Microsoft

PlayReady

Protecting Premium Live TV Services with PlayReady

April 2015

Abstract

Microsoft® PlayReady® is the premier platform for the protection and distribution of digital content. This white paper discusses PlayReady features that provide highly scalable, studio-grade protection of Live TV services across a variety of business models.

Microsoft

Table of Contents

Introduction

In recent years the media industry has seen significant shifts in consumer viewing habits and expectations of Live TV and other premium video content. Instead of watching premium content according to broadcast schedules and on a limited set of devices, consumers prefer and expect to watch it at their convenience and on a multitude of consumer electronic devices — phones, tablets, game consoles, Smart TVs, and more. Media consumption is shifting to an "anytime, anywhere, any device" ethos that marks a major change from the past. Consequently, well-understood business models for creating, distributing, and monetizing video content are experiencing considerable upheaval. In addition, piracy continues to be a costly risk, yet access points are proliferating and media formats and protection systems can no longer be tied to a single consumption platform.

Microsoft PlayReady provides highly scalable, studio-grade protection features that help content owners and providers address these trends and implement end-to-end protection of Live TV services across a variety of business models. As the world's most deployed digital rights management (DRM) technology, PlayReady also provides tremendous ecosystem support. Partners includes device manufacturers, media encoders and packagers, system integrators, and application and video service providers. Customers include top-tier cable operators, IP TV providers, network operators, and content owners. In addition, PlayReady is approved and adopted by major Hollywood studios, the Digital Entertainment Content Ecosystem, UltraViolet™, and HbbTV®.

In this white paper, we'll summarize industry trends that informed the design of PlayReady features for Live TV services. We'll also discuss the design principles and architecture of those features and how they can be used to protect and deliver premium Live TV services.

Industry Trends

We're in the midst of a major historical shift from watching scheduled TV programming to watching Live TV and video on a plethora of devices. Content formats and protection systems are no longer tied to a single consumption platform, and there is a growing trend to bring managed content experiences to unmanaged devices. The devices are now internet-enabled and mobile, and the number of devices continues to grow. In addition, the availability of multiple consumption outlets blurs the distinction between web and broadcast media as users consume content on phones, tablets, game consoles, Smart TVs, IP streaming boxes, and connected Blu-ray™ players. Web and broadcast media are becoming relatively indistinguishable to users and high-value movie content is often available via over-the-top (OTT) services the same day as theaters.

Across the industry, we're experiencing four major trends:

- The number of consumer media devices continues to grow.
- Consumers are spending more time consuming content on these devices.
- Content providers are delivering higher quality and higher resolution content.

- Tactics to protect and monetize content are more critical now than the past.

The proliferation of devices and consumption outlets combined with technological advances in areas such as network connectivity and media formats have incented users to spend more time consuming content.

Overall, consumers are spending more time watching video from both a Live TV and a VoD perspective, and they prefer and expect to watch video at their convenience and on different devices. The TV in the living room no longer defines how, when, and where people watch TV.

Consumers also expect to have a broad range of high-quality, high-resolution video to choose from. Consequently, content is a huge investment for content owners and providers.

However, a great deal of that investment is often lost to piracy. During the 2014 FIFA World Cup games, experts estimate that more than 3,700 streams were pirated, resulting in approximately 10.6 million illegal views. The estimated loss in revenue for European and North American broadcasters is in excess of $120 million.

As content becomes more expensive to produce and more of that investment is lost to piracy, the question becomes how to protect and monetize premium content while also meeting consumer demand to watch high-quality content anytime, anywhere, and on any device.

Live TV providers have responded by implementing multiscreen services and offering premium content as part of those services. However, to mitigate piracy costs and create monetization opportunities, those capabilities need to be coupled with robust protection mechanisms. In 2013 for example, Live TV streams — TV production broadcasts on broadcast schedules — comprised approximately 60 percent of all online video consumed worldwide. Of that, only 10 percent was protected by a content protection system versus no protection at all or only basic encryption. With a robust content protection system, providers can add services for previously recorded content, draw larger audiences by providing that content, and assign rights and fees for those services. They can also mitigate piracy costs by protecting live broadcasts.

Working closely with partners and customers across the industry, Microsoft has and continues to ensure that PlayReady is a versatile protection system that addresses current challenges and trends in distribution, operation, and business models, and anticipates future models as the industry evolves. PlayReady supports:

- Multiple media distribution models, including subscription, VoD, rental, ad-based, and purchase (download to own).
- Multiple media delivery options, including live and on-demand streaming, and basic and progressive download.
- Emerging and established international and industry standards, including MPEG-DASH, HTML5 Media Extensions, Smooth Streaming, and Apple HTTP Live Streaming (HLS).

- A broad range of consumer electronic devices, including phones, laptops, tablets, STBs, Smart TVs, and connected Blu-ray players.
- All major client platforms, including Android, iOS, Windows®, Windows Phone®, and Xbox®.

PlayReady is also approved and adopted by major Hollywood studios, the Digital Entertainment Content Ecosystem, UltraViolet™, Smart TV Alliance, and HbbTV®.

For Live TV specifically, Microsoft added and enhanced several PlayReady features and technologies to enable highly scalable solutions for protecting Live TV services. With PlayReady, networks can provide OTT services for their channels while retaining the principles and policy requirements of a managed network. They can also integrate PlayReady technologies with the larger ecosystem, including third-party encoders and packagers, key management services, and advertising systems. In addition, PlayReady addresses the unique challenges of managing content keys and policies within Live TV workflows. Overall, PlayReady provides a comprehensive set of highly scalable, studio-grade features and capabilities that support efficient workflows for protecting and delivering premium Live TV services to all platforms and types of devices.

Understanding Live TV

In the context of content protection and compared to other video distribution scenarios, Live TV presents unique challenges for managing digital encryption keys (*content keys*), policies, and licenses across a distribution system. Unlike other scenarios, both content keys and policies can change at any point in a media stream. In addition, content keys need to be rotated frequently and policies need to be changed quickly for more robust security and contractual requirements. To ensure end-to-end protection, client devices and applications need to support and enforce changes to both content keys and policies in real time.

These requirements alone create two primary challenges:

- A large number of licenses need to be generated and issued continuously to match frequent and continuous changes to content keys and policies.
- Computation time and complexity for licensing and decryption tasks increases dramatically.

If not addressed, both challenges cause significant scalability and performance issues for servers and clients. On license servers, the computation time and resources required to generate and issue licenses increases substantially. On both servers and clients, network traffic for license acquisition processes increases dramatically, which also degrades available network bandwidth for client playback. On clients, storage requirements for licenses becomes inordinately large as the number of locally stored licenses increases. In addition, the complexity and resources required to decrypt licenses and content increases significantly, an issue that is compounded if local resources are taxed by license storage.

In addition to these challenges, Live TV services must address requirements and considerations that are true of any online video distribution system. In the next section, we'll discuss challenges, requirements, and considerations for Live TV scenarios and how PlayReady can be used to address them.

How PlayReady Supports Live TV

To address industry trends and support Live TV scenarios specifically, Microsoft added and extended several PlayReady features to provide a highly scalable solution for Live TV services. When designing those features, a primary goal was to enable networks to provide OTT services for their channels while retaining the principles and requirements of a managed network. Another goal was to support integration with the wider ecosystem and existing infrastructure, including encoders and packagers, key management services, and advertising systems.

Additional design goals centered on addressing the specific challenges of managing content keys and policies in Live TV distribution workflows. The following table describes those challenges and how PlayReady technologies can be used to address them.

Challenge	PlayReady Solution
Policies aren't necessarily driven by content.	Unlike other scenarios where policies might be defined by factors such as studio requirements for a specific media asset, policies for Live TV scenarios vary based on a multitude of factors — for example, broadcast time, delivery region, channel, distribution method, and the content itself. To address this challenge, PlayReady provides a *scalable chained license* scheme that enables providers to create policies based on a combination of factors that they define.
Policies are affected by client region.	Live TV policies are frequently dictated by region-based factors, including but not limited to the geographical area where content is consumed. To address this challenge, PlayReady allows providers to create custom definitions of regions and related policies. For example, a provider might define a region as a geographical area or a specific group of users. To implement their concept of "region" and specify region-based policies, providers can use scalable chained licenses and define a full range of policies.

Challenge	PlayReady Solution
Content keys and policies can change at any point in a stream.	Providers must be able to change content keys and policies frequently and quickly. For example, a regional policy change might occur only a few minutes before a sporting event starts, if the event broadcast must be "blacked out" in a specific region. To address this challenge, PlayReady can embed content keys and policies directly in media streams. This means that clients can decrypt content and enforce policy changes in real time, without adverse impact on the service or playback.
Content keys need to be synchronized across multiple origin servers.	To avoid service disruptions in previous solutions, content keys needed to be synchronized across multiple servers, which created significant scalability and performance issues. To address this challenge, PlayReady can embed licenses directly in the media stream for each video segment. This means that a client can switch between streams and content keys without disrupting playback, and providers have greater flexibility in determining how and when to synchronize content keys.

These solutions represent only a subset of PlayReady features and capabilities for Live TV scenarios. Overall, PlayReady technologies offer a comprehensive set of highly scalable features and capabilities that support a variety of business and distribution models. In the next section, we'll provide a brief overview of the PlayReady product suite and we'll discuss PlayReady features and capabilities that apply most directly to Live TV scenarios.

The PlayReady Product Suite

To enable end-to-end content protection, the PlayReady product suite includes component technologies for both servers and clients. It also includes software development kits (SDKs) and a device porting kit for implementing those technologies on any platform or type of device.

The following diagram identifies the primary components of the PlayReady product suite.

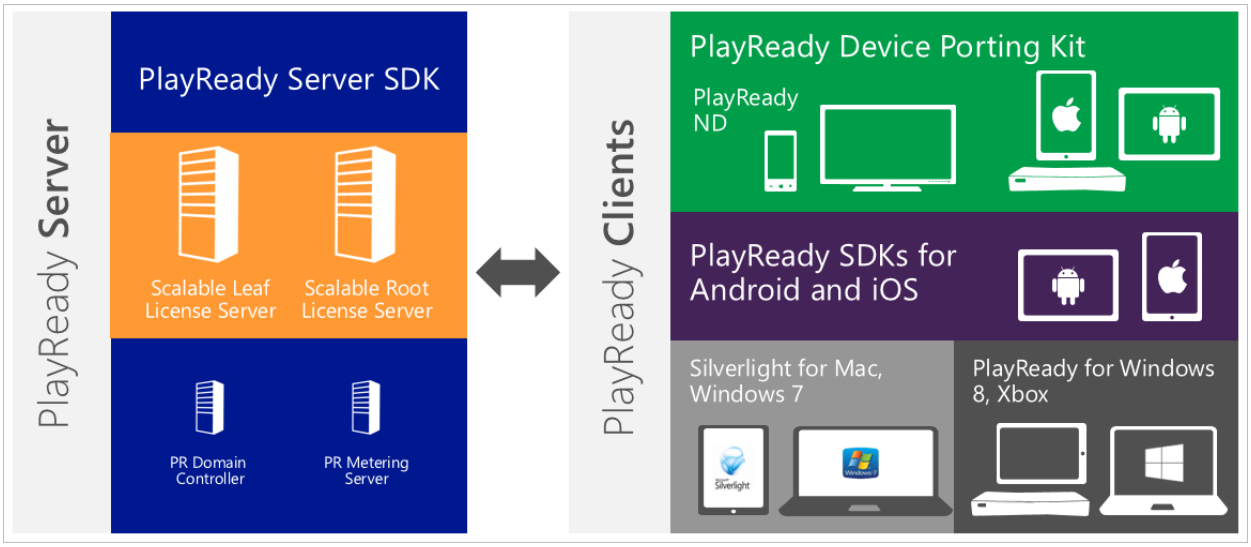

Figure 1 – The PlayReady Product Suite

PlayReady server technologies help prepare and distribute content, issue licenses to PlayReady clients, and optionally manage domains and meter content usage. They can be configured and deployed to one or more servers and hosted on premises or in the cloud. For Live TV scenarios, PlayReady server technologies are configured, deployed, and managed by using the PlayReady Server SDK.

Of the server technologies, PlayReady license services are integral to content protection. They handle critical tasks such as receiving and responding to authentication and license requests from PlayReady clients, building and issuing licenses to valid clients, managing usage rules and policies that are applied to licenses, and implementing business logic for authorization and control of usage rights.

License services can be integrated with other components and services in a provider's infrastructure such as encoders and packagers and key management services. For Live TV specifically, license services need to be deployed as two components, a scalable root license service and a scalable leaf license service, which enables implementation of a scalable chained license scheme that we'll discuss in the next section. Root and leaf license services can be hosted on the same or different servers.

A PlayReady client is a device or component — for example, a STB, app, media application, or browser plug-in — that can use PlayReady technologies to acquire and interpret licenses, decrypt and play protected content, and enforce the policies defined by a provider. To perform those tasks, PlayReady clients use a combination of the APIs provided by the host platform and PlayReady APIs, which add layers of content protection to that framework.

PlayReady provides a multitude of client options spanning all major platforms and virtually any type of device. Those options derive from native support on some platforms, the latest web

standards and technologies, PlayReady SDKs that are optimized for specific platforms, and a device porting kit that can be used to implement PlayReady functionality on any device, regardless of platform. The porting kit is typically used for devices such as STBs, game consoles, Smart TVs, kiosks, and mobile devices. Note that all current Microsoft platforms — Windows, Windows Phone, and Xbox — provide native support for PlayReady, including features for Live TV. For other platforms, the latest client SDKs and device porting kit enable developers to implement support for Live TV features.

For more information about PlayReady server and client technologies, see Deploying PlayReady Technologies and other resources on the PlayReady website.

Key Features

PlayReady provides a comprehensive set of studio-grade features and capabilities that support highly scalable, efficient workflows for protecting and delivering premium Live TV services to all platforms and types of devices. These features also enable optimal user experiences across a wide range of distribution and consumption options. For Live TV specifically, PlayReady offers features and capabilities that address critical areas spanning key and policy rotation, access control, multiscreen capabilities, and ad insertion.

Key and Policy Rotation

With PlayReady, providers can rotate content keys and policies frequently and quickly, and they can do so dynamically or automatically according to schedules that they define. In a typical deployment, rotations are implemented by integrating PlayReady license services with a provider's key management service and content encoders and packagers.

Note that rotations aren't limited to content keys. Each rotation can include changes to the full range of policy settings — for example, switching from protected to unprotected ("in the clear") content for a specific broadcast such as an infomercial, applying region-based restrictions such as a blackout for a sporting event, and changing output-protection levels such as limiting playback options for a premium movie. PlayReady offers this high degree of facility primarily by providing a *scalable chained license* model and by embedding and delivering licenses in stream.

Scalable chained licenses extend the chained license model used in other distribution scenarios. In the traditional model, a chained license is a series of associated licenses that collectively specify content keys and policies for one or more media files. Each license chain includes a *root license* and one or more *leaf licenses* that are associated with and dependent on the root license:

- Root license – Contains a *root key* that is used to encrypt *leaf keys* in all the leaf licenses that are associated with it. Clients use a root key to decrypt associated leaf licenses and play media streams that are protected by those leaf licenses. A root license also stores policies that apply to all content protected by those leaf licenses.

- Leaf license – Contains the leaf key (content key) for a specific media file and defines any policies that are specific to that file. A leaf license also contains a key identifier for the root key in the root license that it is associated with and inherits policies from.

To link elements in a license chain, each leaf license contains an *uplink identifier*. The value of the uplink identifier is the key identifier of the next parent license in the chain, a root license.

Unlike the traditional model, the scalable chained license model is designed to grant access to multiple services while also minimizing the number of content keys that need to be issued. For this reason, a scalable root license contains multiple uplink keys (root keys), one for each service, instead of only one. A service might be a channel or one of many A/V streams that are part of a channel, such as a sub-channel or component of a virtual channel. In this way a scalable root license enables a client to access the overall service and it conveys policies for a user's account, such as the user's geographical region, channel subscriptions, and time-based policies. Scalable root licenses can be issued to as many clients as a provider allows for each user.

A scalable leaf license contains one content key and is bound to a specific uplink key (root key) in a scalable root license. Scalable leaf licenses are generated by PlayReady license services, sent to a packager, and then embedded and delivered in stream to clients. This design enables a provider to deliver the same streams to every client across the service and delivery doesn't require a secure channel — the streams are already encrypted and protected, and clients can't play them unless they have a root license that has the requisite keys for decrypting those streams. Scalable leaf licenses typically contain limited policy data because leaf keys can change as frequently as every two seconds for a given stream. However, they can be helpful if a provider wants to implement granular output-protection policies — for example, limiting playback for specific channels or shows to an HDMI port with HDCP enabled.

Scalable chained licenses also minimize the number of uplink keys that a provider needs to issue to clients. Instead of issuing all required uplink keys for all subscribed content, a provider can issue only a subset of keys through use of a *master key set* and a key derivation scheme that PlayReady applies programmatically. For complex systems with many channels, use of any other model requires a provider to issue a significant number of licenses to support rotation. With PlayReady, providers deliver licenses more efficiently by limiting the amount of key data sent to clients and they provide scalable delivery of streams by letting clients enforce access policies.

In a typical implementation, a provider first defines a master key set that specifies all the channels and regions for the service. This key set is then used to generate scalable root and leaf licenses and to enforce policies around region restrictions and channel access for clients. Before it streams any content, a client requests a scalable root license that defines the channels and regions associated with the user's account. For this reason, a new scalable root license should be issued to a client automatically on a regular basis, such as every 2-24 hours, to reflect any policy or other changes for the user's account or region. In addition, a client should verify that it has a valid scalable root license each time it starts and, if not, request one. Otherwise, the client cannot stream any content. After a client obtains a valid scalable root license, it has all the data that it needs to access the service.

To deliver scalable leaf licenses in stream, PlayReady uses the ISO Common Encryption Scheme (CENC), as specified in ISO/IEC 23001, and provides built-in support for embedding licenses in MPEG-DASH and Smooth Streaming (PIFF) content. This support can be extended to content in the MPEG-2TS format.

Customizable Access Controls

To support a variety of business models and a wide range of options spanning service-wide to user-specific levels, PlayReady gives providers complete control of how they define and implement access controls for their content and services, including:

- Customizable channel packages – Through scalable chained licenses and other features, providers have the flexibility to define custom channel packages that match their business models and user preferences. For example, a user might subscribe to only a series of channels (package) or a specific channel in addition to a specific package. PlayReady supports a flexible, fully customizable approach to defining channels and packages.

- Customizable subscriptions – Providers also have complete flexibility in defining subscription types and levels for channels and packages, as well as time-based access controls for those subscriptions. For example, a provider can use scalable root licenses and policies to give some users free temporary access to specific channels while also giving paid subscribers complete access to those channels. Similarly, a provider can use scalable chained licenses to provide users with free extended previews for movies that are playing on a specific channel.

- Customizable regions – Providers can define their own concept of regions and apply region-based policies that align with that concept. For example, a provider might define a region as a geographical area or a specific group of users. To implement their concept of "region" and specify region-based policies, providers can use scalable chained licenses and define a full range of policy settings. In addition, those policies can be changed (rotated) with content keys.

- Blackout signaling – PlayReady supports highly responsive blackout services based on geographical regions or other criteria that a provider defines. Blackout services can be implemented through key and policy rotations, which means that blackout signals can be sent and applied quickly.

Overall, PlayReady gives providers complete flexibility and control of how they define access controls and it offers the right mix of features for implementing those controls across a broad range of scenarios.

Multiscreen Capabilities

As with other distribution models, PlayReady technologies are compatible with all major client platforms and virtually any type of device — they work on a variety of device classes, architectures, and system environments. This means that you can implement a PlayReady client on any type of in- or out-of-home device, including mobile phones, tablets, laptops, game

consoles, STBs, Smart TVs, kiosks, and embedded displays. You can also implement PlayReady on network device (ND) transmitters and receivers, which enable users to share media content between devices that are connected to the same IP network. Overall, any device that is capable of receiving and playing media content can host PlayReady client technologies.

In addition, PlayReady technologies offer built-in support for protecting both offline and online video recordings, whether those recordings are stored locally on a device such as a DVR or in the cloud. Once content is protected, it remains protected. Encryption, licensing, and authentication mechanisms ensure that the content can be stored anywhere yet consumed only by valid clients and users according to license policies. Note that the PlayReady authentication model handles authentication at both user and device levels. This means that PlayReady supports subscriber-based authentication models that allow, restrict, or prevent users from consuming media content from multiple access points and devices, and providers can use policies to enforce those restrictions.

Dynamic Ad Insertion

PlayReady enables capabilities for signaling any type of advertising metadata to clients and integrating with external advertising systems. These capabilities align with national and local advertising allocations and decisions and they use industry standards. Consequently, providers can easily integrate PlayReady technologies with their preferred advertising system and they can tailor advertisements based on criterion such as a user's region.

Ad insertion may be performed as an independent presentation that occurs during a main presentation. An ad or series of ads is signaled in a sparse stream (Smooth Streaming) or an event stream (MPEG-DASH) through server- and application-dependent signaling. The signal contains advertising metadata such as when an ad should be played and what advertising system URL to query in order to determine which ad to play. When a PlayReady client receives an advertising signal in a stream, it processes the metadata in the signal, switches to the ad player at the correct point in the presentation, plays the specified ad, and then switches back to the main presentation. In addition to streamlining delivery of both media content and ads, this design enables providers to tailor advertisements to specific users by seamlessly switching to and playing, for example, a local ad instead of a national ad.

PlayReady doesn't require any specific standard to signal an ad. Instead, clients subscribe to ad insertion events by using an API that allows them to retrieve advertising data. In many cases however, ad insertion is signaled by using the ANSI/SCTE-35 standard.

Live TV Streaming Architecture

The following diagram illustrates the high-level architecture for integrating PlayReady technologies with other systems to protect media assets delivered through Live TV services.

In the diagram, note that the overall system uses a provider's existing infrastructure and resources for encoding, packaging, encrypting, hosting, and delivering content. The system uses a provider's preferred third-party service for managing key and policy rotation, advertising data,

and blackout services. For key and license management, the PlayReady license server has two components, a scalable root license service that generates and issues scalable root licenses to clients and a scalable leaf license service that works with an encryptor/packager server to generate scalable leaf licenses that are delivered in stream to clients.

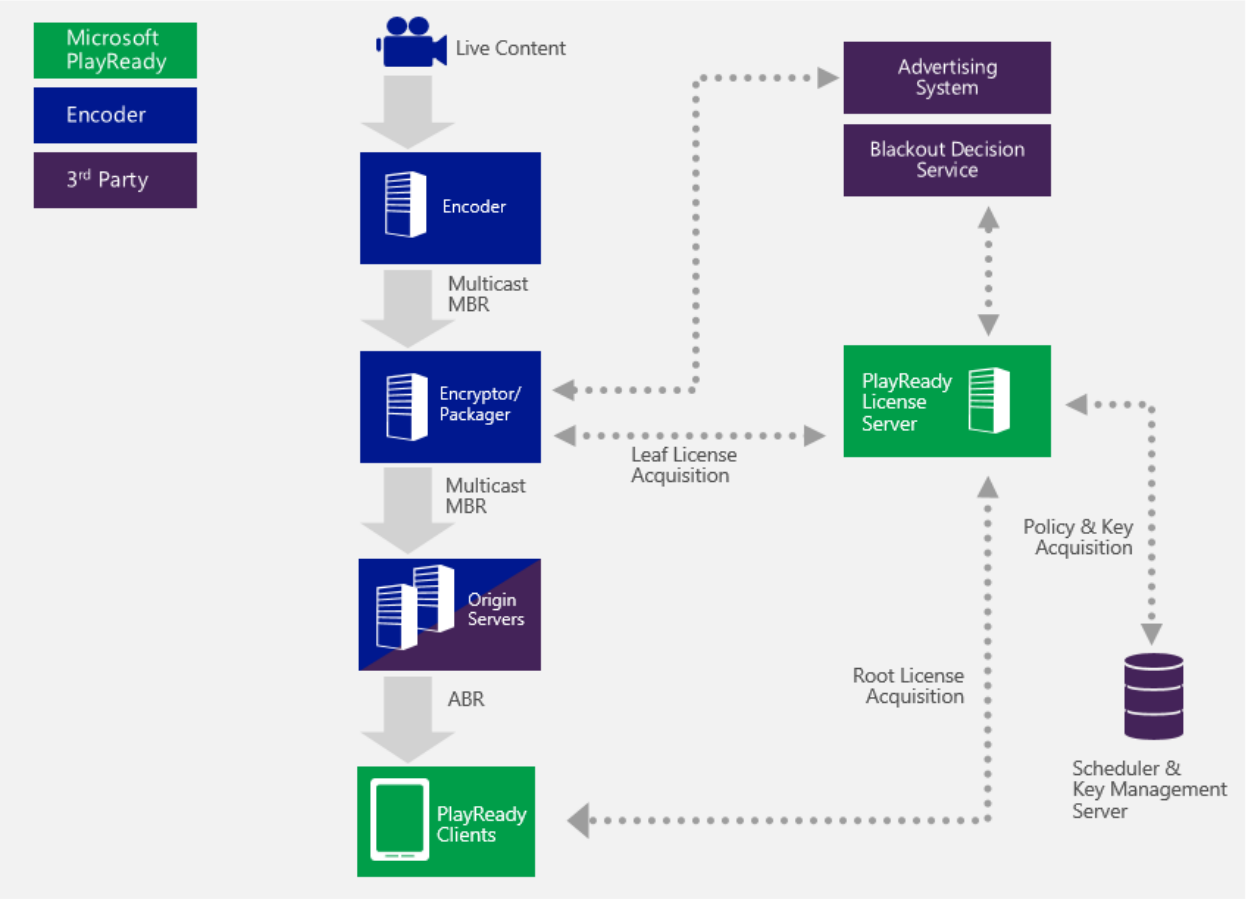

Figure 2 – High-Level Architecture for Protecting Live TV Services with PlayReady

Moving downward and then from right to left in the diagram, the primary purpose and communications between each component is as follows.

Component	Purpose
Encoder	Encodes media assets at multiple bit rates and in a streaming format that the provider chooses. The resulting media files are sent to a packager for encryption and packaging.

Component	Purpose
Encryptor/Packager	Works with the scalable leaf license service to obtain a content key and leaf license for each stream that will be delivered by a set of channels during a specified time period. Each license minimally contains the key identifier (KID), content key, uplink root key identifier, and any policies for a stream. The packager also communicates with the advertising system to retrieve and add any advertising metadata to each stream. After it receives scalable leaf licenses from the license service, the packager embeds the correct license in each stream as Protection System Specific Header Box ("pssh" box) data, as defined by the ISO CENC scheme (ISO/IEC 23001) for PIFF and MPEG-DASH files, and encrypts each stream. For information about the rights management header that is included in pssh box data, see the appendix. For security reasons, the packager and leaf license server communicate via HTTPS and an SSL connection that requires mutual authentication. The protected streams are then published to origin servers as adaptive bit rate (ABR) assets.
Origin Server	Hosts encrypted streams and distributes them to authorized clients.
PlayReady Client	Accesses the system and decrypts and plays content according to the policies specified in scalable root and leaf licenses. A scalable root license authorizes the client to access the overall service and specifies policies for a user's account, such as region, channel subscriptions, and time-based policies. Scalable leaf licenses are delivered in stream and allow the client to decrypt and play specific streams. Each time a client starts, it verifies that it has a valid scalable root license. If it doesn't, it requests a new one from the scalable root license service. After it has a valid scalable root license, the client uses a key in the scalable root license to decrypt and play content in individual streams according to policies.
Advertising System	Stores advertising data and works with the encryptor/packager to add that data to the appropriate streams. It also works with PlayReady clients to ensure that the correct advertisement is played at the correct time when a user views the content in a stream.
Blackout Decision Service	Stores blackout data and works with the scalable leaf license service to ensure that scalable root and leaf licenses reflect blackout decisions.

Component	Purpose
PlayReady License Server – Scalable Root License Service	Working with most other system components, builds scalable root licenses that authorize clients to access the overall service, specify policies for a user's account, and contain content keys and policies that enable clients to decrypt and play specific streams. It also issues scalable root licenses to PlayReady clients in two ways: • Proactively according to a provider-defined schedule – A new root license should be issued to clients automatically on a regular basis, such as every 2-24 hours, to reflect any policy or other changes that might occur for a user's account or region. • Reactively in response to requests from clients – Each time a client starts, it should verify that it has a valid scalable root license. If it doesn't, the client requests a new one from the scalable root license service. Otherwise, the client cannot decrypt and play any protected streams. After a client obtains a valid scalable root license, it has all the data it needs to access the service and play content according to the policies specified in scalable root and leaf licenses. Scalable root licenses can be issued to as many clients as a provider allows for each user.
PlayReady License Server – Scalable Leaf License Service	Builds and sends scalable leaf licenses to the encryptor/packager for each stream that will be delivered by a set of channels during a specified time period. In most deployments, the scalable leaf license service determines what needs to be delivered in each license by using business logic that a provider implements in the service and data that the service receives from the scheduler and key management server. For security reasons, the packager and leaf license server communicate via HTTPS and an SSL connection that requires mutual authentication.
Scheduler & Key Management Server	Stores policy data and works with PlayReady license services to rotate content keys and master keys, which are used to generate master key sets for scalable chained licenses.

Delving into the relationship between the scalable leaf license service and the encryptor/packager, those components communicate primarily by using a SOAP method that defines the packaging data required to encrypt and protect a stream. For detailed information about this SOAP method, see the appendix: AcquirePackagingData Web Method specificationAppendix: AcquirePackagingData Web Method specification.

The encryptor/packager sends a request that includes the following data about each stream:

- Program identifier, as defined by the provider
- Time offset from the program's start
- Time interval within the program, from the program start offset
- Blackout information such as start time offset and duration
- Any custom data that a provider chooses to include

The scalable leaf license service processes the request and responds with the requisite protection data for each stream, including:

- Encryption type
- Key identifier (KID) for the content key
- Content key
- When the next policy rotation will occur, as an offset from the program start
- Amount of time (duration) until the next key rotation
- The scalable leaf license to embed in the media file

Note that the data is driven primarily by offsets instead of specific times. This eliminates requirements to synchronize clocks across servers.

In many deployments, the encryptor/packager requests batches of licenses from the scalable leaf license service at a regular frequency, such as every 1-2 hours, for a specific program identifier. The license service then determines the content key, policy, and other requirements for the licenses based on business logic that a provider implements in the service and data that the service receives from the scheduler and key management server. Ultimately, the license service generates and sends the scalable leaf licenses to the encryptor/packager.

It's also possible to implement business logic directly on the encryptor/packager server and configure the encryptor/packager server to communicate directly with the scheduler and key management server. In this integration model, the encryptor/packager server might use a custom template to request specific protection data from the scalable leaf license service. The license service then generates the correct license without applying any additional business logic. This model can be useful for simpler scenarios such as license requests that are driven by only one or two factors such as key rotation or time-based policies.

Scenarios

The following sections provide visual representations and descriptions of how PlayReady client and server technologies can be used in specific Live TV scenarios — regional blackouts, à la carte channel subscriptions, and localized ad insertion.

Region-Based Blackout

PlayReady can use region sets and region-based keys to enable blackout and other types of group-based restrictions. If a provider knows which regions to implement when building and issuing scalable chained licenses, region sets and region-based keys can be especially helpful for blackouts — blackouts can be implemented by deriving and binding leaf keys for event streams to both the root key associated with region(s) that are not allowed to view the event and the leaf keys for the channel that is broadcasting the event. This means that clients can enforce blackout restrictions in real time. It also means that providers can minimize the number of root and leaf keys that they need to generate and issue.

In this scenario, the provider defined regions as a hierarchy of large geographical areas, time zones, states, and cities. The provider implemented the hierarchy when defining the master key set and scalable chained licenses for its subscriber base and content. In the context of master key sets and scalable chained licenses, the resulting hierarchy (*binary tree*) served as a key derivation scheme that was used to define content keys that were associated with each other through scalable root and leaf licenses.

The following diagram illustrates the key derivation scheme that the provider defined.

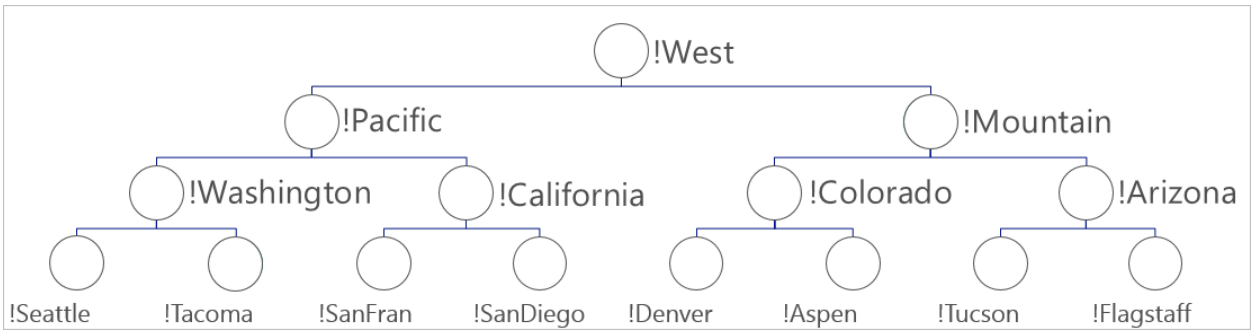

Figure 3 – Region-Based Key Derivation Scheme for Scalable Chained Licenses

The provider then needed to black out a sporting event in the "Seattle" region while ensuring that clients in all other cities in the "West" region could decrypt and play the event streams. To accomplish this goal, the provider applied an exclusionary approach to the key derivation scheme and did the following:

- Well before the event, issued scalable root licenses that contain root keys and policies that reflect user account settings, including region.

- Generated leaf keys for event streams and derived those keys from both the root key for the "Seattle" region and keys for the broadcast channel.
- For clients in all regions except the "Seattle" region, issued scalable root licenses that contained the root key for the "Seattle" region and all other regions within the "West" region. These root keys allowed the clients to derive the correct leaf keys for all streams delivered to all regions within the "West" region.
- For clients in the "Seattle" region, issued scalable root licenses that contained root keys for only the "Tacoma", "California", and "Mountain" regions. These root keys allowed the clients to derive the correct leaf keys for all streams delivered to regions within the "West" region except the "Seattle" region.

The following diagram illustrates the key derivation scheme and how it was used to generate keys and issue licenses for clients in the "Seattle" region. In the diagram, green circles indicate region-based root keys that were issued to clients in the "Seattle" region. The black circle with an "X" indicates the region-based root key that was not issued.

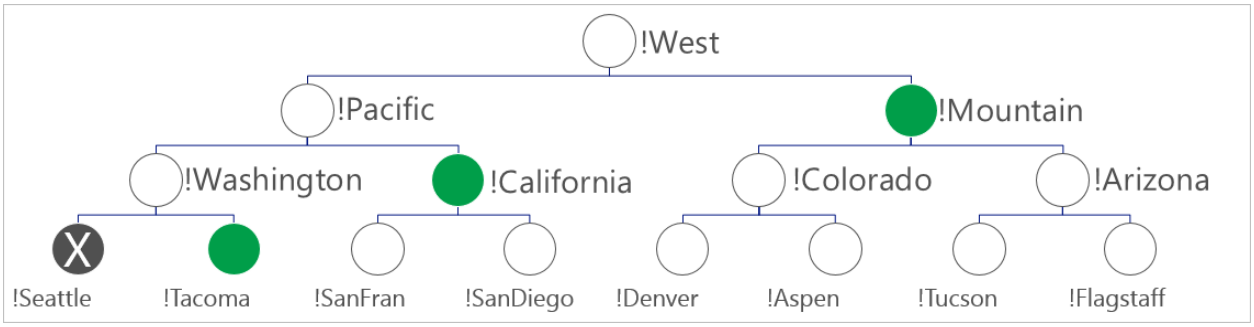

Figure 4 – Key Derivation and License Scheme for Blacking Out an Event in Seattle

In this scheme, clients in the "Seattle" region did not have the root key needed to derive the correct leaf key for decrypting and playing the event streams. Clients in all other regions did.

Note that this design also reduced the number of keys that needed to be generated and issued. With a different design, the provider might have needed to issue eight root keys, one for each city, and generate leaf keys derived from all eight keys. In this design, the provider issued root keys for only three regions — "Tacoma", "California", and "Mountain" — and derived leaf keys from only those three root keys.

À La Carte Channel Subscription

In this scenario, the provider delivered live streams to seven channels within a specific time frame. The provider did not need to define any region sets for the service. The provider supports à la carte channel subscriptions in addition to package-based subscriptions.

To protect the streams, the provider defined a master key set based on the number "7", which matches the number of channels to protect, and used PlayReady APIs to build the key derivation

scheme for generating content keys that are associated with each other through scalable root and leaf licenses. The following diagram illustrates the key derivation scheme that the provider defined.

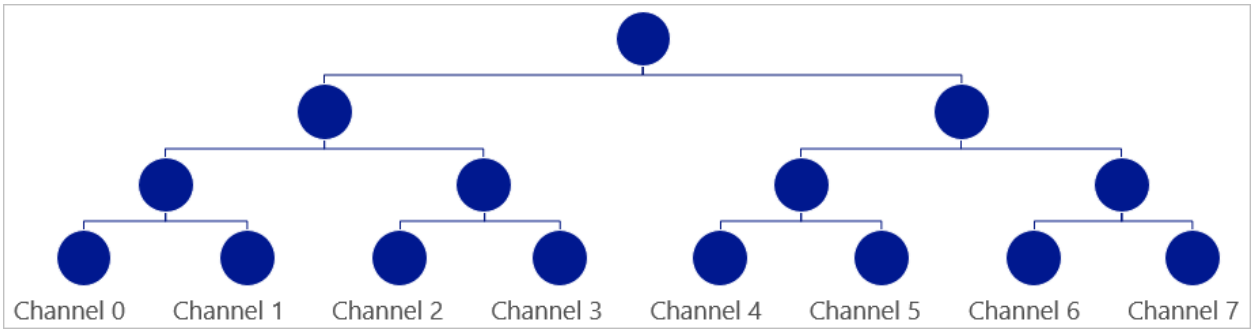

Figure 5 – Channel-Based Key Derivation Scheme

Instead of generating and sending keys and licenses for all seven channels to all users (clients), the provider used this scheme to limit the amount of key data that was sent — scalable root licenses contained only those parts of the master key set that each user needed to access streams for the channels to which they subscribed. The following diagram illustrates the how the derivation scheme was used to support specific combinations of channel subscriptions.

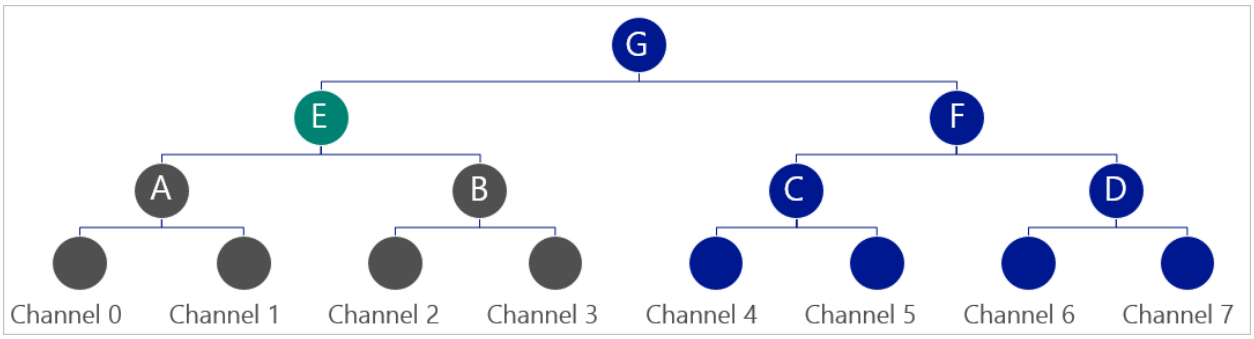

Figure 6 – Channel-Based Key Derivation Scheme for Scalable Chained Licenses

Specific subscription scenarios for the diagram include the following:

- If a user subscribed only to channels 0 through 3, the scalable root license for the user's client contained only key, "E", instead of one key for each channel (four keys).
- If a user subscribed to channels 0 through 3 and channel 6, the scalable root license for the user's client contained key "E" and the key for channel 6, instead of five or more keys.
- If the user subscribed only to channels 0 and 1, the scalable root license for the user's client contained only key "A", instead of one key for each channel (two keys).
- If the user subscribed to all channels, the scalable root license for the user's client contained only key "E" and key "F", instead of one key for each channel (seven keys).

By using this model, the provider significantly reduced the complexity and amount of key data that needed to be generated and issued to clients. Clients received only those keys that were relevant to the user. In addition, the provider was able to offer à la carte subscriptions to users without compromising protection mechanisms. For complex systems with many channels, use of any other model would require a provider to issue a significant number of keys and licenses, which is prohibitively inefficient, especially if the system supports key and policy rotation.

Localized Ad Insertion

In this scenario, ad insertion is performed as an independent presentation that occurs during a main presentation. The client received an ANSI/SCTE-35 advertising signal in a stream, processed the metadata in the signal, switched to the ad player at the specified time, played the specified series of ads for the client's region, and then switched back to the main presentation.

The following diagram illustrates the sequence of steps and communication paths for the scenario.

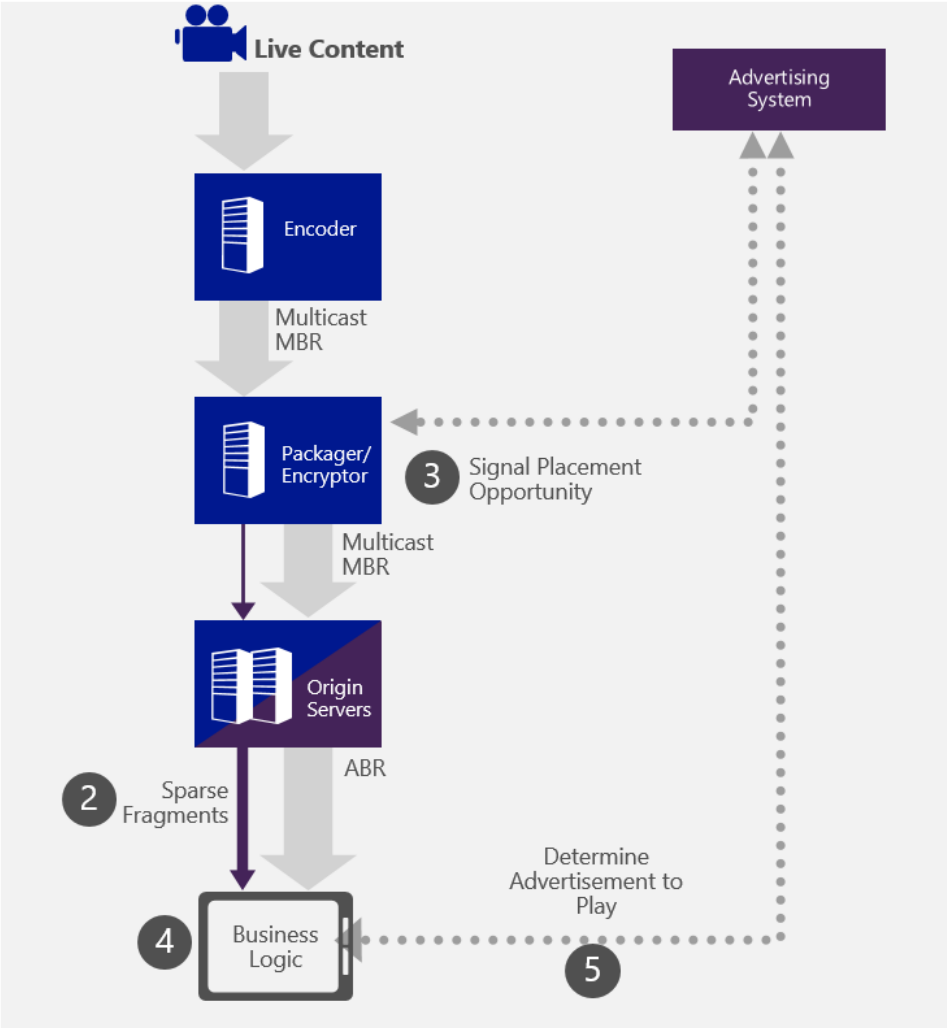

Figure 7 – Sequence of Steps and Communication Paths for Ad Insertion

In the diagram, the primary steps are:

1. A channel on the origin server received an insertion cueing message in an MPEG-2 transport stream. The cueing message conformed to the ANSI/SCTE-25 standard.
2. The origin server parsed the metadata in the cueing message signal and added it to the payload of a sparse fragment that was sent to clients.
3. The origin server sent the same metadata to an advertising system to ensure that the system was aware of the placement opportunity in the stream. (This step is optional in other scenarios.)

4. When the client received the sparse fragment, a business logic component in the client used Smooth Streaming APIs to access the sparse fragment data. Metadata in the sparse fragment indicated when to play the ad and the URL of the advertising system to query when determining which ad to play.

5. The client queried the advertising system at the URL specified in the sparse fragment. The advertising system sent the client the URL for the ad to play. The client then used Smooth Streaming APIs to play the specified local ad instead of a national ad in the stream that the client received from the origin server.

Note that PlayReady doesn't require use of the ANSI/SCTE-35 standard to signal an ad. Instead, clients can subscribe to ad insertion events by using an API that allows them to retrieve advertising data and parse fragments of an ad stream.

Licensing Options

Microsoft offers several PlayReady licenses, depending on how you plan to use and deploy PlayReady technology. The following table lists each license agreement and outlines the applicable scenarios and products that are included in the license package.

Agreement	Scenarios	Includes
Microsoft PlayReady Final Product License	For distributing PlayReady client devices to end users or using PlayReady clients in a commercial deployment.	PlayReady Certificate Generation Kit, PlayReady Client SDKs for iOS and Android, PlayReady Document Pack, PlayReady Windows 8.1 Sample Application with ND, Client SDK SL2000 Library, and Company Device Certificate
Microsoft PlayReady Intermediate Product License	For developing a PlayReady iOS or Android client, or a client device such as an STB, Smart TV, or media player.	PlayReady Device Porting Kit, PlayReady Client SDKs for iOS and Android, PlayReady Document Pack, PlayReady Windows 8.1 Sample Application with ND, CDMi Example Code for PlayReady, Client SDK SL2000 Test Library, Company Device Test Certificate
Microsoft PlayReady Server Deployment License	For using PlayReady server technology in a commercial deployment or end-user distribution.	PlayReady Certificate Generation Kit, PlayReady Document Pack, Deployment Certificate, Premium Deployment Certificate, Domain Certificate, Metering Certificate
Microsoft PlayReady Server	For developing a PlayReady server.	PlayReady Server SDK, PlayReady Documentation Pack, Deployment Test Certificate, Premium Deployment Test

Agreement	Scenarios	Includes
Development License		Certificate, Domain Test Certificate, Metering Test Certificate

For every license, you must also sign the PlayReady Master Agreement.

Note that you do not need a license in the following scenarios:

- Developing a packager/encryptor that supports PlayReady technologies.
- Developing and distributing a PlayReady client for Windows 8 or later, Windows Phone, Xbox, or Silverlight®. Those platforms provide native support for PlayReady technologies. You do, however, need a PlayReady Deployment license to distribute licenses to Windows endpoints.

If you are developing and distributing a PlayReady client for any other platform, you need to purchase the PlayReady license that reflects your business needs.

Instead of licensing PlayReady server technologies directly, you can contract with a Microsoft PlayReady Server ASP licensee to run PlayReady servers on your behalf. Or, if you're interested in deploying a service on behalf of a third-party brand, you can execute the Microsoft PlayReady Server ASP Agreement. For more information about the Microsoft PlayReady Server ASP Agreement, see Approved Microsoft PlayReady Licensees.

For more information about PlayReady licensing, see Licensing Frequently Asked Questions. If you have questions about the PlayReady licensing process, please contact Microsoft at wmla@microsoft.com.

Appendix: Rights Management Header

As mentioned in Live TV Streaming Architecture, a packager embeds the correct leaf license in each stream as Protection System Specific Header Box ("pssh" box) data, as defined by the ISO CENC scheme (ISO/IEC 23001) for PIFF and MPEG-DASH files, and it encrypts each stream. The rights management header (**WRMHeader** element) enables a client to locate or acquire a license for a media file. The **WRMHeader** element is stored in the media file or exposed through an ABR media presentation description, as defined in the PlayReady Header Object specification on the PlayReady website.

PlayReady for Live TV introduced **WRMHeader** version 4.1.0.0, specifically to add support for an optional **DECRYPTORSETUP** element. The **DECRYPTORSETUP** element may contain only one value, "ONDEMAND". If the **DECRYPTORSETUP** element exists in a **DATA** node and its value is set to "ONDEMAND", PlayReady clients should not expect the full license chain for the file to be available for acquisition or present on the client device.

Support and use of the **DECRYPTORSETUP** element addresses Live TV scenarios that deliver scalable root licenses and embedded scalable leaf licenses to clients. The implication is that streams must use **WRMHeader** version 4.1.0.0 and include a **DECRYPTORSETUP** element with a value of "ONDEMAND".

Appendix: AcquirePackagingData Web Method specification

AcquirePackagingData challenge description

The following sample shows an in-stream license delivery protocol challenge using the AcquirePackagingData Web method:

```
POST /playready/rightsmanager.asmx
HTTP/1.1
Host: mycontent-svr
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/DRM/2007/03/protocols/AcquirePackagingData"

<?xml version="1.0"
encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AcquirePackagingData xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols">
      <challenge
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/AcquirePackagingData/v1.0">
        <ProtectionSystems>
          <ProtectionSystemId>string</ProtectionSystemId>
          <ProtectionSystemId>string</ProtectionSystemId>
        </ProtectionSystems>
        <StreamProtectionRequests>
          <StreamInformation>
            <ProgramIdentifier>string</ProgramIdentifier>
            <OffsetFromProgramStart>duration</OffsetFromProgramStart>
            <RequestInterval>duration</RequestInterval>
          </StreamInformation>
          <StreamInformation>
            <ProgramIdentifier>string</ProgramIdentifier>
            <OffsetFromProgramStart>duration</OffsetFromProgramStart>
            <RequestInterval>duration</RequestInterval>
            <BlackoutInformation>
              <BlackoutOffsetFromProgramStart>duration
              </BlackoutOffsetFromProgramStart>
              <BlackoutInterval>duration</BlackoutInterval>
              <BlackoutData>string</BlackoutData>
            </BlackoutInformation>
          </StreamInformation>
        </StreamProtectionRequests>
```

```
        <CustomData>string</CustomData>
      </challenge>
    </AcquirePackagingData>
  </soap:Body>
</soap:Envelope>
```

The following sections describe the challenge fields in the challenge in more details.

```
<AcquirePackagingData xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols">
  <challenge
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/AcquirePackagingData/v1.0">
...
```

Note that the xmlns namespaces need to be put in both elements in order to be a valid challenge

Elements	Requirement	Info
<ProtectionSystems>	Required	Collection of <ProtectionSystemId /> elements. The set of protection systems for which policy is being requested. The business logic must validate each ProtectionSystemId (Guid) against those supported. The server should fail the request if it cannot support one of the protection systems requested and must only return data associated with the requested systems.
<StreamProtectionRequests>	Required	Collection of <StreamInformation /> elements. Information identifying the fragments to be encrypted. The fields in the StreamInformation structure are described below.
<CustomData>	Optional	A service specific string sent from the client to the server. The string data is opaque to the server SDK and is merely passed on to the business logic layer. It is essentially an extensibility mechanism allowing the service to pass its own context-specific data to the server during the request. (string)

<StreamInformation>

The StreamInformation element represents information about a specific period of time in a stream for which the server SDK is asked to issue licenses.

Elements	Requirement	Info
< ProgramIdentifier>	Required	An identifier of the program being encrypted, for example, a GUID or ISAN (96-bit International Standard AudioVisual Number) **(string)**.
<OffsetFromProgramStart>	Required	The time offset from the start of the program up to millisecond granularity. **(XSD Duration)**
<RequestInterval>	Optional	Indicates the time interval within the program from the program start offset for which packaging data is requested. Value can be up to millisecond granularity. **(XSD Duration)**
<BlackoutInformation>	Optional	The blackout information about the fragment to be encrypted. The fields in the `BlackoutInformation` structure are described below.**(BlackoutInformation)**

<BlackoutInformation>

The BlackoutInformation element represents the blackout information of the stream.

Elements	Requirement	Info
<BlackoutOffsetFromProgramStart>	Required	The blackout start time offset from the start of the program up to millisecond granularity. **(XSD Duration)**
<BlackoutInterval>	Optional	Indicates the blackout duration within the program from the blackout start offset for which packaging data is requested. Value can be up to millisecond granularity. **(XSD Duration)**
<BlackoutData>	Optional	The service-specific blackout data sent from client to the server. **(string)**

AcquirePackagingData Response description

The following sample shows an example response to the challenge.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AcquirePackagingDataResponse
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols">
      <AcquirePackagingDataResult
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/AcquirePackagingData/v1.0">
        <StreamProtectionResponses>
          <StreamProtectionInformation>
            <StreamInformation>
              <ProgramIdentifier>string</ProgramIdentifier>
              <OffsetFromProgramStart>duration</OffsetFromProgramStart>
              <RequestInterval>duration</RequestInterval>
            </StreamInformation>

<KeyRotationOffsetFromProgramStart>duration</KeyRotationOffsetFromProgramStart>
            <EncryptionType>NotEncrypted or Aes128Bit</EncryptionType>
            <KeyId>string</KeyId>
            <KeyData>base64Binary</KeyData>
            <TimeToNextKeyRotation>duration</TimeToNextKeyRotation>
            <ProtectionSystemSpecificHeaderBoxes>
              <ProtectionSystemSpecificHeaderBoxContents>
                <ProtectionSystemId>string</ProtectionSystemId>
                <Data>base64Binary</Data>
              </ProtectionSystemSpecificHeaderBoxContents>
            </ProtectionSystemSpecificHeaderBoxes>
          </StreamProtectionInformation>
          <StreamProtectionInformation>
            <StreamInformation>
              <ProgramIdentifier>string</ProgramIdentifier>
              <OffsetFromProgramStart>duration</OffsetFromProgramStart>
              <RequestInterval>duration</RequestInterval>
            </StreamInformation>

<KeyRotationOffsetFromProgramStart>duration</KeyRotationOffsetFromProgramStart>
            <EncryptionType>NotEncrypted or Aes128Bit</EncryptionType>
            <KeyId>string</KeyId>
            <KeyData>base64Binary</KeyData>
            <TimeToNextKeyRotation>duration</TimeToNextKeyRotation>
            <ProtectionSystemSpecificHeaderBoxes>
              <ProtectionSystemSpecificHeaderBoxContents>
                <ProtectionSystemId>string</ProtectionSystemId>
                <Data>base64Binary</Data>
              </ProtectionSystemSpecificHeaderBoxContents>
            </ProtectionSystemSpecificHeaderBoxes>
          </StreamProtectionInformation>
        </StreamProtectionResponses>
        <CustomData>string</CustomData>
      </AcquirePackagingDataResult>
    </AcquirePackagingDataResponse>
  </soap:Body>
</soap:Envelope>
```

The following sections describe the response fields in more details.

```
<AcquirePackagingDataResponse
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols">

  <AcquirePackagingDataResult
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/AcquirePackagingData/v1.0">
```

Elements	Requirement	Info
<StreamProtectionResponses>	Required	Collection of <StreamProtectionInformation /> elements. The set of packaging information returned that is associated with the stream information in the request.
<CustomData>	Optional	The service can choose to include service specific data in the response. **(string)**

<StreamProtectionInformation>

The StreamProtectionInformation element represents the encryption key information used to encrypt the content from a specific period of time in a stream.

Elements	Requirement	Info
<StreamInformation>	Required	The information from the challenge identifying the fragments to protect **(StreamInformation)**
<EncryptionType>	Required	The encryption type for this fragment (NotEncrypted or Aes128Bit) **(string)**
<KeyId>	Required if EncryptionType is different from "NotEncrypted"	The key identifier of the content key. **(string)**
<KeyData>	Required if EncryptionType is different from "NotEncrypted"	The content key to be used to encrypt the content. **(base64 encoded binary)**
<ProtectionSystemSpecificHeaderBoxes>	Required if EncryptionType is different from "NotEncrypted"	Collection of <ProtectionSystemSpecificHeaderBoxContents /> elements. The data to be stored in the protection system specific header box (base64 encoded binary); contains embedded licenses for live TV scenarios. The fields in the ProtectionSystemSpecificHeaderBoxContents structure are described below

<KeyRotationOffsetFromProgramStart>	Optional	The offset for the current policy rotation interval from program start. This is only required if the response contains multiple policy rotation intervals for a given request interval. **(XSD Duration)**
<TimeToNextKeyRotation>	Optional	Indicates the time until the next key rotation, with up to millisecond granularity. **(XSD Duration)** If not present, the encryption machine must make the determination. If present, the encryption machine should select the next fragment boundary occurring within the specified interval. For example, if the period is 10 minutes from stream timestamp 0 and keys can be rotated at 9:59 and 10:01, the encryptor must pick 9:59.

<ProtectionSystemSpecificHeaderBoxContents>
The ProtectionSystemSpecificHeaderBoxContents element represents the content of a Protection System Specific Header box (pssh) in an ISO CENC (ISO/IEC 23001) protected content.

Elements	Requirement	Info
<ProtectionSystemId>	Required	The protection system that is used to generate license. **(GUID)**
<Data>	Required	The serialized header box containing leaf license. (base64 encoded binary)

Note: The **<KeyId>** element is a GUID given as a string value. Note that its representation in the related scalable leaf license is in a converted little endian format.

Note: PlayReady Server SDK uses .NET SoapDuration class to convert between xsd duration and Timespan. SoapDuration class assumes that 1 year = 360 days and 1 month = 30 days.