to be verified locally without requiring the issuing system to remain continuously accessible.

Binding Licenses to Users and Devices

Licenses often need to be restricted to specific devices or users. As such, a JWT can contain information that allows it to be bound to a particular identity or hardware.

For user-bound licenses, an email address or another unique user identifier can, for example, be stored in the token. The application then checks whether the identity of the signed-in user matches the identity specified in the token.

Device-bound licenses can use hardware attributes. Examples include MAC addresses, hard drive serial numbers, or cryptographic identities provided by a Trusted Platform Module (TPM).

Licensing for Very Small Devices

One important use case for JWTs is licensing devices that cannot run the full CodeMeter Runtime.

Examples include very small embedded systems, microcontrollers, and other devices with limited resources. They may lack sufficient memory, processing power, or a suitable operating system to support a full licensing environment.

Even so, these devices often need to support different features that can be activated based on the license purchased. A compact, signed license file is well suited to this purpose. The device only needs to be able to evaluate the relevant license information and verify the cryptographic signature. A full CodeMeter Runtime is not required on the target device.

CodeMeter can handle license creation and management. While the device itself only needs a correspondingly lightweight license validation implementation.

The format used does not have to be JWT. Proprietary formats and other standardized formats can follow the same principle: protected license information, a defined device binding, and a verifiable digital signature.

Temporary Offline Entitlements

Another use case involves applications that normally operate online but also need to function temporarily without a network connection.

Consider a predictive maintenance application originally developed as a browser-based cloud service. The same functionality is later made available through a mobile app on smartphones or tablets. During use, the device may temporarily lose its Internet connection. The user still needs to be able to continue working for a limited period. A JWT with a limited validity period is ideal here.

While connected to the central system, the application receives a signed token containing the required entitlements. This token can be bound to the user's identity and remain valid for a defined period, such as five minutes, ten minutes, or one hour.

If the device subsequently loses its Internet connection, the application can continue to validate the locally stored token and provide the permitted features. Once the token expires, a new entitlement must be obtained. This allows offline access to be limited precisely without having to move all license management to the mobile device.

Centralized Management with CodeMeter License Central

Even if the target application does not use CodeMeter Runtime, CodeMeter License Central can serve as the central license management platform. The **Custom Licensing Adapter** is available for this purpose. This add-on enables the generation of license formats that are not based directly on conventional CodeMeter license containers.

Software vendors can continue to use existing proprietary license formats while consolidating their licensing processes in CodeMeter License Central. This approach is particularly useful for older devices whose licensing mechanisms cannot readily be changed.



The same principle can be applied to JSON Web Tokens. The software vendor defines its products and license options in CodeMeter License Central. During activation, the Custom Licensing Adapter generates a JWT instead of a conventional CodeMeter license.

The license information from CodeMeter License Central is mapped to the fields in the token. Product Codes, expiration dates, and enabled features, for example, can be transferred directly to the corresponding JWT fields.

Alternatively, multiple license options can be mapped to compact bit fields or other application-specific data structures. This mapping is defined to the appropriate license format in question.

CodeMeter License Central thus remains the central source of license information, while the target devices continue to use a format tailored to their requirements.

Generating Signed Licenses Offline

Licensing becomes particularly challenging when both the target device and the entire production process must operate without an Internet connection. One example is the manufacture of small precision motors. Each motor receives an individual, cryptographically protected license during production.

The production environment is completely isolated from the Internet. Neither inbound nor outbound connections are permitted. Online activation through a central license server is therefore not an option.

CodeMeter supports an alternative approach for these scenarios: The entitlement to generate new licenses is provided in a local CodeMeter license container. The example described uses a CmDongle. From a technical perspective, a similar concept can also be implemented with a CmActLicense.

The license container holds the required cryptographic keys, along with information specifying which licenses may be generated and how many may be issued. This makes it possible to generate licenses entirely offline.

Unit Counter Limits the Number of Licenses

A Unit Counter prevents a local production facility from generating an unlimited number of licenses. The software vendor uses it to specify how many signed license tokens the production environment may issue. One example is an allowance of 1,000 units. The production facility receives a license container authorizing it to generate a maximum of 1,000 licenses. The counter is decremented accordingly for each license issued.

This allows the vendor to control the permitted production quantity even when the facility has no connection to a central license server. In addition, a template in the license container can define which basic license information and features may be included in the generated tokens.

The production software is therefore not free to issue arbitrary entitlements. It can only generate licenses within the specified limits. This combines the flexibility of local license generation with control over permitted license contents and quantities.

Protecting Private Keys in the License Container

Creating digitally signed tokens requires a private cryptographic key. This key requires particularly strong protection. If it falls into the wrong hands, it could be used to generate additional tokens whose signatures the target devices would recognize as valid.

CodeMeter therefore provides a way to store key material securely inside a license container. The **Universal Data (UvD)** license option is generally used for this purpose. Among other things, it supports key material for RSA and elliptic curve cryptography.

The private keys are stored so that they cannot be read out at runtime. Instead, the production application can have CodeMeter perform the designated cryptographic operations. This means the production software can have a token signed without gaining access to the private key itself. Combined with Unit Counter, this makes it possible to limit the number of permitted signing operations or licenses issued.

The target devices then need only the corresponding public key to verify the signature of their license. License generation and verification are thus separated. The private key remains in the protected production environment, while verification can take place on the individual device.

The Key Advantage:
CodeMeter without CodeMeter Runtime

An application does not need to use CodeMeter Runtime to benefit from the CodeMeter licensing infrastructure. With JSON Web Tokens and comparable signed license formats, software vendors can centralize their license management in CodeMeter License Central while supporting very small devices and mobile applications that operate offline.

Protected keys and Unit Counter also make it possible to generate individual licenses in completely isolated production environments. CodeMeter thus combines centralized control over license rights with the freedom to use the appropriate license format on each target device.



Automatic Function Encryption Without Source Code Changes

Functions that require particularly strong protection should ideally be present in memory as plaintext only while they are being executed. Until now, AxProtector variants for native applications offered this capability only through selective encryption of individual methods, which required source code changes. With the new NC technology, this technique, previously known as IxProtector, is now available for Windows and Linux binaries without modifying the source code. Automatic Function Re-encryption also makes it possible to automatically re-encrypt selected functions after execution.

What Is NC?

With the release of CodeMeter Protection Suite 11.80, AxProtector Windows and Linux adopted Native Core (NC) technology, joining AxProtector .NET, Python, and JavaScript. Only the metadata required for execution (the "Security Payload") is now embedded in the program to be protected. The actual licensing and protection logic resides in the separate CodeMeter Protection Suite Runtime (CPSRT), rather than in an AxEngine injected into the protected application, as was previously the case. This allows multiple protected modules within a single process to use the same protection engine, saving space and improving performance. CPSRT itself is strongly protected and hardened against reverse engineering attacks.

Because the protection logic now resides in a separate component, Wibu-Systems can provide software updates without forcing the software vendor to re-encrypt the protected application. The protection component can therefore be updated independently of the protected program. This supports vendors in maintaining their products over time and is also beneficial in the context of regulatory requirements, such as those imposed by the Cyber Resilience Act.



Figure 1: Encryption Using Symbols

Setting Up Protection

For encryption, AxProtector Windows and AxProtector Linux require the binary to be protected, its symbols, and the protection definition in WibuCpsConf. For Windows programs, symbols are typically stored in PDB (Program Database) files, which the compiler and linker generate during the build process. For Linux programs, they are usually embedded directly in the binary as DWARF (Debugging With Arbitrary Record Formats) information. These symbols allow AxProtector to analyze the application more effectively and provide function encryption options. Protection is configured in WibuCpsConf, where licenses, options to prevent reverse engineering, and mappings of functions, classes, or namespaces to licenses can be defined. Based on this configuration, AxProtector with NC technology protects the unprotected binary using modern cryptographic methods and, where required, establishes a strong binding to a license, as shown in Figure 1.

Encrypting the Entire Module

AxProtector with NC technology uses the symmetric cryptographic primitive AES-256 to securely encrypt code and data. Its key length provides a substantial security margin, even when known quantum attack methods are considered. However, data protected in this way remains secure only until the CPU needs it for execution. The fundamental distinction here is between protecting code and data "at rest" and "in use." "At rest" refers to a program stored on disk that is not being executed. "In use," by contrast, refers to a program that has been loaded into memory and is actively being used. The new NC technology integrates multiple layers of protection to secure code and different types of data against a range of attacks.

Module Encryption provides an outer layer of protection for the code and all payload data in a program. Entire code and data sections are encrypted, effectively protecting them against static analysis at rest. This protection is particularly powerful when combined with a base license, since decryption is possible only when that license is present. When a program bound to a license is loaded, the license is checked first; if the check succeeds, the sections are decrypted, using a key inside the license. Figure 2 illustrates how the state of the module changes in memory over time during execution. Initially, all code is encrypted; after loading, the functions shown in turquoise are present in memory as plaintext. This allows the program to achieve maximum performance, since no further decryption is required at runtime. The drawback, however, is that an attacker can extract all functions directly from memory and obtain all the information through a memory dump at any point after startup.

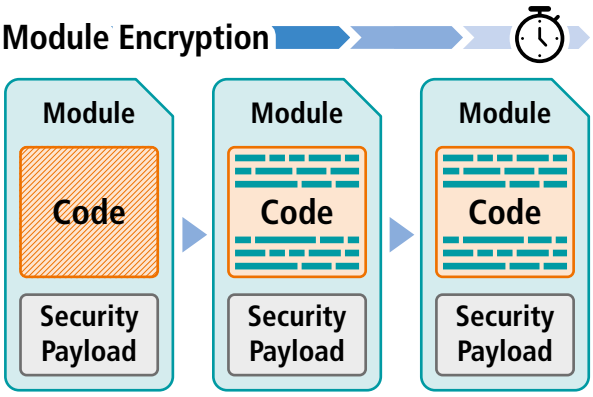


Figure 2: Module Encryption

Encrypting Individual Functions

Encrypting the entire module protects the code at rest. AxProtector also provides a simple way to protect individual functions so that they are decrypted only when first used (***"on demand"***). Previously, this method required source code changes using WUPI (Wibu Universal Protection Interface). AxProtector uses symbol information to integrate protection into the already compiled binary. As with .NET and Python, this can be configured with a few clicks in AxProtector GUI NC or configured individually in WibuCpsConf. Different licenses can be assigned to functions, as with the established IxProtector technology. When base licensing is used, eligible functions are automatically protected with on-demand encryption by default. The entire program is also secured through Module Encryption. Figure 3 shows how the state of the protected program changes over time.

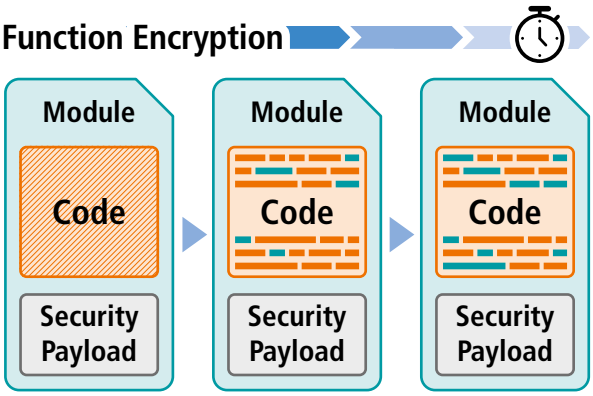


Figure 3: Function Encryption

Before loading, the program is secured by Module Encryption. Once it has been loaded into memory and decrypted, only individual functions become available. Functions that have not yet been executed remain protected (orange) thanks to on-demand decryption. Functions that have already been executed are decrypted (turquoise) and present as plaintext. Even when decrypted, certain functions contain validation logic that prevents unauthorized execution if the required license is missing. For functions called particularly frequently during program execution, automatic optimization reduces the number of license checks. This improves runtime performance while maintaining a good balance between security and performance.

With automatic decryption, more and more functions are decrypted as program execution progresses, leaving them in memory as plaintext. In complex programs, users rarely use 10-20% of the functionality. Even automatic testing typically

reaches 50-90% of the code. A memory dump therefore does not automatically give an attacker access to every function in a program. Dedicated traps can also be added to counter automated attacks aimed at fully decrypting all functions at the binary level.

This provides two layers of protection that build on each other: Module Encryption protects code and data sections before the program is executed. Function Encryption decrypts individual functions only when they are first used.

Temporary Decryption for Stronger Protection

To make the execution of protected programs even more secure, AxProtector incorporates a new technique for automatically re-encrypting functions. The corresponding code is present in memory as plaintext only while the function is executing and is re-encrypted immediately when the function exits. This minimizes the time a function remains in plaintext, leaving it unprotected only while it is actually in use. Automatic Function Re-encryption can be enabled selectively for individual functions, although the added security comes at the cost of reduced performance for those functions. This is particularly noticeable when a protected function is executed in a loop. To improve performance, optional caching can be enabled so that re-encryption takes place after a few seconds rather than immediately. Repeated calls within this window therefore incur significantly less overhead. Nested and concurrent calls are accounted for in both cases. Automatic Function Re-encryption is particularly suitable for functions containing valuable intellectual property, such as proprietary algorithms, calculation methods, or security-critical validation logic. For performance-critical functions called very frequently, caching can be applied selectively, or automatic re-encryption can be disabled.

Figure 4 shows the effects of Automatic Function Re-encryption on the program. The transition from State 2 to State 3 is particularly revealing. A comparison with Figure 3 shows that previously decrypted functions are now encrypted again. Only one function remains in plaintext in both states, because it calls the other functions in sequence. An attacker therefore needs to know when specific functions are exposed in memory, making the attack even more difficult.

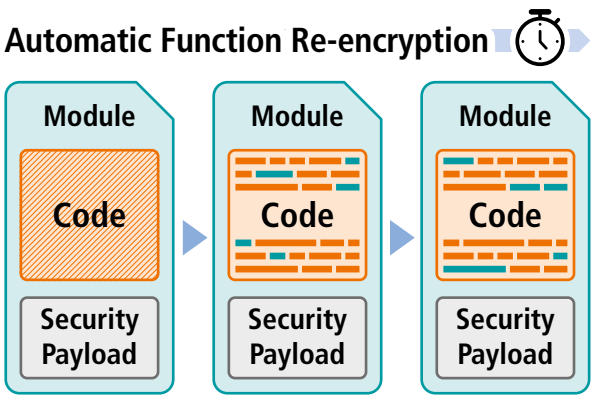


Figure 4: Automatic Function Re-encryption

WUPI

With automatic function encryption, if the license required to call a protected function is missing, CPSRT displays an appropriate message by default. General error behavior can be customized through CpsEvents. However, programmatic control cannot be returned to the original program in this case. The additional licensing error was not anticipated in the original program, and automatic function encryption cannot add a corresponding return path to the source code. WUPI remains available for greater control and dedicated error handling. It allows licenses to be checked before a protected function is called and the corresponding error scenarios to be handled at the source code level. This enables the program to respond specifically to licensing errors. Functions can also be decrypted or re-encrypted at the desired times using the appropriate commands.

Conclusion

AxProtector Windows and AxProtector Linux with the new NC technology build on the proven IxProtector technology and make it particularly easy to use. Protection is integrated into the compiled binary using existing symbol information, without requiring source code changes. Module Encryption protects the entire program at rest, while Function Encryption and Automatic Function Re-encryption provide targeted protection for particularly sensitive functions in memory as well. Selective configuration and optional caching help balance security and performance.

Function Protection at a Glance

- NC is the new technology generation for the proven AxProtector Windows and AxProtector Linux
- Module Encryption provides an outer layer of protection by fully encrypting eligible code and data sections
- IxProtector technology is integrated directly into AxProtector Windows and AxProtector Linux
- AxProtector Windows and AxProtector Linux with NC technology require symbols (PDB/DWARF) for function encryption
- No source code changes are required to encrypt functions
- Automatic Function Re-encryption provides temporary code decryption, with optional caching
- Explicit decryption and re-encryption, as well as error handling, are available through WUPI
- CPSRT serves as an external security runtime, allowing maintenance without re-encryption



Still Shipping Parts, or Printing Them On Demand?

Additive manufacturing (AM) has become an established part of industrial production. According to the Wohlers Report 2026, global AM revenue reached $24.2 billion in 2025, an increase of nearly 11% over the previous year. For the first time, production parts account for the largest share of additively manufactured components. What was once a niche technology for prototyping has become an established part of industrial production, and the question of how to protect and monetize the underlying 3D printing data is now moving to the center of business strategy.

Two Emerging Growth Drivers: Robotics and Defense

Alongside established sectors such as automotive and aerospace, two markets are currently driving particularly strong growth.

Robotics: Additively manufactured components are increasingly becoming part of robots themselves. Customized grippers, lightweight structures for robotic arms, and housings with integrated functions are produced directly from 3D printing data and subsequently processed in automated cells. This places the digital component model at the heart of the connected production line, where it moved through design, integration, and manufacturing systems. As a result, this digital asset deserves the same level of protection as conventional CAD data. Without control over access and production quantities, a distributed print file can be used to manufacture unauthorized copies at any point in the chain.

Defense: The trend is even more pronounced in the defense sector. The U.S. Department of Defense has nearly doubled its additive manufacturing budget for 2026, from approximately $1.8 billion to $3.3 billion. Analysts currently value the global additive manufacturing in defense market at $5 billion, with a projected growth rate of more than 17% through 2035. European and international defense programs are also increasingly adopting 3D-printed spare parts, drone components, and manufacturing directly in operational areas, including mobile printing centers that produce parts in the field. The reason is clear: On-demand production that is independent of long, vulnerable supply chains offers armed forces a strategic advantage.

Yet this decentralization also creates the greatest risk. As more partners, service providers, and field units require access to digital blueprints, the exposure to product piracy, espionage, and unauthorized manufacturing increases. For safety-critical or classified components, the consequences can be particularly serious.

A Complex Process, a Proven Solution

The fundamental challenge remains unchanged: At the begin-

ning of the process chain is a digital design that represents the manufacturer's intellectual property. Today, that design may pass through integrators, local 3D service providers, or even mobile field systems before it is actually printed. Control over its use and the number of parts produced must be maintained at every stage.

Wibu-Systems has been reliably addressing this challenge with CodeMeter technology for years. Digital blueprints are cryptographically encrypted, while the keys are securely distributed as licenses, available worldwide around the clock. Distribution is also possible offline through file exchange, which is essential for isolated defense environments or autonomous field units. A preparation license permits the processing of print data, while a separate, order-specific print license limits production quantities through a tamper-resistant Unit Counter. This protects intellectual property at every stage, from the designer and integrator to the printer on site, regardless of how many stages are involved.

Ready-to-use CodeMeter libraries are available for manufacturers of preparation software and 3D printers across numerous platforms and programming languages. Secure license storage options range from cloud licenses (CmCloudContainer) and software-based activations (CmActLicense) to hardware dongles (CmDongle) where the highest level of protection is required, such as in defense environments. Centralized license management and rollout are handled by CodeMeter License Central, which integrates seamlessly with existing back-office systems.

From Pilot Project to Production Blueprint

What began several years ago as a pilot project with an automotive manufacturer is now an established production application: Daimler Buses uses CodeMeter to protect and monetize 3D-printed spare parts. This model, a protected digital object with transparent control over production quantities, can be applied directly to the requirements of emerging growth sectors, from robotic production lines to highly sensitive components for defense and security. As additive manufacturing becomes increasingly integrated into critical infrastructure and supply chains, transparent licensing and protection technologies such as CodeMeter become all the more important.

KEYFLASH

01010100100
01001010101
10100100010
10010101001
01010010101

Subscribe to our KEYflash and stay up to date with all our latest news, from innovative product features to virtual and on-site events, and inspiring success stories and partnerships.

wibu.com/newsletter

Subscribe to our KEYflash

01010100101
11101010100
10100101010
10010101001
10111010101

NEWSLETTER



India's Industrial Ambitions Meet Software Expertise

India's investment in advanced manufacturing creates opportunities for the software that enables production. Wibu-Systems' October 15 launch event connected that opportunity with a practical question: how can businesses protect industrial software and turn its capabilities into lasting commercial value?

Industry Partners Meet in Bengaluru

The October 15 inauguration of WIBU-SYSTEMS India Pvt. Ltd., organized with VDMA India at The Park in Bengaluru, brought together CEO and co-founder Oliver Winzenried, Dr. Ashok-Kumar Chandra-Sekaran, Managing Director of WIBU-SYSTEMS India Pvt. Ltd., and fellow board member Elke Winzenried, customers, and technology partners. Discussions and a live demonstration explored how software vendors and intelligent device manufacturers can protect their intellectual property and build licensing models around it.

Rajesh Nath, Managing Director of VDMA India, highlighted the growing value of software IP as Indian innovations become commercial products. Dr. Christoph Dorschfeldt, Consul General-Designate of Germany in Bengaluru, addressed Indo-German trade and German investment in Karnataka. Representatives of Knorr-Bremse, a global leader in braking systems for rail and commercial vehicles, and Infineon Technologies, Wibu-Systems' long-standing partner in hardware-based security for CodeMeter, also participated.

An Industrial Opportunity Built on Software

The scale of the relationship is substantial. SAS Partners' August 2026 report, Indo-German Economic Partnership: Trade, Investment and the Road Ahead, puts German exports of goods and services to India at approximately $24.8 billion in 2025 and bilateral foreign direct investment stock at $30.4 billion in 2024. The report identifies advanced manufacturing, semiconductors, industrial software, and technology partnerships among the opportunities for deeper cooperation.

MVTec Software offered a concrete perspective in a video message from Dr. Olaf Munkelt, Managing Director and co-founder. India's ambition to increase domestic value creation, including in semiconductor manufacturing, creates demand for automation and machine vision to improve yields and quality. Its software expertise is a strong fit for these needs. Munkelt also emphasized the reliability and long-term availability required of industrial software and its protection components. MVTec has utilized Wibu-Systems' tools since 2014 and is ready to address Indian customer requirements together.



The negotiated EU–India Free Trade Agreement offers a prospective framework for closer cooperation through provisions on intellectual property, digital trade, and smaller businesses, while companies remain responsible for standards, certification, and protecting their technology.

Local Implementation Expertise

Wibu-Systems' regional operations are based in Bengaluru for South India and Pune for West India. Hyderabad is the registered-office location, where CA Ravi Ladia supports day-to-day operations and compliance. The local ecosystem also includes IntrinsigTech Solutions LLP for embedded hardware, firmware, and system integration, complementing the technology partnership with Infineon Technologies and industry cooperation with VDMA India.

For software and device businesses, implementation involves deciding which functionality to license, how customers will access it, and how to protect the underlying code. CodeMeter combines protection against unauthorized copying, tampering, and reverse engineering with control over licensed features and support for subscription and usage-based models. The Indian team works with Wibu-Systems' specialists in Germany to address these product and business requirements.



Three Awards, One Mission: Protecting Digital Innovation

Wibu-Systems has won Cyber Defense Magazine's Top InfoSec Innovator Award 2026 in **Embedded Security**. CodeMeter protects the software-defined value embedded in devices, from proprietary algorithms to AI models. Its interoperable architecture combines software protection, integrity verification, and controlled access across microcontrollers, industrial systems, and connected platforms. With hardware-based, device-bound, and cloud-based license containers, it can support a wide range of deployment scenarios, including offline and air-gapped environments.

CodeMeter ASIC has won a KES Innovation Award 2026 in **Electronic Components & Materials**. Wibu-Systems is the only German company among 35 winners. Soldered onto a circuit board, the 5 × 5 mm component anchors software protection, licenses, cryptographic keys, and device identities in hardware. USB and SPI interfaces enable integration into industrial electronics, while protected memory stores thousands of licenses. Manufacturers can activate software features on a common hardware platform, linking embedded trust with digital upgrades.

CodeMeter has won in the **IT category** of Hankook Ilbo's 21st Digital Innovation Awards 2026. CodeMeter Protection Suite and its AxProtector technologies help secure proprietary code from compilation through execution. Secure licensing connects this protection with subscriptions, feature-based access, and token-based business models. A modular architecture helps vendors add license management and services as their needs evolve, allowing software publishers and device manufacturers to turn protected innovation into scalable digital business.

Join Wibu-Systems at the following events:

DSEI
9-12 March 2027
Hanover, Germany

Embedded World
16-18 March 2027
Nuremberg, Germany

Hannover Messe
5-8 April 2027
Hanover, Germany

INNO DAYS
22-23 June 2027
Karlsruhe, Germany

Wibu-Systems' INNO DAYS 2027

The premier event for digital strategists aiming to steer their business with a clear, future-focused vision. Attendees will benefit from high-level networking with industry peers, insights from leading experts, and exclusive entertainment. Wibu-Systems' key staff, along with solution partners, clients, and renowned speakers, will lead the discussions. While IP protection and monetization technologies remain central, the event will foster broader engagement on critical industry trends.
Register asap! The future begins now.

www.wibu.com/
inno-days.html

Wibu-Systems' Masterclasses

Our masterclasses are designed to share the knowledge and experience behind our technologies and mission, empowering cutting-edge protection techniques, secure software licensing best practices, and versatile monetization models worldwide. Each session combines practical insights from our CodeMeter ecosystem with our commitment to innovation, reliability, and digital trust. Stay tuned for upcoming announcements on our website or through our newsletter, and be ready to register for the sessions that best match your interests.

www.wibu.com/
webinars.html

Get in touch with our local representatives

WIBU-SYSTEMS USA, Inc.
USA: +1 800 6 Go Wibu
+1 425 775 6900
sales@wibu.us

WIBU-SYSTEMS K.K.
Japan
+81 45 565 9710
info@wibu.jp

WIBU-SYSTEMS (Shanghai) Co., Ltd.
Shanghai: +86 21 5566 1791
Beijing: +86 10 8296 1560
info@wibu.com.cn

WIBU-SYSTEMS Korea Ltd.
Republic of Korea
+82 2 6206 9490
sales@wibu.co.kr

WIBU-SYSTEMS India Pvt.Ltd.
India
+91 9632423407
info@wibu.in

WIBU-SYSTEMS BV
The Netherlands: +31 74 750 14 95
Belgium: +32 2 808 6739
sales@wibu.systems

WIBU-SYSTEMS LTD
United Kingdom | Ireland
+44 20 314 747 27
sales@wibu.systems

WIBU-SYSTEMS SARL
France
+33 1 86 26 61 29
sales@wibu.systems

WIBU-SYSTEMS
Spain | Portugal +34 91 123 0762
Scandinavia | Baltics +46 8 5250 7048
sales@wibu.systems

WIBU SYSTEMS

SECURITY
LICENSING
PERFECTION IN PROTECTION

www.wibu.com

Imprint
KEYnote 52 Issue
Fall/Winter 2026

Publisher
WIBU-SYSTEMS AG
Zimmerstrasse 5
76137 Karlsruhe, Germany
Tel. +49 721 93172-0
info@wibu.com
www.wibu.com

Responsible for the content
Oliver Winzenried

Editors
Stefan Bamberg
Axel Engelmann
Ruediger Kuegler
Daniela Previtali
Oliver Winzenried

Design
Studio Eugen Olchin
Stuttgart, Germany

Print
PASSAVIA Druckservice GmbH & Co. KG, Passau, Germany

Wibu-Systems expressly reserves the right to change its programs or this documentation without prior notice.

Blurry Box®, CmReady®, CodeMeter®, SmartBind®, SmartShelter®, and Wibu-Systems® are registered trademarks of WIBU-SYSTEMS AG. All other brand names and product names used in this documentation are trade names, service marks, trademarks, or registered trademarks of their respective owners.

Copyright ©2026 Wibu-Systems. All rights reserved.

Picture credits:
Cover: ChatGPT
Page 3: SICK
Page 4: ChatGPT
Page 7: ChatGPT
Page 11: ChatGPT
Page 14: ChatGPT
Page 17: ChatGPT
Page 20: ChatGPT
Page 22: ChatGPT
& Wibu-Systems
Page 23: Wibu-Systems

All remaining images are copyrighted by their owner.

INFORMATION

5060-002-02 / 2026-11-12